tinbergen *institute*

Solving large structured Markov Decision Problems for perishable inventory management and traffic control

René Haijema

Research Series

Universiteit van Amsterdam

Solving large structured Markov Decision Problems

for perishable inventory management and traffic control

ISBN 978 90 361 0101 1

Cover design: Crasbon Graphic Designers bno, Valkenburg a.d. Geul

This book is no. **444** of the Tinbergen Institute Research Series, established through cooperation between Thela Thesis and the Tinbergen Institute. A list of books which already appeared in the series can be found in the back.

Solving large structured Markov Decision Problems

for perishable inventory management and traffic control

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. D.C. van den Boom
ten overstaan van een door het college voor promoties
ingestelde commissie,
in het openbaar te verdedigen in de Agnietenkapel
op vrijdag 19 december 2008, te 14:00 uur

door

René Haijema

geboren te Sijbekarspel.

Promotiecommissie

Promotores:	Prof. dr. ir. J. van der Wal
	Prof. dr. N. M. van Dijk
Overige leden:	dr. ir. I. J. B. F. Adan
	Prof. dr. ir. R. Dekker
	Prof. dr. M. R. H. Mandjes
	Prof. dr. C. Th. Smit Sibinga
	Prof. dr. H. C. Tijms

Faculteit Economie en Bedrijfskunde

To my parents and Claire

Dankwoord

Bij het schrijven van een proefschrift zijn vele personen direct dan wel indirect betrokken. In de eerste plaats is het een product van de auteur, de promovendus. Echter de inhoud en kwaliteit van het proefschrift wordt eveneens bepaald door zijn begeleiders, de promotoren. In die rol ben ik Jan van der Wal en Nico van Dijk zeer dankbaar. Jan zorgde ervoor dat, ondanks mijn nevenaanstelling als docent, ik me voldoende op het onderzoek kon richten. Door zijn immer kritische houding werd ik regelmatig verleid alternatieve onderzoekspaden te bewandelen. Ik heb goede herinneringen aan Jan als begeleider; in de persoonlijke sfeer herinner ik me nog goed de fietstocht van Claire en mij naar zijn huis in Frankrijk. Nico wil ik bedanken voor zijn enthousiasme ten aanzien van het door ons verrichte onderzoek, hetgeen geresulteerd heeft in een aantal artikelen en de implementatie van de ontwikkelde programmatuur bij een van de bloedbanken. Ook wil ik Nico bedanken voor de persoonlijke gesprekken die we, ondanks een immer volle agenda, gevoerd hebben.

Ten aanzien van de implementatie van de software gaat mijn dank ook uit naar Nikky Kortbeek, die na mijn vertrek bij de UvA, samen met Fred Jansma van Incontrol de software verder geschikt gemaakt heeft voor implementatie. De implementatie was uiteraard niet mogelijk zonder de inzet en het enthousiasme van managers en staf van de Nederlandse bloedbanken, Sanquin.

Tevens ben ik Cees Smit Sibinga, van de World Health Organization, UMC Groningen en Sanquin Consulting services, dankbaar voor de waardering van ons werk en voor de boeiende gesprekken over de uitdagingen waarmee bloedbanken geconfronteerd worden. Tevens ben ik Cees zeer erkentelijk voor het internationaal verkondigen van ons werk in de 'bloedwereld', o.a. op congressen van de AABB (de Amerikaanse Associatie van Bloedbanken). Uiteindelijk heeft dit geresulteerd in een prestigieuze publicatie in het internationaal toonaangevende tijdschrift *Transfusion*.

Mijn dank gaat uit naar Sandjai Bhulai voor zijn presentatie en onze gedachtewisseling over de decompositie van multi-dimensionale Markov beslisproblemen. Medewerkers en studenten van de Civil Engineering department van de University of Florida (UF) in Gainesville, dank ik voor de uitnodiging een presentatie te geven. In het bijzonder dank ik David Hale van de McTrans center (UF, Gainesville) voor zijn uitleg omtrent de modelleermogelijkheden in Transyt-7F. Tevens ben ik Luc Prinsen van Goudappel Coffeng dankbaar voor de introductie in de praktijk van 'traffic engineering'. Ook ben ik de Dienst Infrastructuur en Vervoer van de Gemeente Amsterdam dankbaar voor de reflectie op ons werk. De gesprekken met de mensen uit het veld, bevestigen de relevantie van het onderzoek en in welke richting er toekomstig onderzoek gedaan kan worden.

Naast andere vakgenoten die ik heb mogen ontmoeten op congressen, wil ik expliciet mijn directe collega's van de Universiteit van Amsterdam (UvA) en van Wageningen Universiteit & Research center (WUR) bedanken. Mijn UvA collega's wil ik bedanken voor de leuke tijd die ik bij hen had, voor de samenwerking in onderwijs en onderzoek, en vooral ook voor de taarten die traditiegetrouw verorberd werden wanneer er weer een artikel gepubliceerd was. In het bijzonder wil ik Assil bedanken, waarmee ik als mede-AIO de afgelopen jaren veel leuke en goede gesprekken gevoerd heb, die veelal ook nog nuttig waren voor de voortgang van ons beider AIO onderzoek. Het was voor mij dan ook logisch om Assil te vragen als paranimf.

Met mijn vertrek bij de UvA sluit ik een lange periode aan de UvA af, die begon met mijn studie Operations Research. Bij deze wil ik een ieder bedanken met wie ik heb mogen samenwerken tijdens de vele student-assistentschappen die ik heb mogen vervullen. Het bestuur van de Afdeling Kwantitatieve Economie wil ik bedanken voor de goede organisatie, de geboden mogelijkheden en de prettige sfeer.

Mijn collega's aan de Wageningen Universiteit wil ik bedanken voor de prettige sfeer in mijn nieuwe werkomgeving. Het is leuk om met hen samen te werken en zo nu en dan eens naar De Kater te gaan. Tevens wil ik met name Jack van der Vorst bedanken voor zijn begrip en de mogelijkheid om mijn proefschrift deels in mijn nieuwe functie in Wageningense tijd af te schrijven.

En uiteraard bedank ik alle vrienden en collega's (uit discretie laat ik de persoonsnamen maar even achterwege...) met wie ik regelmatig 'vergaderd' heb in De Krater. Ook bedank ik het barpersoneel van de Krater dat de logistiek van het bier verzorgde, waardoor deze 'vergaderingen' altijd succesvol zijn verlopen. Deze aangename afleiding heeft het schrijven van dit proefschrift vergemakkelijkt.

Ook de barbecues, gourmetetentjes en weekendjes weg met Herman, Klaas, Cees, Katya, Maurice, Ines, Arnold, Cecilia, Jasper en Roya zijn van onmetelijk belang geweest als tegenwicht tegen de serieuze taak waar dit proefschrift het resultaat van is. Net zo waardevol zijn de leuke afspraken en etentjes met Micha en Yvonne, en Irene en Roel. Basil en Katrien bedank ik voor de stevige concerten die we proberen ieder jaar bij te wonen. Music keeps the spirit alive! In dit kader, waardeer ik ook zeer de optredens van de band Nasty Noise (Herman, Toon, Peter, Bert, Dirk en Fred), die van zich zal laten horen op het promotiefeest. Jan Hontelez wil ik bedanken voor zijn stelling dat het helemaal niet gek zou zijn om Claire te vragen als paranimf.

Mijn broer en schoonzus, Johan en Silvia, wil ik bedanken voor ons oom-en-tante-schap. Mijn nichtjes en neefje (Romy, Maikel, en Demi) doen mij beseffen dat het belangrijk is om je af en toe weer kind te voelen. Ook wil ik mijn schoonfamilie (Jacques, Annie, Maurice en Nancy) bedanken voor hun getoonde interesse en begrip voor onze drukte.

De meeste steun en liefde die ik de afgelopen jaren heb ontvangen dicht ik toe aan mijn ouders en aan mijn aanstaande vrouw, Claire. Ik heb een fors beslag moeten leggen op Claire's begrip vanwege de druk en drukte die het schrijven van een proefschrift met zich meebrengt. Haar spontaniteit en liefde zijn de kwaliteit van dit proefschrift ten goede gekomen. Het doet me dan ook goed om haar als paranimf aan mijn zijde te hebben ten tijde van de promotie. Nu de mijlpaal van het voltooien van het proefschrift bereikt is, kijk ik er naar uit om samen naar de volgende mijlpaal, onze bruiloft, toe te leven.

Het laatste woord wil ik richten aan mijn ouders: "Pa en ma, ik ben jullie zeer dankbaar voor jullie onvoorwaardelijke steun, zorg en liefde. Ik kan geen betere ouders wensen. Zonder jullie had ik het niet gekund."

Contents

Chapter 1

Introduction, outline and preliminaries

1.1 Introduction and outline

1.1.1 Large structured Markov decision problems

Many optimization problems arising in practice involve sequential decision making and can be formulated as a Markov decision problem (MDP). Solving MDPs numerically is however restricted to problems that have a relatively small state space, of say at most a few million states. An optimal solution, often called an optimal strategy or an optimal policy, prescribes a best action to take in each individual state that may occur. Many MDPs arising in practice tend to have too many states, due to the dimensionality of the state space. An optimal solution can then not be computed in reasonable time. Fortunately, often an approximate solution will do and might even be favored.

Constructing an approximation algorithm requires a good intuition and insight in the problem under consideration. The problem often exhibits some structure, which can be very helpful in solving the problem. A general approach of how to exploit the problem structure does not exist for all problems, since the structure may be problem specific. In this thesis, we illustrate how the formulation of a problem as an MDP helps to construct numerical solutions. Simulation turns out to be crucial as an evaluation tool.

1.1.2 Two applications

We focus on two different optimization problems:

I. the production-inventory management of perishables; in particular that of blood platelet pools, and

II. the dynamic control of traffic lights.

Production-inventory management of perishables

In the first optimization problem, we consider perishable products with a fixed maximal shelf life of m periods, say $m = 5$ days. One is interested in optimal production volumes, or order quantities, that balance the occurrence of outdating and shortages. Ordering too many products, results in excessive outdating after m periods. Ordering not enough products results in shortages; depending on the context unmet demand is considered as lost sales or is backlogged, eventually it may require an emergency order.

We study a perishable inventory problem that arises at blood banks and hospitals, which keep blood products in stock to meet the uncertain demand. Blood banks place production orders and hospitals place replenishment orders to keep their inventory at a safe level. Most inventory managers have difficulty in making the trade-off between the risk of shortages and outdating, since demand is uncertain and the ethical and economic impact of shortages and outdating is great.

So-called *blood platelet pools* (BPP) are thrombocyte concentrates that have a very short shelf life of only 5 to 7 days. The production of a single BPP requires platelets of 5 donors, expensive material and laboratory test to guarantee safe transfusions. This makes a BPP the most expensive blood products. Nevertheless, 10-20% of the BPP production becomes non-transfusable, because the transfusion date expires.

When solving this problem of finding good order sizes, a number of complicating aspects has to be acknowledged. We deliberate on them in the first part of this thesis, starting with Chapter 2. Without going into the details here, the state description shall contain a description of the number of so-called platelet pools of a specific age-category. Each age-category corresponds to a dimension of the state space. The resulting number of states, which is the product of the possible number of states in each dimension, becomes very large when applying real data. For example, suppose the maximal shelf life of a product is 5 periods (days), and at most 10 products are in stock of each of the 5 age categories, then the stock state consists already of 10^5 states.

In the practice of the Dutch blood banks, the state space is much larger and a number of complications have to be dealt with, such as the distinction of blood groups. A straightforward MDP approach is thus doomed to fail. But an approximate solution is of general interest, since it may reduce shortages and outdating of platelet pools significantly. The quality of the pools deteriorate over time. For some category of patients, very young platelet pools are strongly preferred to extend the period until the next transfusion. Therefore, we also study several issuing policies that prescribe which pools are selected from stock to meet the demand.

The structure of the perishable inventory problem allows to aggregate different states into a single state: e.g. multiple pools of different blood groups may be aggregated into batches. This reduces the state space significantly and thus we are able to solve the problem numerically. By simulation one may detect which states are visited most frequently and which actions are taken in these states. This way we may detect some structure in the thus obtained optimal strategy, maybe only after aggregating the age-categories. In a number of cases, we show that the optimal strategy turns out to resemble a simple rule. As we will show in Part I, there are a number of obstacles to take; such as the problem that the aggregation of states affects the transition law from one state to a next state. Therefore, the approach needs to be validated by an extensive sensitivity study, from which we obtain insights under what conditions the approach is valid.

Making a nearly optimal trade-off between high availability and low outdating figures is of general importance. Examples are found not only at blood banks and in hospitals, but also in the food-processing industry amongst other.

Dynamic control of traffic lights

The second problem that we study in this thesis, is the minimization of the expected waiting time per car at a traffic light. The optimization problem can be formulated as a MDP. For a detailed problem description, we refer to Chapter 6 in the second part. The state description contains, amongst other, the number of queued cars waiting at each stopping line. When cars approach the intersection from four directions and each direction consists of three separate lanes for respectively right turning, left turning and 'through' traffic, then the state space consists already of (at least) 12 dimensions. When at any time maximal 9 cars are queued at each lane, then the state space contains already (at least) 10^{12} states. Given the number of states, the MDP is intractable for most real intersections. An approximate solution is thus needed.

As will be shown in Part II, we may add additional structure to this problem by imposing a special strategy that allows the decomposition of the state space. Through a single policy improvement step, we obtain an approximate solution that appears to perform quite well: it results in low average waiting times compared to other control policies. The approach for a single intersection allows the inclusion of information on arrival times of near future car arrivals. Applying the approach to networks of intersections shows significant reduction in the overall average waiting times compared to the several existing control policies.

1.1.3 Simulation

Throughout this thesis (discrete) simulation plays three important roles:

- Due to the high dimensionality and the scale of the problem, the obtained approximate policies can be evaluated by simulation only.
- One may obtain a profound insight into (nearly) optimal strategies, by studying simulation results.
- Simulation enables to bridge the gap between the real system and the abstract mathematical models. By simulation one may validate in great detail the use of simplified decision models and approximate solutions.

1.1.4 Outline

In the next sections of this first chapter we discuss some preliminaries on Markov decision problems (MDP) and how to solve them numerically by Stochastic Dynamic Programming (SDP). In addition, we explain how numerical solution procedures can be exploited to evaluate a Markov chain (MC) of a specific (fixed) strategy. The concept of relative values of states is discussed, and finally we show how (numerically computed) relative values can be used in a one-step policy improvement algorithm. Readers not interested in, or already familiar with, these topics may skip the remainder of this chapter and may find the overview in Table 1.1 helpful in reading the thesis.
The remainder of the thesis is divided into two parts related to the two applications that are discussed. Part I consists of Chapters 2 to 4, and addresses the production-inventory management of perishables with a focus on blood platelet pools. Chapter 2 provides a detailed problem description and an extensive review of the history of perishable inventory models. In Chapter 3 a combined SDP-Simulation approach for solving the

Table 1.1: Outline of thesis

1	Introduction and Overview of this thesis
PART I	PERISHABLE INVENTORY MANAGEMENT
2	Perishable inventory theory
3	Combined SDP-Simulation approach
4	Extensions
PART II	DYNAMIC CONTROL OF TRAFFIC LIGHTS
5	Inroduction
6	Single intersection in isolation
7	Using arrival information
8	Arterials and Networks of intersections
9	Epilogue
	Appendices
	Bibliography
	Summary in English and Dutch

production-inventory problem at blood banks is introduced and applied to a case study, including a detailed sensitivity study. In Chapter 4 the problem of interest is extended to include non-stationary periods and fixed order costs, which may apply to hospitals. Despite the technical details, the results reported in this first part are of interest to managers of perishable inventory, and blood bank managers in particular, next to Operations Researchers.

Part II consists of Chapters 5 to 8 and introduces an MDP approach for controlling traffic lights. In Chapter 6, we explain how the waiting times at single intersections in isolation can be reduced by a one-step policy improvement algorithm. In Chapter 7, the approach is extended to include arrival information. Finally, in Chapter 8, we apply the approach to a network of intersections. In Chapter 9 we conclude the thesis by postulating some conclusions and discussing to what extent the two approaches can be generalized. After the Appendices, one finds summaries of the thesis in English and in Dutch.

Notations – Throughout this thesis some notations and conventions are in use as summarized in Appendix A. In some occasions, we allow ourselves using different notations (sometimes for typographical reasons and sometimes to avoid confusion).

1.2 Preliminaries on Markov Decision Problems

In this section we will present the main modeling assumptions of a Markov Decision problem (MDP), we introduce some solution procedures and discuss the computational complexity. We restrict the discussion to discrete time MDPs, in which the time is slotted into time intervals of fixed length, called slots. For a more general and more accurate discussion of MDP theory we refer to text books such as [14], [68], [120] and [144].

1.2.1 Modeling MDPs

Suppose one is responsible for controlling a dynamic system in discrete time. Therefore decisions, or actions, are taken at fixed discrete points in time, say at the start of each slot. Just before an action is selected, the state of the system is inspected. As a result of an action and a number of uncertain events, the state of the systems changes according to a probability distribution. The probability distribution captures the uncertainty of events that may happen between two decision epochs.

Related to a slot are (expected) direct or immediate costs, which depend on the state at the start of that slot, and possibly on the decision taken and the transition probabilities. In addition to direct costs, direct rewards may be included. In this discussion, we consider costs only; rewards can be seen as negative costs. The objective is to minimize the costs over some finite horizon, or to minimize the average costs per slot over an infinite horizon. An alternative, which we do not consider, is the minimization of the total discounted costs by discounting future costs.

Below we deliberate on the components of an MDP: the states, decisions, transitions, and related probabilities and costs.

States – The above sequential decision problem is a Markov decision problem if the so-called Markov-property or memoryless property holds: the state description should contain all relevant information (about the past and the current situation) for taking a decision, for incurring (expected) direct costs and for deriving the transition probabilities. No information on previously visited states is needed, other than the information included in the description of the current state.

We focus on MDPs where the state space is multi-dimensional. States will be represented by a vector $\mathbf{x} = (x_1, ..., x_m)$, where x_i can take values in $\mathcal{X}_i$. We allow a state space $\mathcal{X} = \prod_{i=1}^{m} \mathcal{X}_i$ to be an infinite countable state space, but we will truncate it to a finite

one such to allow for the numerical computation of an optimal state-dependent policy. (In finite horizon problems, the slot or stage number may be included in the state space, resulting in different mutually exclusive subspaces: one for each time slot.)

Decisions – Every decision epoch, say at the start of each slot, a decision a is selected out of the action set $\mathcal{A}$, which may be state-dependent: $\mathcal{A}(\mathbf{x}) \subseteq \mathcal{A}$. In some states $\mathbf{x}$ only one single action is feasible, then the action set $\mathcal{A}(\mathbf{x})$ consists of one element only. (In principle, when multiple decisions are to be taken simultaneously, the action space may be multi-dimensional. In this thesis, we will not deal with multi-dimensional action spaces.)

Transitions – For determining the transition from one state to a next state, one needs to specify the course of events that may happen during a slot. The state of the system, $\mathbf{x}$, is inspected at the start of a slot. Based on the observed state $\mathbf{x}$, an action a is selected. Next, any certain and uncertain events are modeled in a pre-specified order. Finally, one computes the resulting state $\mathbf{y}$ at the end of the slot, or say at the start of the next slot. When action a is taken in state $\mathbf{x}$, only states $\mathbf{y}$ in $\mathcal{X}(\mathbf{x}, a) \subseteq \mathcal{X}$ can be reached. Different sequences of events that may happen can result in the same next state $\mathbf{y}$.

At the start of the next slot, the system will be in state $\mathbf{y}$ with probability $P_{\mathbf{x},\mathbf{y}}(a)$, given (state, action)-pair $(\mathbf{x}, a)$. These transition probabilities are computed from known probability distributions of the uncertain events that may happen during a slot.

Direct costs – The total costs to incur over a planning horizon, can be broken down in cost directly related to the successive slots. When at the start of a slot action a is taken in state $\mathbf{x}$, then direct costs $c(\mathbf{x}, a, \mathbf{y})$ are incurred when the next state is $\mathbf{y}$. As multiple uncertain events may result in the same transition from state $\mathbf{x}$ to $\mathbf{y}$, the direct costs are the expected costs over all possible events related to that transition.

As multiple states y can be visited after taking decision a in state $\mathbf{x}$, one often computes the expected direct costs: $\mathbb{E}C(\mathbf{x}, a)$. The expected direct costs $\mathbb{E}C(\mathbf{x}, a)$ to incur when action a is selected in state $\mathbf{x}$ depend on the transition probability $P_{\mathbf{x},\mathbf{y}}(a)$, and is computed as follows:

$$\mathbb{E}C(\mathbf{x}, a) = \sum_{\mathbf{y} \in \mathcal{X}(\mathbf{x},a)} c(\mathbf{x}, a, \mathbf{y}) P_{\mathbf{x},\mathbf{y}}(a). \tag{1.1}$$

Criterion – The planning horizon of the MDP is either finite or infinite. In finite horizon MDPs, one usually minimizes the expected cost over the planning horizon (with or without discounting). In infinite horizon models, one minimizes either the long-run average cost or the expected total discounted costs.

1.2.2 Solving an MDP by Stochastic Dynamic Programming

When the planning horizon is finite, say N time slots, then an MDP is solved by (discrete) stochastic dynamic programming (SDP) in N stages. In general the time slots may differ in length, but we assume them to be of fixed identical length. When the planning horizon is infinite, a long-run-optimal policy can be derived by a Successive Approximations (SA) algorithm, which is in essence an SDP algorithm. We briefly present the finite and infinite horizon problems and discuss how they are solved by SDP. We limit the discussion to unichain cases. From the discussion of an SA algorithm, we learn about the complexity of solving infinite horizon MDPs.

Another solution procedure called Policy Iteration (PI) is discussed in Section 1.2.3. A third solution procedure, which we do not discuss, is based on Linear Programming (LP). For a more complete and rigorous treatment of this subject we refer to standard textbooks (such as [68], [120], and [144]).

Finite horizon problem

In a finite horizon problem one often labels the time slots backwards from 1 to N, and so do we in the discussion below. These labels refer to the N stages or iterations of the solution process by SDP. Usually the action space, the expected direct costs or the transition probabilities are stage dependent. An optimal strategy will then be non-stationary: i.e. optimal actions depend on the stage number $n \in \{1, 2, \ldots, N\}$. We add the slot number n to the state description. State $(n, \mathbf{x})$, thus denotes state $\mathbf{x}$ at the n-th stage of the solution process. Let $\mathcal{X}$ be the set of all state vectors $\mathbf{x}$ that are feasible at one ore more stages. The state space at the start of slot n, denoted by $\mathcal{X}(n)$, is then a subspace of $\mathcal{X}$. Further, we change the notation $\mathbb{E}C(\mathbf{x}, a)$ into $\mathbb{E}C(n, \mathbf{x}, a)$, and the transition probability will be $P_{n,\mathbf{x},\mathbf{y}}(a)$.

For n running from 1 to N, one computes for all $\mathbf{x} \in \mathcal{X}(n)$ the following dynamic programming (DP) recursion:

$$V(n,\mathbf{x}) = \min_{a \in \mathcal{A}(n,\mathbf{x})} [\, \mathbb{E}C(n,\mathbf{x},a) \; + \sum_{\mathbf{y} \in \mathcal{X}(n-1|\mathbf{x},a)} P_{n,\mathbf{x},\mathbf{y}}(a)\, V(n-1,\mathbf{y}) \,], \tag{1.2}$$

where $\mathcal{X}(n-1|\mathbf{x},a)$ is the set of states that can be reached from state $\mathbf{x}$ when action a is selected at the n-th stage. Thus, $\mathcal{X}(n-1|\mathbf{x},a)$ is a subset of the state space at the start of the next slot (= slot $n-1$): $\mathcal{X}(n-1|\mathbf{x},a) \subseteq \mathcal{X}(n-1)$.

Element $V(n,\mathbf{x})$ of the value vector equals the total expected costs over the last n slots of the planning horizon, when starting slot n in state $\mathbf{x}$ and only optimal actions are taken until the end of the horizon. This includes terminal costs, which will be zero when $V(0,\mathbf{y})$ is set to 0 for each state $\mathbf{y}$.

An optimal strategy π^* prescribes for each stage $n \in \{1,2,\ldots,N\}$ and every state $\mathbf{x} \in \mathcal{X}(n)$ an optimal action according to Equation (1.2).

Remark – Note that the numerical computations of (1.2) can only be executed when the state space is finite. When this is not natural for the underlying problem, one needs to truncate the state space. Then one accepts a modeling error, for which one may need to compensate in the direct cost structure. Anyway, the truncation should not affect the optimal policy too much.

Infinite horizon – the Successive Approximation (SA) algorithm

An infinite horizon MDP is usually assumed to be *stationary* and *aperiodic*, i.e. the cost structure, transition probabilities, and the action space are identical in each stage. As a result the optimal policy is stationary. Solving an infinite horizon MDP is done in a way very similar way to solving an finite horizon MDP. An optimal policy can be computed by a Successive Approximations (SA) algorithm, which is based on SDP. The next four steps constitute an SA algorithm for determining a policy that minimizes the (long-run) average cost per slot for a stationary, aperiodic, infinite-horizon problem.

Successive Approximations algorithm

Step 1. Set $n = 0$, $\forall \mathbf{x} \in \mathcal{X}: \ V_0(\mathbf{x}) = 0$, and $\epsilon =$'small', i.e. ϵ is small compared to the (unknown) minimal long-run average cost per slot, that relates to the values that $\mathbb{E}C(\mathbf{x}, a)$ can take,

Step 2. $n = n + 1$, and
Compute for all $\mathbf{x} \in \mathcal{X}$:

$$V_n(\mathbf{x}) = \min_{a \in \mathcal{A}(\mathbf{x})} [\ \mathbb{E}C(\mathbf{x}, a) \ + \sum_{\mathbf{y} \in \mathcal{X}(\mathbf{x},a)} P_{\mathbf{x},\mathbf{y}}(a)\ V_{n-1}(\mathbf{y})\ ] \tag{1.3}$$

and store a minimizing action a: $\pi(\mathbf{x}) = a$,

Step 3. Compute:

$$U_n = \max_{\mathbf{x} \in \mathcal{X}}[V_n(\mathbf{x}) - V_{n-1}(\mathbf{x})], \quad \text{and} \quad L_n = \min_{\mathbf{x} \in \mathcal{X}}[V_n(\mathbf{x}) - V_{n-1}(\mathbf{x})],$$

Step 4. If $span(\mathbf{V}_n - \mathbf{V}_{n-1}) \equiv U_n - L_n \geq \epsilon$ then return to Step 2,
else $N = n$ and π approximates a (nearly) optimal stationary policy π^*. The average costs g^π can be approximated by $\frac{L_N + U_N}{2}$, and will be ϵ-close to the optimal average cost level g^*, as $g^\pi - g^* < \epsilon$.

Explanation – An optimal policy π^* is thus approximated by successively extending the planning horizon n by one time slot per iteration. Since the length of the planning horizon is indefinite, the range of n is in principle unbounded: $n \in \{1, 2, ...\}$. After a large number of, say N, iterations, the difference between two successive value vectors $\mathbf{V}_n$ and $\mathbf{V}_{n-1}$ has converged, with ϵ-precision, to a vector of constants equal to the long-run average costs per slot, g.

The convergence of the vector $\mathbf{V}_n - \mathbf{V}_{n-1}$ is checked by the so-called span, which is defined in Equation (1.4):

$$span(\mathbf{V}_n - \mathbf{V}_{n-1}) = \max_{\mathbf{x} \in \mathcal{X}} [\ V_n(\mathbf{x}) - V_{n-1}(\mathbf{x})\ ] - \min_{\mathbf{x} \in \mathcal{X}} [\ V_n(\mathbf{x}) - V_{n-1}(\mathbf{x})\ ]. \tag{1.4}$$

The SA algorithm stops when $span(\mathbf{V}_n - \mathbf{V}_{n-1}) < \epsilon$. The convergence of the span happens exponentially fast, but the speed of convergence, i.e. the required number of iterations N, depends on the problem data.

The optimal strategy π^* is approximated by the policy π obtained in the last iteration. When not stored in Step 2, the optimal actions follows from another improvement step:

$$\pi^*(\mathbf{x}) \approx \arg \min_{a \in \mathcal{A}(\mathbf{x})} [\, \mathbb{E}C(\mathbf{x}, a) \; + \sum_{\mathbf{y} \in \mathcal{X}(\mathbf{x},a)} P_{\mathbf{x},\mathbf{y}}(a)\, V_N(\mathbf{y}) \,] \tag{1.5}$$

The minimal long-run average costs are estimated by that under policy π, i.e. by g^π, for which an upper bound U_N and a lower bound L_n is available: $U_N - g^\pi < \epsilon$ and $g^\pi - L_N < \epsilon$.

Periodicity – If the problem is by nature periodic, e.g. in the action space, in the cost structure, or in the transition probabilities, with period D, any feasible policy is periodic. Then the state space is divided into D periodic classes. Under any policy the long-run average cost per slot is then class dependent: the long-run average costs per slot in class $d \in \{1, 2, \ldots, D\}$ is g_d. The long-run average costs over D successive slots is $\sum_{d=1}^{D} g_d = D \cdot g$, where g is the *gain* of the underlying MC, which equals the (overall) long-run average cost per slot.

The limit $\lim_{n\to\infty} span(\mathbf{V}_n - \mathbf{V}_{n-1})$ does not exist, as the differences between $\mathbf{V}_n$ and $\mathbf{V}_{n-1}$ change periodically with n. Therefore, Step 2 in the SA algorithm is executed D times before the algorithm proceeds to Step 3 and Step 4. In Step 4, $span(\mathbf{V}_n - \mathbf{V}_{n-D})$ is checked rather than $span(\mathbf{V}_n - \mathbf{V}_{n-1})$.

Even when the problem itself is aperiodic, an optimal policy may behave virtually periodic. Then the convergence of the SA algorithm is slow but it will converge after many iterations. If one suspects the optimal policy to behave virtually periodic with period D, one may speed up the solution process by checking $span(\mathbf{V}_n - \mathbf{V}_{n-D})$ rather than $span(\mathbf{V}_n - \mathbf{V}_{n-1})$.

Computational complexity of an SA algorithm

Solving an infinite horizon problem by successive approximations requires the computation of (1.3) for all states $\mathbf{x} \in \mathcal{X}$. The number of state vectors is $|\mathcal{X}|$ and depends on the number of values each element of the state vector $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ can take: $|\mathcal{X}| = \mathcal{O}(\prod_{i=1}^{m} |\mathcal{X}_i|)$. For each state $\mathbf{x}$, all $|\mathcal{A}(\mathbf{x})|$ feasible decisions are considered. Given any state-action pair $(\mathbf{x}, a)$, there are $|\mathcal{X}(\mathbf{x}, a)| \leq |\mathcal{X}|$ 'next-states' $\mathbf{y}$ to consider.

Hence, an upper bound on the total number of transitions computed in a single iteration is $|\mathcal{A}| \cdot |\mathcal{X}|^2$. The number of iterations N depends on the problem (and data) under consideration. The overall computational complexity is $\mathcal{O}(|\mathcal{A}| \cdot |\mathcal{X}|^2)$.

1.2.3 Solving an MDP by Policy Iteration (PI)

An alternative way of solving an MDP is by a *Policy Iteration (PI) algorithm*. The algorithm is presented below. For a more detailed discussion we refer to text books such as [68] and [120]. The algorithm consists of the following four Steps I to IV.

Policy Iteration algorithm

Step I. Select a stationary initial policy π_0, and set $\pi = \pi_0$ and set $n = 1$,

Step II. Value determination step
In the n-th iteration of the PI algorithm, evaluate the MC for policy π: i.e. derive a so-called relative value vector $\mathbf{v}^\pi$ and the gain, g^π, which must satisfies at least the following set of equations (in matrix notation):

$$\mathbf{v}^\pi + g^\pi \cdot \mathbf{1} = \mathbf{c}^\pi + \mathbf{P}^\pi \mathbf{v}^\pi, \tag{1.6}$$

where $\mathbf{c}^\pi$ has elements $\mathbb{E}C(\mathbf{x}, \pi(\mathbf{x}))$ and $\mathbf{P}^\pi$ has elements $P_{\mathbf{x},\mathbf{y}}(\pi(\mathbf{x}))$.

Note that a solution $(\mathbf{v}^\pi, g^\pi)$ to (1.6) is not unique. Therefore one may set an arbitrary element of $\mathbf{v}^\pi$ to zero, say $v^\pi(\mathbf{x}) = 0$. Alternatively, one may add the constraint $\mathbf{P}^\pi \mathbf{v}^\pi = \mathbf{0}$, such that the relative values $\mathbf{v}^\pi$ are the so-called *bias terms* $\widetilde{\mathbf{v}}^\pi$ of policy π.

Step III. Policy improvement step
Compute, $\forall \mathbf{x} \in \mathcal{X}$:

$$\pi_n(\mathbf{x}) = \arg \min_{a \in \mathcal{A}(\mathbf{x})} [\, \mathbb{E}C(\mathbf{x}, a) \; + \sum_{\mathbf{y} \in \mathcal{X}(\mathbf{x},a)} P_{\mathbf{x},\mathbf{y}}(a)\; v^\pi(\mathbf{y}) \,] \tag{1.7}$$

and set $\pi_n(\mathbf{x}) = \pi(\mathbf{x})$, whenever possible.

Step IV. If $\pi_n = \pi$ then stop as $\pi^* = \pi$ is an optimal policy,
else set $\pi = \pi_n$, and $n = n + 1$ and return to Step 2.

Remark – The core of the algorithm is Step II and Step III. In an alternative PI algorithm, Steps II and III are interchanged. Then the algorithm starts with an initial value vector $\mathbf{v}$, which may or may not be related to a (possibly unknown) strategy. Next, a policy improvement step is executed using the initial value vector $\mathbf{v}$. In the third step, the gain and relative values are determined for the resulting policy.

Computational complexity of a PI algorithm

The computational complexity of a PI algorithm is set by the value determination step. In that step the relative values of states for a given policy π, as well as the gain g of the underlying MC, are determined. Equation (1.6) represent a system of $|\mathcal{X}|$ equations in $|\mathcal{X}| + 1$ unknowns. To obtain a unique solution, one of the relative values can be set to zero (or any other constant). Then a solution can be obtained by applying a Gaussian elimination, or any other procedure to solve a set of linear equations.

Usually, the number of iterations that a PI algorithm requires is much smaller than that of an SA algorithm, but each iteration takes more time. When Gaussian elimination is used the computational complexity is $\mathcal{O}(|\mathcal{X}|^3)$. When the transition matrix $\mathbf{P}$ is sparse, more efficient procedures to solve the equations can be used. In the Section 1.3.2, we discuss how an approximate solution can be found by an SA algorithm with actions fixed to those set by policy π.

1.2.4 Computation times and memory usage

The numerical example below shows that deriving an optimal policy for a multi-dimensional MDP can be very time-consuming. For an SA algorithm we discuss both the (evolution of the) computation time as well as the memory usage.

Computation times – a numerical example

Consider an MDP with a 5-dimensional state space $\mathcal{X} = \prod_{i=1}^{5} \mathcal{X}_i$, and $|\mathcal{X}_i| = 20$ for all $i = 1, 2, \ldots, 5$. The total number of states is thus $20^5 = 3.2\,10^6$. When $|\mathcal{A}(\mathbf{x})|$ equals 15 for all states $\mathbf{x}$, then 15 possible actions are considered per state. When all $|\mathcal{X}|$ states can be reached from any state $\mathbf{x}$, then the number of transitions to evaluate per iteration is $15 \cdot (3.2\,10^6)^2 \approx 1.5\,10^{14}$. Fortunately, for many MDPs the transition matrix is sparse and well-structured: given action a, the number of states that can be reached from state $\mathbf{x}$ is much less than $|\mathcal{X}|$. If $|\mathcal{X}(\mathbf{x}, a)| = 30$ for all state-action pairs $(\mathbf{x}, a)$, then the number of

transitions to consider is $30 \cdot 15 \cdot 3.2\,10^6 \approx 1.4\,10^9$. Suppose that a computer evaluates 1 million transitions per second, then evaluating all transition of a single iteration would take 1,400 seconds, or 23 minutes.

The total computation time depends also on the number of iterations N is problem specific and further depends on the required precision in the evaluation of $span(\mathbf{V}_n - \mathbf{V}_{n-1}) < \epsilon$. When $N = 30$ iterations are required, then in total about $4.2\,10^{10}$ transitions are to be considered. An optimal solution is thus obtained after 11.5 hours, which may or may not be acceptable depending on the context of the problem. Anyway, we are interested in solving problems of this size and much larger.

Evolution of computation time

When $|\mathcal{X}_i|$ doubles for all $i = 1, 2, \ldots, 5$, then the state space $|\mathcal{X}|$ will become 32 times as large. When $|\mathcal{X}(\mathbf{x}, a)|$, $|\mathcal{A}(\mathbf{x})|$ and N are unaffected, the computations will thus be 32 times as long. This dramatic increase of the number of states when the problem size increases is often referred to as the *curse of dimensionality*. Whether a problem can be solved in a reasonable time depends mainly on the total number of states. One should thus carefully investigate the structure of a problem to limit the number of states.

Memory usage

Another limitation may be the available memory. Fast RAM memory is used to store and update the value vectors, and to retrieve the transition probabilities and the direct costs. In 2008, the RAM memory on most computers was limited to a few Gb. For each state $\mathbf{x}$ two reals, $V_n(\mathbf{x})$ and $V_{n-1}(\mathbf{x})$, each of size 8 byte, are to be stored. In addition, one may store the transition probabilities in RAM memory. For the example given with $3.2\,10^6$ states one needs 51.2Mb fast RAM memory to store the two value vectors. (When $|\mathcal{X}_i|$ doubles for all $i = 1, 2, \ldots, 5$, then almost 1.7Gb RAM is to be used.)

In addition to the value vector, another 11 Gb RAM memory is needed to store all $1.4\,10^9$ transition probabilities $P_{\mathbf{x},\mathbf{y}}(a)$ for the given example. Although RAM memory is easily extended, memory consumption is an obstacle for large problems. To save RAM memory, one may decide to compute the transition probabilities online, rather than storing them all. Then the available computation time is usually the true bottleneck.

Parallel computing

Distributing the work over multiple PC gives a relatively small relieve of the computational trouble. For many problems encountered in practice, the state space is far too large to solve the problem to optimality, even when the computational burden is spread over many computers. For example, when a 5-dimensional problem can just be solved on a single PC, then a similar problem that is twice as large in each of the 5 dimensions requires $2^5 = 32$ computers connected to each other to solve the problem in the same time (ignoring any time lost in the communication between the 32 PCs).

1.2.5 Conclusion

For many multi-dimensional Markov decision problems, the curse of dimensionality necessitate the development of approximate solutions. Exploiting any structure present in a problem may be crucial for a successful solution of the problem. In the next section, we present a one-step policy improvement algorithm, which may be helpful in finding an approximate solution.

1.3 One-step policy improvement approach

1.3.1 Motivation

When computing an optimal policy for a stationary (infinite horizon) MDP is too time-consuming, one is interested in an approximate solution. Therefore, we discuss in this section how to evaluate and to improve an initial policy π through a single policy improvement step.

For a number of problems an approximate solution can be derived by execution of a single iteration of the PI algorithm starting with a well-structured policy π as the initial policy. That is, first a relative value vector $\mathbf{v}^\pi$ is computed for this initial policy π. Next, an improved policy π' is derived by executing a single policy improvement step:

$$\pi'(\mathbf{x}) = \arg \min_{a \in \mathcal{A}(\mathbf{x})} [\, \mathbb{E}C(\mathbf{x}, a) \; + \sum_{\mathbf{y} \in \mathcal{X}(\mathbf{x},a)} P_{\mathbf{x},\mathbf{y}}(a) \; v^\pi(\mathbf{y}) \,]. \tag{1.8}$$

Such an approach is called a *one-step policy improvement approach*. Since the value determination step is the computationally most complex step in a PI algorithm, one looks for well-structured policies for which the relative values of states can be computed or approximated quickly. There are roughly two ways of approximating the relative values: one way is by solving the balance equations in an analytical way, another way is by the numerical computation of a relative value vector.

In either approach one looks for relative value functions or vectors that are separable. Therefore additional modeling assumptions may be needed to allow the decomposition of the state space of the MDP into subspaces. In Chapter 5, we give a number of examples in which a one-step policy improvement approach is followed after the decomposition of the state space.

For some problems closed-forms expressions or approximations of the relative value function exist for well-structured policies. The numerical computation of the relative values of states can be done under more general assumptions, but the initial policy must allow the decomposition of the state space. This means that, in a numerical approach, we look for a policy for which the relative values of states can be computed for each subspace in isolation of the other states. Therefore one has to analyze the MC for policy π using an SA algorithm with actions set to those specified by π. In the remainder of this section we discuss the numerical computation of relative values of states under any policy π.

1.3.2 Markov chain analysis for a fixed policy π

The SA algorithm of Section 1.2.2 can be used to evaluate a (MC) of a stationary policy π, when the action space $\mathcal{A}(\mathbf{x})$ in Equation (1.3) is replaced by a single action $\pi(\mathbf{x})$ as shown in (1.9):

$$\forall \mathbf{x} \in \mathcal{X}: \quad V_n^\pi(\mathbf{x}) = \mathbb{E}C(\mathbf{x}, \pi(\mathbf{x})) + \sum_{\mathbf{y} \in \mathcal{X}(\mathbf{x}, \pi(\mathbf{x}))} P_{\mathbf{x},\mathbf{y}}(\pi(\mathbf{x}))\, V_{n-1}^\pi(\mathbf{y}) \tag{1.9}$$

The long-run average costs or the gain g^π of the policy π follow from the difference between successive value vectors. When π is periodic with period D, the gain can be computed from $\lim_{n\to\infty} \left(\mathbf{V}_n^\pi - \mathbf{V}_{n-D}^\pi\right) = D \cdot g^\pi$. When π is aperiodic, then $D = 1$. To execute a policy improvement step the relative values of states need to be computed. As this is not straightforward for periodic policies, we split the discussion on how to compute a relative value vector: first for an aperiodic policy, next for a periodic policy.

Relative values of states for an aperiodic policy

If π is aperiodic, it can be shown [68] that the difference between the value vector $\mathbf{V}_n^\pi$ and the average cost over n slots converges to a vector $\widetilde{\mathbf{v}}^\pi$ that contains the bias terms of the states under policy π. The bias term $\tilde{v}^\pi(\mathbf{x})$ gives the relative appreciation of state $\mathbf{x}$:

$$\tilde{v}^\pi(\mathbf{x}) = \lim_{n\to\infty} (V_n^\pi(\mathbf{x}) - n \cdot g^\pi). \tag{1.10}$$

As such $\widetilde{\mathbf{v}}^\pi$ is a relative value vector that can be used in (1.8).

When policy π is stationary and aperiodic, the limit in (1.10) exists, for a prove see [68]. The bias terms $\widetilde{\mathbf{v}}^\pi$ satisfy (1.6), and so will do the vector $\widetilde{\mathbf{v}}^\pi - k \cdot \mathbf{1}$ for any k, as the relative value vector $\mathbf{v}^\pi$ is unique up to a constant. Hence when the number of iterations N of the SA algorithm is sufficiently large, $\mathbf{V}_N^\pi$ can be used directly in (1.8) to obtain an improved policy π':

$$\pi'(\mathbf{x}) = \arg\min_{a \in \mathcal{A}(\mathbf{x})} \left[\, \mathbb{E}C(\mathbf{x}, a) \;+ \sum_{\mathbf{y} \in \mathcal{X}(\mathbf{x}, a)} P_{\mathbf{x},\mathbf{y}}(a)\, V_N^\pi(\mathbf{y}) \,\right] \tag{1.11}$$

Policy π' is either an improvement over policy π or both π and π' are optimal. Note that $V_N^\pi(\mathbf{z}) - V_N^\pi(\mathbf{x})$ is the total expected cost difference between starting an horizon of N slots, in state $\mathbf{z}$ compared to starting in state $\mathbf{x}$.

Relative values of states for a periodic policy

When the policy π is periodic with period D, the state space is divided into D subspaces, or classes. The classes are numbered $1, 2, \ldots, D$ and visited in that order in a cyclic way, i.e. after class D, class 1 is visited. The long-run average cost to incur in class d is g_d^π. The gain of the MC for policy $^\pi$ is $g^\pi = \frac{1}{D}\sum_{d=1}^{D} g_d^\pi$.

The limit in (1.10) does not exist for periodic policies, as, when state $\mathbf{x}$ is in periodic class d, $\lim_{n\to\infty}\{\, V_{nD+1}^\pi(\mathbf{x}) - V_{nD}^\pi(\mathbf{x})\,\} = g_d^\pi$, which is usually not equal to the gain g^π.

Therefore, no matter how large N is, $\mathbf{V}_N^\pi$ cannot act as an approximation of the value vector. In words, when a policy is periodic, the periodic class at which the system is at the end of the horizon depends on the class at which the horizon starts. Since the expected costs in a slot after many iterations is class dependent, a fair comparison of states based on $\mathbf{V}_N^\pi$ is only possible for states that are in the same class.

If states of different classes are to be compared, then one has to revert to a value vector that satisfies the Equations (1.6). We claim that for any policy π and for N sufficiently large, the vector

$$\mathbf{v}^\pi = \lim_{n\to\infty} \frac{1}{D}\sum_{d=0}^{D-1}\left(\mathbf{V}_{n+d}^\pi - (n+d)\cdot g^\pi\cdot\mathbf{1}\right). \tag{1.12}$$

can be used for this purpose. A proof will be given Appendix B. In fact, when N is sufficiently large, the vector $\dfrac{1}{D}\displaystyle\sum_{d=0}^{D-1}\mathbf{V}_{N+d}^\pi - k\cdot\mathbf{1}$ can be used for any constant k. When k is set to zero, an (improved) policy π' is set by:

$$\pi'(\mathbf{x}) = \arg\min_{a\in\mathcal{A}(\mathbf{x})}\,[\,\mathbb{E}C(\mathbf{x},a) \;+\; \sum_{\mathbf{y}\in\mathcal{X}(\mathbf{x},a)} P_{\mathbf{x},\mathbf{y}}(a)\;\frac{1}{D}\sum_{d=0}^{D-1} V_{N+d}^\pi(\mathbf{y})\,]. \tag{1.13}$$

The intuitive logic behind this definition is the following. A fair comparison of states in different periodic classes requires to consider each class as the last class of a planning horizon, independent of the initial class. Thus one needs to average over D successive horizon lengths. A formal derivation of Equation (1.12), as a correct definition of the relative value vector to compare states in periodic Markov chains is given in Appendix B.

1.3.3 Conclusions

In a one-step policy improvement approach one executes a single policy improvement step for a given initial policy π. The applicability of the algorithm relies on with whether one is able to find a policy π for which a vector, function or expression can be obtained that approximates the relative values of the states under π. If such a policy π with related relative values is found, an improvement over π is guaranteed [144], unless π itself is already optimal. Whether the improvement is substantial is however not guaranteed, but strongly depends on the problem and the selected policy π.

For the derivation of the relative values of a policy one may revert to an SA algorithm, but due to its computational complexity we may not be able to compute these values for problems with large states spaces. In some cases the state space can be decomposed but only when the problem and the initial policy are well-structured, such that the computational problem can be decomposed as well. We deliberate on the decomposition of multi-dimensional MDPs in Chapter 5. Although not guaranteed, a one-step policy improvement approach may yield a very good, or even nearly optimal strategy.

Part I

Production-Inventory management of perishables

Chapter 2

Perishable inventory theory

In this first part of the thesis we investigate the optimal production and inventory management of a perishable product with a (short) fixed life time. The focus of the study is on blood platelet pools (BPPs), which are the most expensive and most perishable blood products. BPPs are produced at blood banks and transfused to patients in hospitals. In Section 2.1 the platelet production problem (PPP) at blood banks is described in detail. Although some of the problem characteristics are typical for BPPs, one may generalize the problem to other perishable products, such as food as we will discuss in the epilogue in Chapter 9. The focus of the study is on determining (nearly) optimal production volumes. A similar problem arises in an inventory setting, where nearly optimal order quantities need to be set.

Next to the (production) order size, the production-inventory levels are controlled by the so-called issuing policy, which specifies the way in which demand is met. Therefore an overview of standard ordering and issuing policies is presented in Section 2.2. In Section 2.3 the history of modeling (perishable) inventory management is reviewed and many references are given. Finally, in Section 2.6, we formulate research questions to be answered in the next chapters.

2.1 Problem of interest

2.1.1 The platelet production problem (PPP)

The description of the PPP that is given in this section comes from conversations and collaborations with employees of Sanquin, the Dutch association of blood banks. We refer in particular to communications with M. Beun (Physician, [16]), C. Th. Smit Sibinga (Director Sanquin consulting services and member of the WHO, [133]) and B. Hinloopen (logistic manager at Sanquin, division North East, [66]). The Dutch case is to some degree representative for the case at a number of developed countries.

Demand for BPPs

Platelets are of live-saving importance. As small particles in the bloodstream, platelets prevent internal and external bleeding by recognizing and 'repairing' damaged blood vessels. Platelet's quality deteriorates rapidly even inside the blood stream, but most people's production of platelets at the bone marrow is sufficient to retain a safe level. Nevertheless after a major bleeding caused by a trauma or a surgery, patients may temporarily have a lack of platelets. These patients need to be transfused with platelet pools of *any* age up to the maximal shelf life. This demand category is henceforth called the *any-demand*, since there is no strong preference with respect to the age of the pools. The any-demand comprises about a third of the total demand.

The remaining two thirds of the demand is set by patients who suffer from a platelet function disorder. To keep the number of platelets in their blood at a safe level, these patients are transfused frequently (at least once a week) at the Hematology department with *'young'* pools that preferably are at most three days old, counted from the day the pools are added to stock. This demand category is referred to as the *young-demand*. Transfusing older platelet pools is allowed but less desirable.

The distinction between the any-demand and the young-demand is not clearly made at all blood banks. For other blood banks the young-demand may be a smaller portion of the total demand. Although part of the transfusions at the hospitals are scheduled, the demand volume is highly uncertain to blood banks.

According to the eight different blood groups as set by the four ABO-categories (A, B, AB and O) and the presence of the Rhesus-D factor (+ or −), one may distinguish eight different BPP products (A^+, A^-, etc.). Figure 2.1 shows the compatibility of the blood groups and the relative frequencies by which they are present in the Western population.

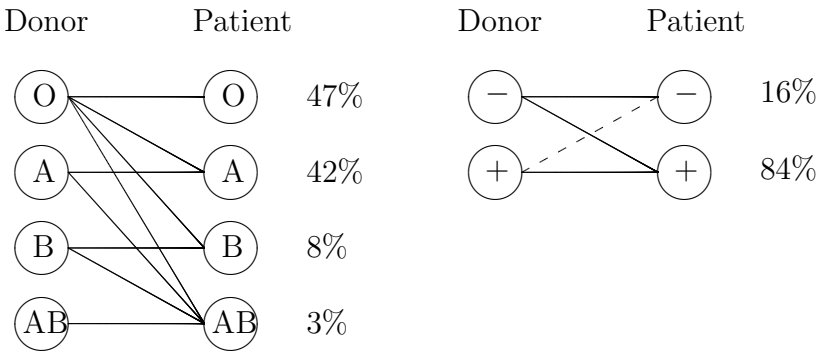


Figure 2.1: Compatibility of blood groups of donor and patient following the ABO-system and Rhesus-D system (+/−)

In Section 3.6 we deliberate on the compatibility of the blood groups. For now, observe that almost 90% of the Western population has blood group O or A and that the O^- donor is the universal donor that can help any patient.

Production process

Blood banks are responsible for the production and distribution of blood products in all sorts. To collect blood, appointments with voluntary donors are scheduled. Most commonly, a donor gives 500 *ml* of whole blood, but a minor part of the donors donates only specific blood components through the much more expensive apheresis technique. In The Netherlands, the fraction of platelet concentrates that comes from apheresis is about 10%. This fraction may be different in other countries, see [96]. We focus on the regular production of products from whole blood donations.

Figure 2.2 shows the product classes and the two processing steps. In a first processing step the red blood cells and plasma are filtered from a whole blood donation. The residual called, the *'buffy coat'*, contains a high concentration of blood platelets (or thrombocytes) and white blood cells. In a second processing step, the blood platelets are extracted from the buffy coats, since white cells might cause complications upon transfusion. The platelets of 5 donors of the same blood group are pooled together in a so-called *blood platelet pool* (BPP). A BPP corresponds to an adult platelet dose or one transfusion unit. Some patients need to be transfused with multiple BPPs. BPPs cannot be shared by patients.

The demand for whole blood donations is, in The Netherlands, dominated by the demand for red blood cell concentrates (RBCs) at hospitals. The residual plasma is used in plasma products and for the production of all kinds of medicines. The number of buffy coats, obtained as a side product of the production of RBCs, usually exceeds the demand for

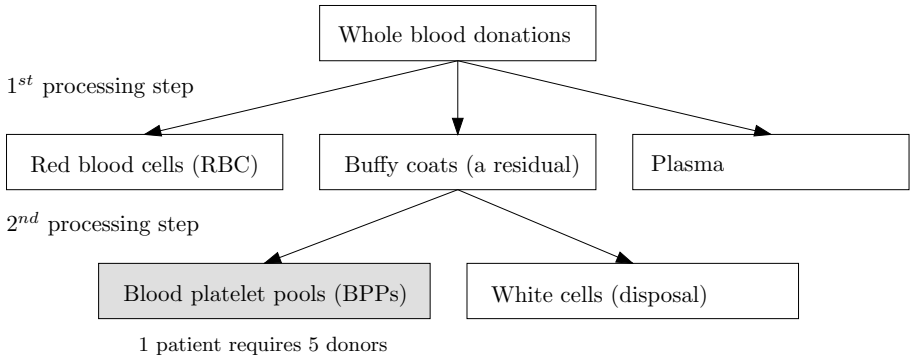


Figure 2.2: Product classes and processing steps at blood banks

platelet pools. As a consequence, on average only a third to a half of the buffy coats is used for the production of BPPs. Furthermore not all buffy coats are used, since production and storage capacity may be limited. Rarely the production of BPPs is hampered by a lack of whole blood donations.

To guarantee safe transfusions, the quality of the whole blood and the processed blood products are tested thoroughly. The whole process of collecting whole blood, processing BPPs and testing their quality takes a full day. The BPPs produced on day 0 are released at the start of day 1, say in the morning, and only then they become available for meeting the uncertain demand at the hospitals. To keep the overall outdating low, BPPs are primarily kept in stock at the blood banks and at a few large hospitals. Delivery of platelet pools happens daily in the morning and during the day upon request of the hospitals. In the Dutch case, each of the four blood banks serves about 20 to 30 hospitals.

In 2004, hospitals paid 458 euros for each BPPs they order at the blood banks, this includes all costs from collection, production, transportation and costs related organizational overhead. This price is set by the association of blood banks, Sanquin, in agreement with the Dutch government. The variable production costs are estimated at 100 to 150 euro. All other blood products are significantly less costly to produce.

Next to being the most expensive blood product, a BPP is also the most perishable blood product. Therefore blood bank managers are interested in a careful production-inventory control of BPPs. In 2004, the maximal shelf life of a platelet pool was only 5 days at most blood banks, counted from the day the pool is released and added to stock (one day after production). Current developments allow raising this maximum to 7 days. All other blood products can be kept in stock for weeks or even months.

Optimization problem

A blood bank's major concern is to produce safe blood products and to distribute these products to hospitals. Both in hospitals and at blood banks efficiency and quality of service get more and more attention. To improve efficiency, one should meet the demand at lowest costs. All costs that we consider in this study are (variable) costs related to the production and inventory management of BPPs.

For keeping a high quality of service, one should have enough pools in stock to meet the demand. Since the quality of BPPs deteriorates in time, the age of the pools upon meeting the demand influences the perceived quality of service. Owing to the demand uncertainty, a blood bank sometimes has to accept low stock levels every now and then. Occasionally a blood bank may even run out of stock. Then demand has to be met from another blood bank's inventory, e.g. by lateral transhipment, or it has to be produced instantaneously (through apheresis). A demanded BPP that cannot be issued from own stock is called a shortage. The variable shortage costs are related mainly to the costs of buying and transporting a BPP from another blood bank and to the loss of goodwill.

When demand is split into two age categories, demand for 'young' BPPs and demand for BPPs of 'any' age, the quality of service depends on the degree to which age preferences are met. Since the quality of a pool is (negatively) correlated with its age, issuing an older BPP than preferred is considered as a mismatch. A mismatch is thus a special type of shortage, namely a shortage of 'young' BPPs. The costs of a mismatch is considerably lower than the shortage costs, but to keep a high quality of service one may account for the risk of losing goodwill.

The occurrence of shortages and mismatches are penalized by introducing (fictitious) costs per BPP mismatched or short. For BPPs that become outdated, we have to accept the loss of labor hours and capital related to the production of the outdated BPPs. These costs are reflected in the so-called outdating costs. Further, minor variable holding costs are accounted for holding BPPs in stock.

From an Operations Research perspective, we are interested in an optimal (stock-age-dependent) ordering strategy, given the variable outdating, holding, shortage and mismatch costs, such that the *long-run average weekly cost* is minimized. Managers of blood banks and hospitals are interested in a practical rule for deriving nearly optimal (production) order sizes of BPPs. These two questions will be answered in this first part of the thesis.

2.1.2 A first modeling step: single-product model

Although a BPP can be of any of the eight blood groups, according to the ABO-system and the RhD- factor, the problem is often studied for a single, say universal, blood group. In the current practice, one produces-to-stock primarily BPPs of blood groups O and A, which fulfill virtually all demand. When BPPs of a specific group are needed, the blood banks commonly receives such a request well before the scheduled transfusion date.

Many perishable inventory studies concentrate on so-called single-product models. Although a great part of the applications concern blood products, only a few studies acknowledge the distinction of blood groups. Given the compatibility scheme, a single-product approach, in which one focusses on BPPs of a single blood group, seems to be a reasonable approximation of the current practice. However, a quantitative justification of such an approximation is not provided in the literature.

2.2 Common issuing and ordering policies

The performance of any perishable inventory system, in terms of outdating and shortages, strongly depends on how the issuing and ordering of products is controlled. The ordering policy prescribes *when* and *how-many* to order at the production department or at an external supplier. The issuing policy prescribes which items are taken from stock upon meeting the demand. To streamline the literature survey in Sections 2.3 and 2.4, we first give an overview of commonly used issuing and ordering policies.

2.2.1 Issuing policies

In the PPP the issuing of BPPs is controlled by the inventory manager. In a different perishable inventory setting, such as at a supermarket, customers may pick themselves the products from the shelf. The order in which products are taken from stock is then called the picking or dispatching order, which may vary between customers: some take an arbitrary product, while others select the freshest product on the shelf. Since our focus in the next chapters is on the PPP, we stick to the term issuing rule.

Inventory managers have some freedom in acknowledging any age-preferences communicated by the doctors in the hospitals. Nevertheless in practice, one tends to issue the oldest products in stock first. Issuing the oldest products in stock first is favored by inventory managers since it keeps outdating at a low level. This issuing policy is referred to as FIFO (First In First Out). Selecting the youngest products in stock first is called a LIFO (Last In First Out) issuing policy. LIFO issuing may contribute to a high quality of service, but it usually results in excessive outdating.

Similar to FIFO is FEFO (First Expired First Out), by which the products that will expire first are issued first. FEFO requires thus a ranking of all products in stock from 'fresh' to 'less-fresh'. Such a ranking is not available for BPPs, but can be approximated by looking at the age, or the residual shelf life of, BPPs. Regarding the 'freshness' of a BPP, the quality of a BPP is measured by the number of active platelets in a pool; this number can be estimated only by laboratory test, using a sample of the platelet concentrate. Since such an evaluation is laborious and expensive, FEFO has not been considered in a blood banking environment. Of course, the quality of a BPP is checked before a BPP is transfused, not only regarding the freshness but also to guarantee a safe transfusion.

Furthermore, demand may be categorized, and each category may prefer a different issuing

policy. In the PPP we distinct two demand categories, since for some patients one may more strongly prefer to transfuse them with young platelets than for other patients. In Section 3.3.1, we present a new issuing policy for one of the demand categories, which we call FIFOR and which is a mixture of FIFO and LIFO. Our FIFOR policy is closely related to the upward and downward substitution rule as presented by Deniz, Scheller-Wolf and Karaesmen [35]. More on these and other less commonly studied issuing rules can be read in Section 2.4.1.

2.2.2 Ordering policies

Stock-age-dependent policies versus stock-level-dependent policies

Given a (composite) issuing policy, one needs to specify when replenishment orders are placed and the number of products to order. An optimal ordering strategy takes all relevant information that is available into account. For the optimal ordering of perishables with a fixed shelf life, one should know the number of products of each age category, rather then solely the total number of products in stock. The optimal ordering strategy for perishables is a so-called *stock-age-dependent policy*: it has to take into account the ages of the products in stock.

Since stock-age-dependent policies are more complex to analyze, given the detailed description of the inventory, most studies are on so-called *stock-level-dependent policies*. A stock-level-dependent policy takes as input only the total number of products in stock, irrespective their ages. One may say that the perishable inventory problem is often approximated by the inventory problem of a non-perishable product.

Periodic versus continuous review models

Existing (stock-level-dependent) ordering policies are usually classified as either continuous or periodic review models. In continuous review models orders can be placed at any point in time: typical ordering moments are when the stock level changes, i.e. when demand is met or when outdated products are removed from stock. In periodic review models order may be placed periodically at fixed moments, say at the start of every period. (In some multi-item ordering models an order for item i may be placed every R_i-th period, but for single-product models, R_i is often equal to 1 and therefore not included in the model.)

In the PPP, the orders are set periodically at the start of each working day. In periodic review models one may formulate a dynamic program and study the functional equations. Then often demand is modeled as a continuous distributed variable, since the analysis would become too complicated for discrete distributed variables. Another approach to study the ordering problem, is by considering a continuous review model instead of periodic review.

Some well-studied ordering rules

To streamline the discussion of the relevant literature in the next sections, we introduce at this point some common ordering policies and notations. For a more detailed overview we refer to the standard text book on inventory management, such as [62], [130], and [8]. As the ordering policies in periodic review models and in continuous review models are quite similar, except for the moments of ordering, we discuss them together.

In the models in which the policies below arise, one commonly incurs holding costs, ordering costs, shortage (or backlogging) costs, and outdating costs. These costs are often taken proportional to, respectively, the number of products in stock, the order quantity, the number of products short (or backlogged), and the number of products that become outdated. In addition fixed order costs may apply per order. When the fixed ordering costs are positive, one may decide to order only when the stock levels are 'too low'.

Ordered products may arrive instantaneously, or after a fixed lead time, or after a stochastic lead time. When the lead time is positive, the order volume often depends on the *inventory position* rather than on the number of products that are actually in stock. The inventory position is the actual inventory level plus the stock-on-order minus all demand that is backlogged. When the lead time is zero, the stock position equals the stock level, which we denote by $x = \sum_{r=1}^{m} x_r$. The same holds in periodic review models when the lead time is one period and the products are added to stock before a next order is placed.

Order-up-to S policy – When the fixed ordering costs are zero, the order quantity may be set such that the inventory position is raised to a fixed level S. The order quantity is set to S minus the stock position. In case the products are added to stock instantaneously, the order-up-to level S may be called the *target inventory level*. When the lead time is positive, the target level may not be reached when the order products are added to stock, as demand may have decreased the stock levels. We refer to this policy as an *Order-up-to S policy*. Other commonly used names for the same policy are *Critical-number policy*, or *Total-inventory-to-S (TIS) policy*.

(s, S)**-policy** – When fixed ordering costs apply, a threshold s for ordering is included. Orders are placed only when the stock position is at s or below. Most commonly, the order size is then derived from a fixed order-up-to level S. When the stock position is above s, no order is placed. This policy is commonly called an (s, S) *policy*.

$(S-1, S)$ **policy** – A special case of the (s, S) policy is when $s = S-1$, this policy may be appropriate when the demand is for discrete products. For the ordering of perishable this so-called $(S-1, S)$ policy, is studied by [126] and [112] for cases with positive (stochastic) lead time and demands may be in multiple products at a time.

(s, nQ) **policy** – A last class of ordering policies that we discuss here, is the (s, nQ) policy: every ordering moment the inventory position is checked and when it is at or below s a quantity $n \cdot Q$ is ordered with $n \in \{1, 2, \ldots\}$ such that the new stock position falls in the interval $(s, s + Q]$. Q is the batch size for ordering, which may be set by the supplier: e.g. order per box or per pallet.

2.3 History of production-inventory models

Inventory control is a classical subject of study to Operations Researchers and practitioners. Practitioners are interested in preferably simple policies or tools that help them balancing their quality of service and the efficiency of their operations. Operations Researchers study inventory problems from a mathematical point of view and provide models and simple rules to help the practitioners running their business effectively and efficiently. Before presenting in the next section a detailed literature survey, we give an overview of the history of inventory theory.

The focus is on articles on the ordering of a single product in a single echelon of the supply chain. Endpoint of the discussion is the numerical computation of stock-age-dependent ordering policies. The literature is much more rich, but multi-product and multi-echelon problems are beyond the scope of this thesis.

The timeline in Figure 2.3 shows the evolution of the (scientific) research in inventory management. The timeline starts with the first mathematical models for non-perishable inventory and shows how the problem of interest is shifted and extended over time:

- from deterministic problems in the early stage,
- to stochastic problems in the fifties,
- to perishable inventory theory in the seventies,
- to the approach presented in this thesis.

Deterministic models – EOQ, Camp's formula, and ELS

In 1913, Harris ([59], [60]) introduced one of the first mathematical inventory models: the economic order quantity (EOQ). According to Erlenkotter [43], the EOQ formula became since 1922 also known as Camp's formula [25] and since 1934 as Wilson's economic lot size formula (ELS) [166]. The EOQ formula sets the optimal order quantity for a non-perishable product in a deterministic problem setting, based on a linear cost structure with both fixed and variable cost components. In many text books, e.g. [62], [131], the basic model and a number of extensions are explained in detail.

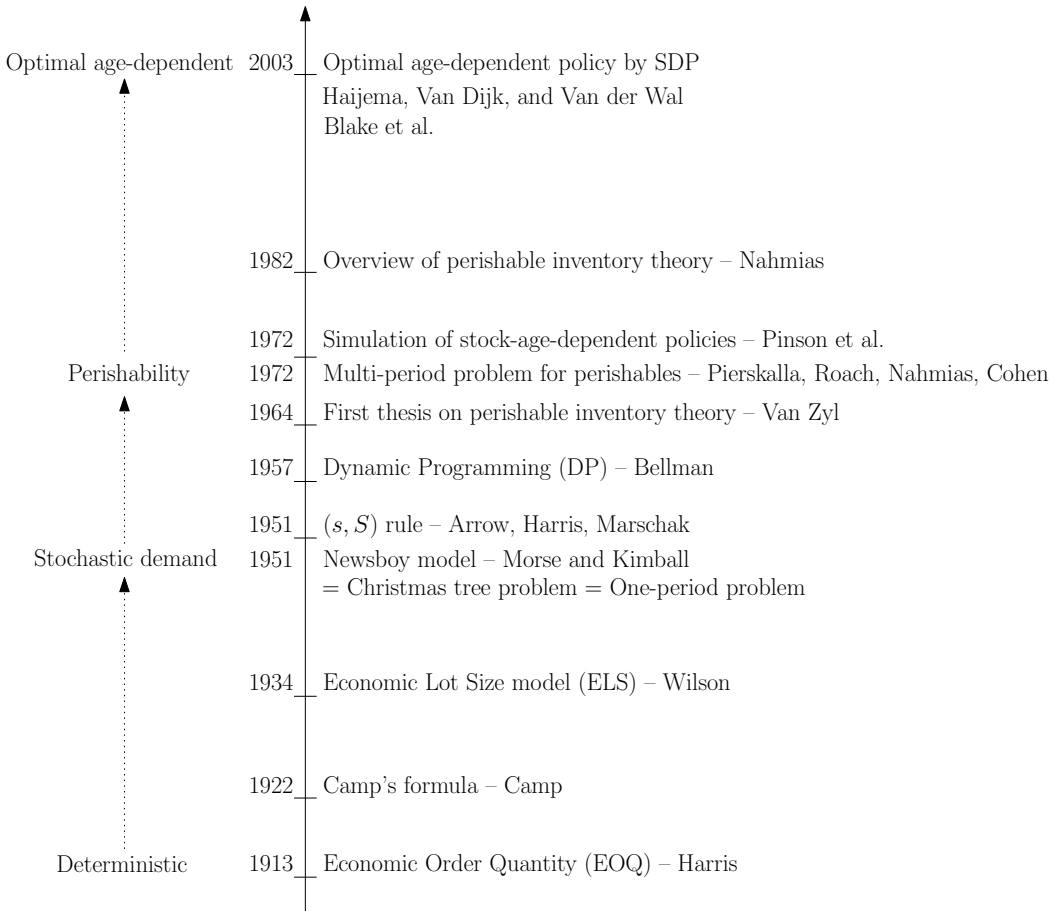


Figure 2.3: The history of inventory management: from simple deterministic models to stochastic perishable inventory models.

Stochastic single-period problems: Newsboy model, Christmas-tree model

One of the first articles on a stochastic inventory model dates back to the early fifties of the last century. In 1951, Morse and Kimball [95] present a one-period problem with stochastic demand. The model is often called the Newsboy model and is solved by the so-called Newsboy equation. Later it became also known as the Christmas tree model.

Kraiselburd [78] describes the very interesting history of the Newsboy problem. In fact, the model comes from classified research for the Navy during World War II. Many of the references to the work of Morse and Kimball recall military applications (mainly inventory problems) rather than the problem of a newsboy deciding on the number of newspapers to buy. Over the years the model is extended and applied to various one-period decision problems. The Newsboy model can be seen as a first stochastic inventory model for perishables with a fixed shelf life of one period.

Stochastic multi-period problems: (s, S) policies and Bellman's principle of optimality

Soon after the introduction of the Newsboy problem, the sequential ordering of a non-perishable products over multiple products was studied. The multi-period problems appeared to be significantly more complicated when fixed order costs apply. In 1951, Arrow, Harris and Marschak presented in Econometrica [7] the so-called (s, S)-type inventory control with stochastic demand. In 1952 and 1953, Dvoretzky, Kiefer and Wolfowitz ([40], [41]) studied the optimal parameter values of the (s, S) strategy. Most of the studies on multi-period problems rely on solving functional equations. The proof that the optimal policy for non-perishables is of the (s, S) type was first given in 1960 by Scarf [125], for a wide class of problems with a linear cost structure.

The study of the optimal control of inventories boosted after the introduction of Bellman's principle of optimality in the fifties. According to Arrow [5], he generalized some previous work on sequential analysis and announced a new technique called Dynamic Programming (DP). The recursive nature of solving a sequential decision problem was already addressed by Arrow, Blackwell and Girshick [6] in 1949. Anyway, DP became a new tool of finding optimal strategies for various decision problems ([14], [68]).

In 1960, Howard introduced a special class of sequential decision problems called Markov decision problems (MDP). In particular, Howard and Bellman deserve the credits for the successful introduction of this field of study. Nevertheless, due to the dimensionality of the state space, the multi-period perishable inventory problem was still considered too complicated to solve.

(Multi-period) Stochastic Perishable inventory models

Virtually all inventory studies before 1960 dealt with non-perishables or with perishables that last for one period only. The inventory management of perishables over multiple periods is considerably more complicated, since products are not only removed from stock to meet the (stochastic) demand but also because they become outdated. The study of inventory management of perishables started already in the sixties with the thesis of Van Zyl [154] and articles Pierskalla and Pierskalla and Roach. In the seventies the research of perishable inventory models flourished and resulted in papers by Cohen, Nahmias and Pierskalla and many others ([113], [114], [101], [29], [30], [97], and [98]). Most studies are on issuing policies and approximations of the outdating and shortages under specific ordering policies, or on finding good parameter values for order-up-to S and for (s, S) policies.

In some numerical studies, in which outdating and shortage figures are low, one claims simple rules to be nearly optimal. The question to what extend these rules are optimal in other cases remained unanswered. A number of features, that may be relevant in real-life perishable inventory problems, were not included in these studies.

Moreover optimal stock-age-dependent policies were hardly studied, in these (early) years. Two exceptions are the studies by Fries [47] and Nahmias [98]: in which some properties of the optimal solution are proved. Numerical approaches to compute optimal strategies for realistic sized problems were doomed to fail at that time. Solving the underlying MDP is seriously hampered by insufficient computer power and the curse of dimensionality in the state space: the number of possible states is too large to solve problems of real size. Only for case with a very short maximal shelf life and zero lead time some results are reported as will be discussed in Section 2.4.2.

Simulation of perishable inventory systems

In the seventies computer simulation rapidly became a more-and-more useful tool to analyze complex dynamic systems. Where the analytical models need to be relatively simple to allow its mathematical analysis, problems could now be studied at a reasonable realistic level by simulation. This different, say more practical, line of research was carried out by quite some researchers and practitioners. Simple order-up-to S rules were simulated to find out whether they yield low shortage and outdating figures under different issuing policies. For simple rules with one or two parameters search algorithms were developed to find optimal parameter values. Analytically obtained insights on the convexity of the cost function proved to be very useful in streamlining the optimization process.

Despite the modeling flexibility of simulation, there is always a tradeoff to make between the degree of realism to add to a model and the required accuracy of the outcomes, given the speed of the computer(s) on which the simulation is executed. This trade-off is inherent to simulation. Since computers were not as fast as nowadays, the level of detail in mimicking a real inventory systems was quite limited; in the early simulation studies the existence of different, limited-compatible blood groups is simply left-out. Most likely, this also explains why no periodic ordering policies are simulated with different order-up-to levels $S_1, \ldots, S_D$ of each of the D periods: i.e. the search for optimal values of all D parameters could have been too time-consuming.

In only a few studies stock-age-dependent ordering rules were developed that acknowledge the perishability of the products. For example, in 1972, Pinson, Pierskalla and Schaefer [115] test by simulation what-they-call *modified* order-up-to S rules. These rules were

developed based on common sense arguments to reduce shortages and outdating. Setting new stock-age-dependent rules requires insight into the perishable inventory problem. Simulation is one way to obtain these insights.

Optimal stock-age-dependent ordering policies for perishables

Although it was known that an optimal policy for ordering perishables should be stock-age-dependent, the actual computation of an optimal policy was not considered for perishables with a fixed shelf life of more than 3 days. At the start of the twenty-first century Blake et al. [19], and Haijema, Van der Wal, Van Dijk and Smit Sibinga ([56], [57], [58], [149]) numerically solved realistic (down-sized) Markov Decision Problems of blood platelets pools (BPPs) with a maximal shelf life of up to 7 periods. They apply independently an aggregation-disaggregation approach: individual BPPs are aggregated into batches. Blake et al. focusses on the cost reduction and the tractability at varying batch sizes. Haijema et al. study the structure of the optimal policy (with one and two categories of demand) and derive simple rules with nearly optimal parameter values. Their approach and results are reported in the first part of this thesis.

2.4 Classification of perishable inventory studies

In this section, we classify and review the perishable inventory studies that are most relevant in the light of solving the PPP.

Careful inventory management of perishables is of general interest and has had considerable attention over the past decades. A number of overviews and surveys are published: e.g. in 1984 by Prastacos [118], in 1991 by Raafat, and in 2001 by Goyal and Giri [51]. The most detailed and most cited review is that of Nahmias [100] in 1982. He reviewed almost one hundred articles on the theory of perishable inventory. A great part of the literature in this field is devoted to blood products.

In this section, we present a detailed review of the most relevant papers on inventory management of perishables with a fixed maximal shelf life. Further we limit the discussion to studies in which demand is stochastic. Since our approach – to be presented in the next chapters – is a methodological one, we organize our detailed review by classifying the studies according to the different methods and techniques that are in use. Some articles may fit into several categories.

The partly overlapping categories are:

- **Analytical studies**
  In analytical studies one primarily makes use of analytical techniques (such as calculus, algebra, functional equations, Markov chain analysis and sample path arguments). Closed-form expressions for the number of shortages and outdated products or for the cost function are only obtained for special, trivial cases of perishable inventory problems. In many analytical studies, one relies on numerical techniques in the end, i.e. for the evaluation of complex (multiple) integrals, for root finding and for estimating gradients of complicated analytical expressions. Nevertheless, the analytical studies may provide insights in how to model and approximate perishable inventory problems. Moreover, analytical techniques do not require data: with respect to this point they may be more general, although they less flexible in modeling the problem.

- **Numerical studies**
  In numerical studies numerical or computational procedures are developed and/or used to evaluate or to optimize the inventory control of perishables. By nature, numerical studies are data-driven and suitable for sensitivity analysis. A special class of numerical studies is the class of simulation studies.

- **Simulation studies**
  In simulation studies one mimics the system under some (fixed) policy. Computer simulation is a fast way of doing this and provides insight in the performance of that policy. In combination with a search algorithm, simulation can be used for optimization. It is important to note that simulation itself is a non-optimizing technique.

- **Experimental or empirical studies**
  Experimental or empirical studies report on current practice and on testing or implementing new policies in practice. Computer simulation experiments do not fall in this category. Since experimenting and empirical testing in practice is expensive and time-consuming, most studies do not belong to this class.

In the next subsections we discuss the most relevant papers that compose these four classes of studies.

2.4.1 Analytical studies

Analytical models are parameterized models and thus free of (case-sensitive) data. Often they are used to attain insights into properties of a control rule. For perishable inventory problems closed-form expressions for the number of shortages and outdating can only be derived for simplified models under specific well-structured ordering, such as the (s, S) policy. In most perishable inventory models, one studies the performance of ordering policies that stem from studies of non-perishables. Therefore, we start this review of analytical models with a brief description of the main results derived for non-perishable products. Next, we discuss some studies on perishable inventory management for both single-period and multi-period problems. Since the PPP is a multi-period problem, we pay particular attention to this class of problems.

Ordering policies for non-perishable products

In the early fifties, Arrow, Harris and Marschak [7] and Dvoretzky, Kiefer and Wolfowitz ([40], [41]) studied the (s, S) ordering policy, under a linear cost structure with holding costs, shortage costs and fixed order costs, in the context of stationary continuous (unknown) demand for non-perishable products. Since for the inventory control of non-perishable products, the age of the products is irrelevant, the stock is described by a single variable: the total stock level x.

In 1960, Scarf [125] showed, for the stationary (in)finite horizon problem with stochastic demand, that an (s, S) policy is optimal for ordering non-perishable products when fixed order cost and variable holding and shortage (or backlogging) costs apply. When the fixed order costs are zero an order-up-to S policy is optimal.

Analytical models for perishables

One may say that the study of ordering policies for perishables started with the so-called one-period problem. Soon thereafter the sequential ordering was studied for the so-called multi-period problems.

One-period problem

The one-period problem, often named the *'Newsboy' problem*, was first studied in the forties and fifties by Morse and Kimball [95], as already discussed in Section 2.3. Although the PPP is a multi-period problem, the Newsboy model deserves some attention in this review. In its simplest form the model deals with a newsboy who faces variable order costs c per product and a selling price p per newspaper sold. At the start of a period he decides on the number of newspapers to order, given the uincertainty in the demand and the fact that the unsold newspapers are worthless at the end of the day. Ordering a products costs $c \cdot a$, but yields p times the expected number of sold products.

The number of products sold depends on the demand distribution and is limited by a. Let $P[D = j]$ denote the probability that the demand over a single period is for j newspapers. The optimal order quantity is then the greatest value of a for which holds

$$\sum_{j=0}^{a-1} P[D = j] \leq 1 - \frac{c}{p}.$$

Clearly, the scope of the newsboy model is not limited to the ordering of newspaper, but is also applicable to the ordering of perishable and non-perishable products and services [78].

Multi-period problems for perishables

First, we discuss the properties of an optimal ordering policy for a perishable with fixed maximal shelf life. Next, an overview of commonly adopted, simplifying assumptions is given. Finally we report on most relevant studies on the ordering policy, on the ageing and outdating process, and on the issuing policy.

Structure of an optimal ordering policy for perishables

Multi-period problems for perishables with a fixed shelf life are much more complicated than one-period problems or order problems for non-perishables. Whereas the (s, S) is optimal for a broad class of non-perishable inventory problems, such a policy is in general not optimal for ordering perishables over multiple periods. This was first shown in 1975 by Fries [47] and Nahmias [98]. Both authors analyze independently the functional equations for a dynamic program of the fixed life perishable inventory problem. They show that an order-up-to S policy nor an (s, S) policy is optimal for the general case with maximal shelf life m days. Although the optimal order point might depend on a threshold that is stock-level-dependent, the order quantity should take into account the age distribution of the stock on hand.

In their studies they assume stationary, continuous, independent and identically distributed (i.i.d.) demand, which is met in a FIFO order. The following cost components are included: linear order costs and convex holding, shortage and outdating costs. To limit the dimension of the dynamic program the lead time is assumed to be zero periods. According to Nahmias, the work of Fries is a bit more complete compared to that of Nahmias. Fries obtained results for both finite and infinite horizon models with discrete demand.

Both authors agree upon the computational complexity due to the dimensionality of the dynamic program. Fries states that the direct computation of optimal policies generates further insight into the form of truly optimal policies. Nahmias concludes that future research should be on various simple non-optimal policies, next to the determination of approximate computational methods.

Simplifying assumptions

Given that an optimal ordering policy depends on the ages of the products in stock, its analysis is in general too complicated for an analytical approach. Therefore analytical

studies are limited to classes of well-structured strategies, e.g. (s, S) strategies, and additional assumptions are imposed to model the ageing and outdating processes of products in stock. In most studies stock-level-dependent strategies are analyzed, only a few studies consider well-structured stock-age-dependent policies.

Although analytical models are quite general, since they are free of data, the models can be treated only under simplifying assumptions. Commonly adopted assumptions concern:

1. The ordering policy:

   - Most analytical studies concern the evaluation of stock-level-dependent policies that are well structured, e.g. order-up-to S or (s, S) policies, which are proven to be optimal for non-perishables.
   - In a few studies one reports on stock-age-dependent policies that exhibit a special structure, which allows the mathematical analysis under strict assumptions, e.g. Tekin et al. [141].
   - A truly optimal stock-age-dependent policy for a general multi-period problem under stochastic demand is not achieved through analytical methods.

2. The maximal shelf life and the planning horizon:

   - If the (planning) horizon is shorter than or equal to the maximal shelf life, the analysis is simple when the initial stock is zero, since outdating only happens then at or after the horizon, see the work in the early sixties by Hadley and Whitin [52], [53] and Brown et al. [21].
   - A special case is the one-period problem or the Newsboy problem.
   - When the maximal shelf life is at most two (or three) periods, the problem can be formulated as a Markov chain with a low dimensional state space, see Van Zyl [154], Bulinskaya [24], and Nahmias and Pierskalla [101].

3. The ageing and outdating process:

   - The outdating process is in some studies approximated by assuming that outdating is a fixed or unknown fraction of the total stock on hand. When the shelf life is exponentially distributed, the decay of a product is a memoryless process (see Ghare and Shrader [49] and Emmons [42]). This way one avoids a high dimensional state space in a continuous-review model.

- In some other studies, one explicitly assumes that all products in stock have the same expiration date. This assumption may hold when replenishment only arrive when no products are left in stock. Alternatively, products keep a constant quality as long as they are packed in a sealed box, but the deterioration of products start when the box gets unsealed. When boxes are opened only when all old products are removed from stock, a model with only one age category suits ([85], [141]).

4. Other aspects: e.g. lead time and backlogging:

- In many analytical studies one assumes zero lead time, which implies that replenishment happens instantaneously. The state vector describing the number of products in stock of each age category at the start of a period is then of dimension $m-1$. When the lead time is exactly 1 period, the state vector consists of m dimensions. When the lead time exceeds k periods, one needs to include in the state description the outstanding orders placed in the last for $k-1$ periods. Assuming zero lead time thus simplifies the analysis. (The PPP is thus a bit more difficult at this point.)

- Introducing positive lead time makes the analysis much more difficult, since next to the age distribution also outstanding orders need to be included in the state description. When demand occurs one-by-one according to a Poisson process, and 1 product is ordered at a time, according to an $(S-1,S)$ policy, exact results for positive lead-time can be derived, see Schmidt and Nahmias [126] and Perry and Posner [112].

- Under some policies the analysis becomes easier when shortages are backlogged. However, in the PPP backlogging is not allowed as meeting the demand is critical as a patient's live could be at risk. In the PPP shortages are resolved by sending BPPs from another blood bank.

The review of analytic multi-period problems is divided into three parts. First, we report on studies with a focus on optimal ordering policies (mostly under a FIFO issuing rule). Next, a few articles on the ageing and outdating process are discussed. Finally, analytical studies on alternative issuing policies are reviewed.

Analytical studies on optimal ordering policies

Most analytical studies are on the (s, S) policy under continuous review. Where the analysis of inventory systems with zero lead time is already complicated, this certainly is the case if the lead time is positive. For a special well-structured strategy, the $(S-1, S)$ policy, and under the assumption of demand to happen by a Poisson process, Schmidt and Nahmias [126] and Perry and Posner [112] have obtained some exact results for the special class of $(S-1, S)$ policies with positive lead time. Under an $(S-1, S)$ policy an order is placed after each occurrence of demand or when products become outdated. The inventory position, which includes any outstanding orders, is thus kept at level S.

Whereas most of the analytical studies for non-perishables could be formulated as continuous review problems, a discrete time model is more natural for the control of perishables with a fixed maximal shelf life. Lian and Liu [85] presented in 1999 a discrete time model for (s, S) policies. All demand that cannot be met directly from stock is backlogged and new stock arrives instantaneously. Under these assumptions they study the (s, S) policy under periodic review with $s \leq -1$: a new order is thus placed only when no items are left in stock is. Consequently all products in stock are of the same age. This property simplifies the analysis significantly, hence a closed-form expression of the cost function can be obtained, which includes fixed order costs, holding costs, shortage costs and outdating costs. Since the properties of the resulting cost function are difficult to study analytically, they conduct a numerical study to investigate the impact on the costs of, what they call, the backlog level $s < 0$ and the order-up to level S.

Also Tekin et al. in [141] adopt the assumption that all products in stock are of the same age category. They compare a (continuous-time) modified lot-size-reorder policy with three parameters (s, Q, T) with an ordinary (s, Q)-policy. Under the (s, Q, T) policy, one orders Q units whenever the total stock level drops below s or when T periods have elapsed since the last time the inventory level was Q. The decay of the batch of Q products starts upon opening the batch for consumption. A batch is opened only when all products of the previous batch are either consumed or removed from stock because they have become outdated. Parameter T makes the rule stock-age-dependent. For the (s, Q, T) policy Tekin et al. derive analytically a cost function by applying renewal theory with the stock level Q as a renewal point. The cost components that they include are fixed order costs and variable holding and outdating cost. Instead of the usual direct costs for lost sales, they impose a service level constraint: the fraction of sales that is lost must be less than some prespecified value. Investigating the properties of the cost function requires an extensive numerical study from which they conclude that the stock-

age-dependent (s, Q, T) rule indeed performs better than the (s, Q) policy, especially in cases where outdating is an issue.

In 2002, Zhou and Pierskalla [171] analyze an inventory system with regular orders and emergency orders. Regular orders are less expensive than emergency orders, but have a much longer lead time. At the beginning of a cycle a regular order of Q products is placed. Whenever the inventory drops below s, an emergency order is placed to return the inventory to level s.

Studying the ageing and outdating process

A few perishable inventory studies focus on the ageing and outdating process of perishables.

In 1970, Pegels and Jelmert [111] track a single unit of blood over its maximal shelf life of 20 days. By a Markov chain approach with two absorbing states (expiration and transfusion of the unit), they derive the probability of outdating and the age distribution upon transfusion. The difference between FIFO and LIFO issuing is reflected in the transition probability: at FIFO the probability of transfusing a unit is higher when the unit is older. Through numerical evaluation of the Markov chains with a given transition matrix, the conditional probability of expiration of a unit is computed for each age category as well as the expected days until transfusion. The usefulness of this paper is however limited, as argued by Jennings and Kolesar in [71]. Most critical is Pegels and Jelmert's assumption that the transition probabilities are known. These probabilities should dependent on the demand distribution and the issuing and replenishment policy, but in their paper these components are not modeled explicitly. Even when the transition probabilities are as observed in practice, the contribution of the paper is marginal since the Markov chain analysis provides a seemingly exact way of computing of what is already known from the observations.

Another Markov chain approach that lacks an explicit controlled ordering policy is by Bar-Lev and Perry [9] in 1989. They analyze a discrete time Markov chain of stock levels for a perishable product (having a fixed maximal shelf life). The Markov chain analysis requires the assumption of stochastic supply instead of an explicit policy for ordering products and is therefore beyond the scope of this thesis.

When the replenishment is controlled rather than modeled as a stochastic process, one imposes a well-structured ordering policy and zero lead time to enable an analytical approach. Cohen [27] and Chazan and Gal [26] investigate simultaneously, yet indepen-

dently, the age distribution of the stock-on-hand under an order-up-to S rule. Whereas Cohen examines the continuous demand case, Chazan and Gal consider the discrete demand case and prove that the expected cost function as well as the expected outdating are convex in S. In addition, Chazan and Gal develop bounds on the expected outdating and generate closed-form solutions for expected outdates for an approximate model with Poisson demands. Usual assumptions in these analytical studies are i.i.d. demand that is met by FIFO issuing and that replenishments arrive instantaneously.

Analytical studies on the issuing policies

The most common issuing and ordering policies are reported in Section 2.2. Often the issuing policy can be kept very simple, since there is only a single category of demand. In case of multiple demand categories, as in the PPP, the issuing policy might be more complicated, e.g. a mixture between FIFO and LIFO.

In 1972, Pierskalla and Roach [114] consider a problem with multiple demand categories that are related to the different age categories. The demand for products of some specific age can be met by products of that age or by younger products, but not by older products. They show that issuing the oldest products available that meet the age preference, i.e. FIFO, is optimal under most common cost structures. For that reason one imposes a FIFO issuing rule in most studies. One of the few exceptions is Cohen and Pekelman [28] who study the less commonly studied LIFO issuing policy.

More recently, in 2004, Deniz, Scheller-Wolf and Karaesmen [35] investigate two types of order-up-to S rules under different issuing policies to meet demand for products of different age categories. The products of different ages are to some degree substitutes to each other, as reflected in the issuing rules, which they call substitution rules. Upward substitution means that the demand can be met in a FIFO way starting with the products of the preferred age category. Downward substitution implies that, when not enough products from the preferred category are available, older products are issued in a LIFO way. Full substitution is the combination of upward and downward substitution: demand for old products is met with products of that age or older, demand for 'young' is met by products of that age or younger. Meeting demand by products of another age category implies substitution costs for each demand category. Two order-up-to S rule are considered: one rule takes the total stock level as input, the other rule looks only at the number of 'new' products in stock. Using sample path arguments they discuss sufficient conditions on substitution costs that ensure dominance relations between the substitution policies.

Another interesting study is by Kopach et al. [76] in 2006. They model the perishable inventory problem as a queueing model with two types of customers: urgent versus non-urgent. Non-urgent customers can be blocked in case of low stock levels to reduce the chance of falling short to the urgent customers. In their queuing model the arrival stream relates to the supply of products, the queue length represents the total stock level, the outdating of products relates to impatience customers leaving the queue, and finally the demand for products is modeled by the server that picks customers from the front of the queue. The first come first served (FCFS) queueing discipline thus relates to the FIFO issuing policy. The mean service time represents the mean inter-demand time. The control problem they face is switching from a high speed server (that accepts both types of demand) to a slow speed server (that accepts only urgent demand) and vice versa, such that one minimizes shortages occurring when the queue is exhausted (out of stock) and outdating (represented by the loss of customers). Since this problem is quite different than ours, we do not report their work here in more detail.

Remark – In a number of studies, not reported in this section, both analytical and numerical techniques are adopted. One often reverts to numerical techniques to evaluate analytically obtained expression for the performance measures and the cost function. In particular, optimal parameter values can only be found through numerical experiments.

Conclusions

1. An optimal policy for perishables with a fixed maximal shelf life is stock-age-dependent.

2. In general, a stock-age-dependent policy is too complicated to study analytically.

3. Simplified analytic models are restricted to well-structured classes of policies, such as order-up-to S and (s, S) policies.

4. Nevertheless, analytic models provide insights into the ageing and outdating process and may approximate the cost function, but closed-form expression are hard to obtain.

5. For insights in the properties of the cost function, e.g. to determine optimal parameter values of an ordering policy, one often reverts to numerical techniques for evaluation.

2.4.2 Numerical studies

Shortly after the introduction of Dynamic Programming by Bellman, Dreyfus stated in 1957 the potential and the limitations of computational procedures for dynamic programming. Dreyfus [38] discussed how to deal with the computational limitations for deriving value functions. As one of the ways to overcome the computational difficulties, he explains the symbioses between sophistical mathematical analysis and digital computation. Mathematical analysis provides approximations and allows doing the computations more efficiently; it is not unusual for analysis to establish the nature of the function without being able to produce the function. In 1962, the book *'Applied Dynamic Programming'* [15] of Bellman and Dreyfus appeared. Other interesting titles are the books by Howard ([68], [69]), in which the focus on DP in the context of Markov models. In these early days, the computational resources were very limited. Nowadays the computational limitations are shifted but still present.

In 1999, Smith [134] presented an overview of the development of the relationship between inventory management and dynamic programming (DP). He concluded that DP has one disadvantage: its curse of dimensionality; nevertheless, DP helps developing and evaluating heuristics. He envisioned that numerical techniques can be very useful in solving complex inventory management problems, such as where the products are perishable.

Before presenting the few numerical studies on optimal ordering of perishables, we first discuss the acceptance and application of numerical procedures in a non-perishable inventory setting.

Non-perishable inventory: optimal parameter values

Although inventory management problems have acted as key examples to illustrate the dynamic programming principle, the application of dynamic programming to solve inventory management problems did not really flourish in practice during the first decades. In 1960 Sasieni [123] discussed a stochastic inventory problem of non-perishable products with fixed order costs next to variable holding costs. He thoroughly discusses the dynamic programming principle for minimizing the long-term average costs. Numerically he showed how optimal parameter values for an (s, S)-rule are computed through value iteration.

Johnson [72] presented in 1968 a policy iteration algorithm that exploits the special (s, S) structure; thus speeding up the search for optimal values of the parameters s and S. Most other methods for finding optimal parameter values are essentially based on full

enumeration over the two-dimensional grid in the $(s, s+k)$ plane, the interested reader is referred to [156], [10] and [4].

Despite all efforts in developing numerical procedures, these algorithms were hardly used to optimize the parameters. Federgruen and Zipkin presented in [46] a similar algorithm and showed that optimal parameter values could be computed quicker than generally believed.

Further improvement is attained in 1991 by Zheng and Federgruen. In [169] and [170], they present an efficient search algorithm based on an analysis of the convexity of the seemingly ill behaved cost function under an (s, S)-rule. Although the cost function may have several local optima and generally fails to be quasi convex, they pretentiously state that optimizing an (s, S) policy is almost as easy as evaluating a single policy. Although this might be the case for non-perishable products, it may not be the case for perishable products. The analysis and computations are much more difficult when products have a fixed maximal shelf life.

Perishable inventory: optimal stock-age-dependent policy

Truly optimal policies are generally hard to compute, due to the dimensionality of the state space. The state description contains the number of products in stock of each age category. Nevertheless, some efforts were made by Nahmias [99] in 1977, Parlar [110] in 1985, Nandakumar and Morton [102] in 1993, and most recently, in 2003, by Blake et al. [19] and Haijema, Van der Wal and Van Dijk ([56], [57]).

Nahmias [99] and Nandakumar and Morton [102] consider a very simplistic Dynamic Programming (DP) formulation of the perishable inventory problem. The general perishable inventory problem is simplified such that the state space for the truly optimal MDP problem is two-dimensional: the maximal shelf life is set to (at most) three periods ($m = 3$) and products are delivered instantaneously (zero lead time). Several complicating aspects as the periodicity of the demand, production stop during weekends, the distinction of two types of demand and issuing policies other than FIFO are not considered. By successive approximation an optimal production volume is computed. By simulation they compare the truly optimal strategy with an approximately optimal stock-age-dependent ordering policy of lower dimensionality and with an approximate order-up-to S rule. For a more detailed discussion of their work see the next section on simulation studies and the references [99] and [102].

In 1985, Parlar [110] applies Linear Programming (LP) to solve a stationary MDP for

setting an optimal ordering policy. He investigate the problems in a very simplistic setting with a maximal shelf life of only 2 periods to allow efficient computation of the truly optimal policy.

From 2003 onwards Blake et al. [19] and Haijema, Van der Wal and Van Dijk [56, 57, 58] show that the problem is nowadays tractable (via successive approximation) for a maximal shelf life of up to 7 periods, when the scale of the demand is not too high. When demand plays at a large scale, the problem is approximated by down-sizing for efficient computation. Blake et al. investigate the tractability and applicability of numerical techniques for solving an MDP formulation of the PPP. The PPP that they investigate under linear cost structure is solved over a finite horizon by assuming that production and demand happens in batches (of multiple pools). They show that reducing the batch size and lengthening the planning horizon results in lower costs but requires more computation time.

In Haijema, Van der Wal, and Van Dijk [56, 57, 58] the structure of numerically computed optimal strategies is investigated for several cases. For simple cases, it is shown that the optimal policy resembles a simple order-up-to S rule with fixed order-up-to levels for each day of the week. When operating at a small scale or when multiple demand groups are distinguished more-advanced replenishment rules fit better to the optimal strategy. Nevertheless they conclude that the outdating and shortage figures of simple order-up-to S rules (with levels read from the optimal policy) are nearly optimal for the realistic platelet production problem that they consider. As will be presented in this thesis the problem can be modeled at a more realistic level than before. From the solution of the dynamic program one learns new ordering rules that closely resemble the structure of the optimal policy.

Conclusions

1. Numerical solution procedures such as successive approximation and linear programming are very suitable for finding optimal stock-age-dependent policies,

2. Nevertheless these techniques are hardly used because of the so-called curse of dimensionality, which prevents the computation of optimal strategies for realistically large problems.

2.4.3 Simulation studies

In simulation studies of perishable inventory systems one mimics the system controlled by some fixed ordering and issuing policy to evaluate the performance in detail. Again blood banking applications and order-up-to S policies get much attention in the literature. A simulation approach allows to model a problem in much more detail than an analytical approach. Nevertheless one needs to carefully avoid over-specification of the simulation model, since obtaining accurate performance estimates from simulation becomes very time consuming when the model gets more and more detailed. A few optimal stock-age-dependent order-up-to S rules are investigated, but since simulation is by nature a non-optimizing procedure the truly optimal stock-age-dependent policy remains intractable.

Simulation is thus mostly used to evaluate a system to gain insights in search for improved policies. One may use a simulation program in combination with a (heuristic) search procedure to find good and hopefully optimal parameter values within a class of policies. Before reviewing the most relevant simulation studies, we briefly give a few references of simulation based search techniques.

Simulation-based optimization concerns the optimization of some objective (or cost) function over different parameter values of some control rule. Since there is in most cases no analytical, closed-form expression available for the cost function, one draws a response curve through points evaluated by simulation. When the curve is proven to be convex, or seems to be so, an efficient search algorithm can be constructed to find (nearly) optimal parameter value(s). An efficient search algorithm for finding an optimal order-up-to level of an order-up-to S rule is Fibonacci search, as described in [115], [29], [30], [97].

Another way for setting parameter values is by simple rules of thumb, such as "Set S equal to the expected demand over some period(s) eventual plus a fixed multiple of the standard deviation of the demand over some time interval".

When order-up-to levels should depend on the day of the week, as in periodic problems, the search for the best parameter values is more complicated. An exhaustive search would be very time-consuming given the number of scenarios to consider. In the next chapters we will present two approaches to deal with this difficulty.

Survey of simulation studies

We start our review with the work of Pinson, Pierskalla and Schaefer in 1972 [115] and that of Cohen and Pierskalla (in 1974-1979, [29], [30], [31]) in which they generalize the class of order-up-to S strategies. By adding weights to the stock level for each age category, they create, what we call, a *weighted* order-up-to S rule: the production volume a is set according to:

$$a = (S - \sum_{r=1}^{m} w_r x_r)^+, \quad \text{with constant weights } w_r \tag{2.1}$$

In case all w_r are equal to 1 then the rule is the order-up-to S rule. Otherwise, the ordering policy in Equation (2.1) is stock-age-dependent.

Pinson et al. and Cohen and Pierskalla assume in their studies increasing weights: 'young' products get higher weight than old products in aggregating the stock-on-hand. The production volume is thus lower when more young products are in stock. When many old products are in stock the production volume is relatively higher to anticipate outdating of old products.

In the simulation experiments of Pinson et al. [115] and of Cohen and Pierskalla ([29], [30], [31]) the focus is on the ordering of whole blood products that have a long maximal shelf life of 21 days ($m = 21$). Typical for the problem they investigate is that (issued) products may return to the inventory after a so-called cross matching. They use data obtained from several blood banks and hospitals in Illinois (USA). Under a FIFO issuing policy and a linear cost structure the impact of the weights (w_r) is studied. For different weights w_r, they draw response curves that visualize the impact of the order-up-to level S on the occurrence of shortages and outdating. From the seemingly convex curves a range of nearly optimal values S is read. Since the curve appears to be rather flat around the optimum, and the accuracy of the simulation is limited, multiple values of S seem to be optimal.

Furthermore, Cohen and Pierskalla study the impact of the issuing policy (LIFO versus FIFO). The products they investigate have a relatively long shelf life and outdating is thus a non-issue when using an order-up-to S rule in combination with FIFO issuing. Weighting the age categories does thus not make much sense.

In 1975 Nahmias [97] reasons that production volumes for a general perishable product should be set lower than in the non-perishable case. He investigates three *modified* order-up-to S rules:

1. production is a fixed fraction of the production volume set by the optimal order-up-to level of the non-perishable case (S^{np}):
   $a = \beta \cdot (S^{\text{np}} - x)^+$

2. production is derived from the order-up-to level S^{np} for the non-perishable case, but is bounded by an upper bound $\overline{a}$ to prevent excessive production:
   $a = \min\{\overline{a}, (S^{\text{np}} - x)^+\}$

3. production is set by a simple order-up-to S rule with order-up-to level S smaller than the optimal ones for the non-perishable case ($S < S^{\text{np}}$):
   $a = (S - x)^+$.

None of these rules is stock-age-dependent. Nahmias is primarily interested in the comparison of the performance of these rules. Therefore he considers a clean simple problem setting, leaving out periodicity of demand, production stops and any other problem specific complicating aspects. It is assumed that any shortages are backlogged. By simulation and Fibonacci search techniques he evaluates and optimizes the rules under FIFO issuing, for varying choices of the variable ordering, outdating, and shortage costs and for different demand distributions (exponential, Erlang-2 and Erlang-5). From the execution of simulations he concludes that all rules perform almost equally well since outdating is not a big issue under these rules. A simple order-up-to S rule is preferred for its simplicity in practice.

In 1977 Nahmias [99] is the first to report numerical results for the truly-optimal policy as obtained through successive approximation. Since the computation of a truly optimal policy is seriously hampered by the dimensionality of the state space, he suggest an approximate policy based on a reduction of the dimensionality by leaving out the oldest age categories. He argues that such an approximation is justified under FIFO issuing, since the number of old products is normally relatively small compared to the number of young products.

To allow 'efficient' computation the problem is simplified such that the state space for the truly optimal MDP problem is two-dimensional: the maximal shelf life is set to three periods ($m = 3$) and he assumes zero lead time. Thus the number of products with residual shelf life three days, x_3, equals today's production volume. The number of products left

behind from yesterdays replenishment is x_2. The x_1 oldest products do perish at the end of the day if not taken from stock. In an approximate MDP model Nahmias leaves out the oldest products as if no products will be in stock for more than 2 days. The optimal order quantity in this approximate model thus solely depends on stock level x_2. The approximate strategy is compared to the optimal stock-age-dependent MDP policy, which depends on both x_1 and x_2. Nahmias shows by simulation that the approximate MDP policy performs nearly optimal under the given linear cost structure (which includes variable holding, ordering, shortage and outdating costs). Note that in the (late) seventies solving the two-dimensional MDP was already very time-consuming. The approximation strategy was obtained in a few minutes, whereas computing the optimal MDP policy took more than an hour.

Nahmias thus derives numerical bounds on outdating and shortages and thus provides a benchmark to simple rules. Nahmias shows that the approximate MDP policy is better than the approximate order-up-to S rule formalized in Equation (2.2):

$$a = (S - \sum_{r=i}^{m} x_r)^+. \tag{2.2}$$

The approximate order-up-to S rule is a simple stock-age-dependent rule based on the available 'young' stock with a residual shelf life of at least i periods (with $1 \leq i < m$). Note that for $m = 3$ and $i = 2$ this rule can be seen as the weighted order-up-to S rule with $w_1 = 0$ and $w_2 = w_3 = 1$ of Equation (2.1). In 1993, Nandakumar and Morton [102] revisit Nahmias' work for testing some new approximations based on upper and lower bounds on the outdating and shortages figures using the 'Newsboy theory' and other results for non-perishable inventory.

When production does not happen daily or when demand strongly depends on the day of the week, an order-up-to level with (different) order-up-to levels for each day of the week is more appropriate. This clearly complicates both the analysis as well as the search for nearly optimal parameter values. Therefore one is interested in a simple rule for setting good parameter values. In 1983, Katz et al. [75] study the $(\mu + T \cdot \sigma)$-rule for setting order-up-to levels. In the formula, μ is the mean demand for the particular production day and σ is the standard deviation of the demand.

In 1991, Sirelson and Brodheim [132] report in more detail on a case study, in which they compare by simulation several ways to set order-up-to levels S for the order-up-to S rule. One approach is to set $S = (\mu + 3 \cdot \sigma)$, where μ is the average demand until the next order

point, and σ is the standard deviation. This approach results in the $S = (\mu + 3 \cdot \sigma)$-rule.

For practical use, they suggest order-up-to levels that do not depend on the second moment of the demand (σ), but solely on μ. Therefore they propose to let S be $k \cdot \mu$, for any real value k. For $\mu = 1$ they report on, what-they-call, normalized stock levels S_μ. The value of S_μ follows from simulation experiments and typically falls in the range of 1.5 to 2.5 depending on the desired balance between shortages and outdating. Through quadratic regression Sirelson and Brodheim estimate outdating and shortages curves in terms of S_μ.

For other $\mu \neq 1$ order up-to levels S are derived from the normalized stock levels S_μ according to

$$S = \mu \cdot S_\mu. \tag{2.3}$$

In their study, Sirelson and Brodheim do distinct A, B and O platelet inventories at 14 hospitals of the Greater New York Blood Program. They do assume that products of different blood groups are no substitutes to each other.

In 1993, Goh, Greenberg and Matsuo [50] present a simulation study, in which two demand categories are distinguished. The demand of one category strongly prefers 'young' products, while the other demand category can be met by somewhat older items. Both demands are satisfied by issuing first the oldest pools in stock that satisfy the respective age restriction. The 'young' pools may or may not be available to meet the demand for old pools. In their simulation experiments the demand for 'young' is 10% up to 50% of the total demand. Analytically obtained upper and lower bounds on shortages and outdating are validated by simulation. The simulation study concerns perishables with a maximal shelf life of 42 days (young products are products up to 10 days old), and the daily demand is Poisson distributed. The study the problem at three scales: the average total demand is respectively 1, 5 or 10 products per day. Outdating appears to be in most cases still 4-6% (and even higher) as some products are return after being issued for a number of days to cross match the RBC product with the blood of the patient. At most a few percent of the demand is lost due to shortages.

In 1997, Hesse, Coullard, Daskin and Hurter [64] report on a case study for a blood bank that serves 35 hospitals in the Chicago area. Over 1995-1996, the overall outdating of platelet concentrates was more than 20%. Hesse et al. classify the hospitals in groups and set 'optimal' periodic review (s, S)-policies for each group. Although they do optimize they do not report on the best parameter values for s and S. In their study they assume

a shelf life of 5 days and that the products are issued in a FIFO order. By deterministic simulation using a data set over 1995 – 1996, they show that outdating at individual hospitals can be reduced from more than 20% to 3.5% – 12%.

Recently, a simulation study of the UK blood supply chain is executed by Katsaliaki and Brailsford [74]. Their study covers many different blood products. The simulation model includes the transhipment of blood products from storage centers with excessive stock to centers with deficient stock, as well as emergency and ad-hoc deliveries. Further they acknowledge the limited compatibility of blood from different blood groups. Although they report in [74] significant savings by improved inventory management of RBCs, they do not report on platelet concentrates.

Conclusions

From the above simulation studies we learn that

1. issuing the oldest available products first greatly reduces outdating, but is inappropriate when demand is age-specific,
2. outdating and shortages happen frequently when a great part of the demand requires 'young' products,
3. great savings on outdating, shortages, and operating costs seem to be possible by applying simple order-up-to S rules.

2.4.4 Experimental and Empirical studies

Experimental and empirical studies report on current practice and on testing or implementing new policies in practice. The results reported are thus utmost realistic, but the conclusions hold primarily for the case under consideration. Since the experiments are performed real time and are not replicable, sensitivity analysis is not possible. Thus it is hard to draw general conclusions other than that the outcome of the experiments does or does not support conclusions from more general numerical and simulation studies. For the case of inventory management of blood products, such as platelets pools, the experimental studies, which we discuss below, indeed show that outdating can be reduced significantly.

In 1984, Ledman and Groh [82] from the American Red Cross Blood Services showed that the outdating of platelet pools can be reduced from about 20% to 3%. Their empirical results come from an experiment over a period of 6 months in the Washington region with one regional blood center supplying 60 hospitals with about 60,000 donor platelet concentrates (= 12,000 platelet pools) per annum. At their study platelets have a shelf life of 5 days, but during the first two months of the experiment a fraction of the storage bags can keep platelets for only 3 days. Ledman and Groh did not report on a formal model; instead a team of experts cooperated to carefully set the production levels based on the expected supply of whole blood and on predictions of the demand for BPPs and other blood products.

For RBCs a similar approach is used in part of The Netherlands, as presented in 2003 by Verhoeven and Smid [159] of Sanquin blood bank (region North West, The Netherlands). Through cooperation and sharing knowledge and information concerning the demand for RBCs at hospitals, they manage to smoothen the demand faced by the regional blood bank. In close cooperation with the donor management team, the call for voluntary donors is managed effectively. In [159] and in [157], they explain the ordering policy that inventory managers at hospitals should use. The ordering policy is set by three stock level parameters: a target level S (to determine the size of an order), a threshold $s^{(r)}$ (that triggers regular orders) and a critical level $s^{(e)}$ (for emergency orders). The target level S is set to the average demand per week, $s^{(r)}$ is set to $\frac{3}{7}S$, and $s^{(e)}$ is $\frac{1}{7}S$. This way one strives for a reduction of outdating at the hospitals and great savings in the transhipment of products.

In 2006, Verhoeven reports on experimenting with Vendor Managed Inventory [158]. Sanquin Blood bank North West and three hospitals experiment in letting the blood bank manage the inventories at the hospitals using the rules settled in cooperation with the hospitals. Hospital managers are positive and they admit hospitals are not well educated

for the administration of inventories. The applicability of the approach to platelet pools, which are stored centrally at the regional blood bank because of their very short shelf life, is however not discussed neither are figures reported on how many hospitals do adopt the suggested ordering policy and what the savings in outdating and distribution cost actually are.

Mercer and Tao [92] study the allocation and distribution of inventory to regional depots for a food manufacturer. When not all depots are visited daily the transhipment costs needs to be taken into account in deciding when to visit which depots. The work on Vendor Managed Inventory (VMI) is beyond the scope of this thesis.

Hesse et al. [64] investigate the platelet inventory management in the Chicago area. In contrast to the situation in The Netherlands the 35 hospitals in the Chicago area do stock platelets. Hospitals do not pay for excessive ordering: hospitals only pay for the platelet pools that are actually transfused. This policy results too often in 'over-ordering', but reduces the involved transportation cost carried by the blood bank. As a consequence, outdating is significant: at the individual hospitals outdating is 5% – 46%. By simulation they show that periodic review (s, S) policies in combination with a FIFO issuing policy reduces outdating to 3.5% – 12%.

Van Donselaar et al. [150] argue that for a better control of perishables in supermarkets, the order-up-to levels or what they call maximum inventory levels should depend on the (maximal) shelf life of the products. Furthermore, existing ordering systems should make better use of the impact of substitution on outdating: outdating of one product can be reduced by order or display less fresh items of competing products.

A different line of research of perishable inventory control is on pricing and discounting strategies. Alternatively one might prioritize demand and refuse low-priority demand when stock is low. Since we focus on ordering policies we will not consider other policies to control outdating.

For more case studies we refer to a few studies based on simulation ([30], [89], [75], [132], [44], and [74]) or on other numerical techniques [19] using realistic data, which are already discussed in the previous sections.

Conclusions

From experimental studies we learn that

1. on average 5 to 20% of BPPs at blood banks and hospitals become non-transfusable due to outdating,
2. at small hospitals outdating can be even close to 50%,
3. outdating can be reduced significantly through careful production and inventory management.

2.5 Blood platelet pool studies

Many of the above mentioned studies on blood bank inventory management deal with RBCs. As RBCs have a long maximal shelf life of five to six weeks, the inventory management seems to be more complicated for BPPs, which have a very short shelf life of only 5 to 7 days. Moreover the demand for BPPs arises at a much smaller scale than the demand for RBCs.

Besides being the most perishable blood product, a BPP is also the most expensive blood product, due to the high processing cost. Disposing outdated pools is thus a waste of money and undesirable for ethical reasons. Nevertheless, a number of studies report high outdating figures of 10-20% or even higher at European and American blood centers (see [155], [64], [132], [82], and [75], [163]).

A major complication of the PPP is that a great part of the demand strongly prefers 'young' BPPs, of say at most 3 days old, whereas the remaining part of the demand may get older BPPs. Nevertheless in most studies this distinction is not made. Current developments (in The Netherlands) of allowing pools to be stocked for a longer time (up to seven days) make the distinction between demand for 'young' pools and the demand for pools of any age more important.

In the remainder of this section, we summarize the results from studies on the inventory management of blood platelets. In chronological order, the most relevant studies on platelet concentrates are reviewed below. Most of the citations stem from the previous sections, in which the focus was on the different methodologies and techniques in use.

Two of the first platelet studies, that we have found, are the PhD thesis of Deuermeyer [36] in 1976 and the paper of McCullough [89] in 1978. Based on a data set of realized demand figures McCullough examines several ways to improve platelet availability. Deuermeyer uses simulation techniques to investigate the relationship between the (total) stock level and the occurrence of outdating and shortages.

In 1983, Katz et al. [75] showed in a simulation study that an order-up-to S strategy may perform very well, when all demand is met by issuing the oldest BPPs in stock first. The different order-up-to levels S are considered according to $S = \mu + T \cdot \sigma$. The mean demands (μ) as well as the standard deviations (σ) for each day of the week, are estimated based on historical data for two years. Production happens 6 days a week. For given values of T they report almost no outdating and shortages. When the shelf life is only 3 days, outdating is about 2% when T is set to 2.

Ledman and Groh [82] showed empirically that by carefully choosing the production

volumes, the outdating of pools can be reduced from about 20% to 2.6% in the Washington region. Instead of using a model, a team of experts met every working day to carefully set the production volume based on the expected needs at hospitals.

In 1991, Sirelson and Brodheim [132] investigate, just as Katz et al., how the replenishment level S of an order-up-to S rule affects the occurrence of outdating and shortages. An appropriate replenishment level S can be read from curves that show the relation between the shortage rate, the outdating rate and the what they call normalized base stock levels S_μ. The curves are obtained by quadratic regression of simulation results. After choosing the normalized stock level S_μ that reflects an acceptable trade-off between shortages and outdating, the order-up-to level S is set to $\mu \cdot S_\mu$, with μ being the mean demand till the next ordering point. Since the replenishment levels are normalized their rule of thumb allows periodic order-up-to S rules and non-stationary demand.

Hesse et al. [64] investigate the platelet inventory management in the Chicago area. In contrast to the situation in The Netherlands, many of the 35 hospitals in the Chicago area keep inventory of platelets. In 1995 – 1996, about 5% – 46% of the BPPs outdate at the hospitals, since hospitals do not pay for excessive ordering. In their study they assume a shelf life of 5 days and FIFO issuing at the hospitals. By setting 'optimal' (s, S)-policies for each stock-keeping hospital outdating is reduced to 3.5% – 12%. A practical question left for future research is on the limited substitutability of platelets units of compatible blood groups.

More recently, Blake et al. [19] modeled the PPP as a finite horizon MDP. Since the number of states may be prohibitively large for realistic problems, Blake et al. round inventory levels and production volumes to what-they-call 'buckets': production and demand happens in multiples of 5 pools. Thus the number of states to consider is reduced significantly. In [19] is shown that the MDP is tractable for varying values of the downsizing factor and how the downsizing affects the cost optimal minimum as well as the computation time.

In 2006, Veihola et al.[155] report on a European study on the efficiency of blood platelet production in 10 European countries. The study includes 17 Red Cross, national and hospital blood centers. It appears that on average about 14% of the platelet units produced is discarded, primarily due to outdating. This figure actually ranges from 7 to 25% when looking at the individual institutes, which vary in annual production scale from 3,345 to 103,643 donor units (or 669 to 20,728 BPPs per year). Veihola et al. state that blood centers may benefit from a closer look at their platelet supply chain. The supply of whole blood donation is not restrictive for platelet production: about 41% of the platelets

obtained by whole blood donations is actually used for the preparation of platelet pools. Although much effort is put into extending the shelf life of platelets, great savings can be expected by careful setting the production volumes and sharing information about stock levels and expected demand.

Conclusions

From the studies on the production and inventory management of platelet concentrates we learn that

1. the inventory management of platelets is complicated because of the very short life time,
2. in most of the (theoretical) studies preferences concerning the age of the platelets are not included, and the distinction of blood groups is commonly not addressed,
3. on average 14% of the production becomes outdated according to a recent European study at 17 European institutes spread over 10 countries.

2.6 Research questions

The PPP is a high-dimensional MDP. Although the number of dimensions can be greatly reduced by aggregating the different blood groups into a single blood group, one needs to distinct between products of different ages for deriving an optimal stock-age-dependent ordering policy. No formal quantitative justification is provided in the literature for approximating the PPP by a single product model, while virtually all analytical, numerical and simulation studies concern single product models.

Despite all the research efforts, anno 2002 the existing simple inventory policies are hardly adopted by hospitals, which still make decisions based on experience [171]. In 2003, the importance of effective strategies and cooperation in managing the blood supply chain is once again stressed in the scientific journal *Transfusion* [73] by Robert Jones of the New York Blood Center. The focus was on improving the quality of the BPPs and extending their maximal shelf life, rather than on the logistical issues to improve both the quality and the efficiency. Why did the work done so far not break through? Were blood bank managers not convinced of the value of the models too simplistic? And if so, why not: were the suggested models and solutions considered too 'mathematical', too simplistic, too specific? Or were they afraid of high costs in money and time to get these models validated and implemented?

Anyway, many characteristics of the PPP, which can also be found in other problem settings, are not included in the single product models found in the literature. Of growing importance to blood bank managers and doctors, two types of demand should be distinguished: one category of patients needs 'young' platelets while other patients may receive BPPs of any age (up to the maximal shelf life). Consequently, it seems that the need for age-depending policies becomes more and more urgent, unless one shows and justifies that stock-level-dependent policies suffice. To what extent stock-level-dependent policies are nearly optimal for the PPP is an open question, especially when a great part of the demand gets the 'youngest' products in stock.

Research questions

We formulate the following research questions to be answered in the next two chapters:

1. Is it justified to approximate the PPP by a single-product model?
2. What is the structure of an optimal stock-age-dependent ordering policy?
   (a) What simple rules closely resemble the optimal stock-age-dependent ordering policy?
   (b) How well do stock-level-dependent ordering policies fit to an optimal stock-age-dependent policy for perishables?
   (c) How close to or far from optimal are such simple rules with respect to outdating and shortages?
   (d) How does the structure of the optimal strategy change when operating at a small scale and when fixed order costs apply?
3. How robust are the approach, the optimal policy and the simple rules?
   (a) What is the impact on outdating and the age distribution when a significant part, of say 30–70%, of the demand is met by issuing the youngest BPPs in stock, and the remaining part of the demand receives the oldest products in stock?
   (b) Do changes in the values of problem parameters dramatically affect the results?
   (c) How do the results change when the supply of buffy coats for the production of BPPs is uncertain and occasionally limiting?
   (d) What if production is not possible during holidays?

By combining the strength of successive approximation for computing optimal policies and the power of simulation to read a simple rule and to justify the resulting strategies, we will answer the above questions. Effective production-inventory strategies are to be derived that eventually can be used by blood bank inventory managers.

Outline of next chapters of Part I

In Chapter 3, the combined SDP-Simulation approach for the PPP is presented. In Chapter 4, first we extend the approach to include non-stationary periods (e.g. irregular production breaks), and next we discuss the order problem when operating at a much smaller scale, such as at hospitals, and fixed order costs apply.

Chapter 3

A combined Stochastic Dynamic Programming and Simulation approach

In this chapter we present an MDP-based approach for solving the PPP presented in Section 2.1, which may be exemplar for a number of other inventory problems of perishables that have a fixed shelf life. First, the approach is sketched in Section 3.1. Next, the five steps of the approach are presented in Sections 3.2 – 3.6. The MDP formulation is provided and the different steps of the approach are illustrated using data provided by one of the four Dutch blood banks. In Section 3.7, the robustness of the approach and resulting policies is shown by an extensive simulation study.

Besides presenting the methodology, this chapter reports on a case study that may be of interest to both Operations Researchers and Blood bank managers. Although the focus is on the PPP at blood banks, the approach of finding nearly optimal ordering policies may also apply to hospitals and other production-inventory problems.

3.1 Approach

The approach to solve the PPP combines Stochastic Dynamic Programming for solving an MDP and Simulation for obtaining simple nearly optimal ordering strategies for the PPP. The approach can be divided into five steps that are discussed in the next five sections:

Step 1. Provide an MDP formulation of the PPP for a single product,

Step 2. For given data, solve the MDP by SDP, eventually after scaling the data to downsize the MDP,

Step 3. Derive simple ordering rules from an optimal MDP policy by simulation,

Step 4. Check whether the simple rules perform nearly optimal for the (scaled) problem,

Step 5. Verify by a detailed simulation study whether the simple ordering rules perform well and are robust in a more realistic setting, which could not be modeled accurately in the MDP model. If the rule was obtained after scaling the MDP, the parameters of the rule need to be first re-scaled such that they relate to the original problem size.

Aggregation-Disaggregation

In Steps 1–4, aggregation may appear in three different ways:

- The different BPPs are to a high degree substitutes to each other and are thus aggregated into a single, say universal, product,
- Next, if necessary, the BPPs are aggregated into batches such that the scaled MDP is tractable,
- Finally, to develop, a simple rule the stock of different age categories is aggregated.

In Step 5, the production volumes are disaggregated and the aggregation of blood groups into a single, universal group is justified.

3.2 Step 1 – MDP formulation

A first step in our approach is to formulate the PPP as an MDP that can be solved numerically. The state space must be small enough, to allow the computation to be done in a reasonable time window. Therefore blood groups are not included in the model, such that the dimensionality of the problem does not explode. Blood groups are aggregated as if all BPPs are of a universal group. This approach is valid since BPPs of the different blood groups are to a high degree substitutes according to the compatibility scheme in Figure 2.1. The justification of this aggregation is presented in Section 3.6.

The MDP is formulated in discrete time and for the PPP the natural length of a time slot is one day. Further, the MDP is characterized by the state of the production-inventory system, the decisions to take, the transitions from one state to a next state, and the related direct costs structure.

3.2.1 States

In the first place, we are interested in an optimal stock-age-dependent ordering policy, hence the ages of the pools in stock is to be registered. Also to record outdating we need to keep track of the ageing of the products. Suppose that the maximal shelf life is m days. The detailed information about the number of BPPs in stock of each age category is an m-dimensional vector, $\mathbf{x} \in \mathbb{N}^m$, with elements x_r being the number of BPPs with a residual shelf life of r days as inspected early in the morning.

Since production does happen only during weekdays, and demand is weekday dependent, the day of the week, $d \in \{1, 2, \ldots, 7\}$ for Monday to Sunday, has to be included in the state description as well. A complete state description is thus the pair $(d, \mathbf{x})$.

3.2.2 Decision and decision epoch

Every morning a decision is taken about the number of BPPs (a) to produce during the day. In the MDP model, it is assumed that during the day the production volume is not revised, and that enough buffy coats become available to meet this production level. At the end of the day, after 24 hours, just before the start of the next day, the a BPPs are released and added to stock.

Note that the issuing policy I is not part of the optimization problem but given for each demand category. In Section 3.3 a number of issuing policies I is discussed.

3.2.3 Transitions

In the transition from day d to the next day $d+1$ (where from now on $d+1=1$ for $d=7$), a sequence of events happen as depicted in Figure 3.1. After the inspection of the stock state $\mathbf{x}$ the production volume a is set early in the morning and during the day the next events take place:

1. BPPs are taken from stock to meet the young-demand for j 'young' BPPs and the any-demand for k BPPs,
2. mismatches occur when the young-demand is met by 'old' BPPs,
3. shortages might occur when not all demand can be met from stock,
4. at the end of the day all BPPs age 1 day,
5. some BPPs might become outdated,
6. the production is released and added to stock.

In our discrete time model, the events happen in the above order. We thus assume that the production volume is fixed before the demand for the coming day is known. The young-demand for j 'young' BPPs is assumed to be met before the any-demand is met. This order keeps the process simple and is assumed to be more realistic than the opposite order: the any-demand happens a bit more ad hoc during the day, when new patients are brought into the hospital, e.g. due to accidents, and when complications arise during a surgery. The order influences the number of shortages to each demand category. In fact, one may model any other chronological order of meeting the demand, such as spreading the demand evenly over the day (e.g. when $j \approx 2k$ then one may consider the issuing order: 2 young, 1 any, 2 young, 1 any, etc until all demand is met).

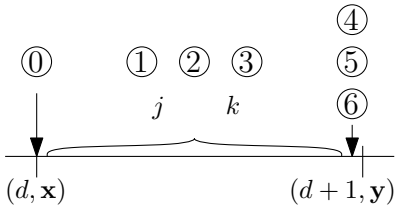


Figure 3.1: The chronological order of events in the MDP model

Each demand category has a specified issuing policy that prescribes which BPPs to select from stock to meet the demand. The composite issuing policy I is kept simple, such that the number of computations to derive a transition is kept low.

Transition function – The notation of a transition from one state $(d, \mathbf{x})$ to a next state $(d{+}1, \mathbf{y})$ is involved. Therefore we denote the next stock state $\mathbf{y}$ as a function $y(\mathbf{x}, j, k, I, a)$ to emphasizes that it depends on the initial stock $\mathbf{x}$, the composite demand (j, k), the composite issuing policy I and the production volume a. In order to track outdating, shortages and mismatches, the following notations and conventions are introduced:

- i_r^{Y} part of the young-demand that is met by BPPs with r days residual shelf life. i_r^{Y} clearly depends on $(d, \mathbf{x}, j, k, I)$ and the order in which the two types of demands occur.
- i_r^{A} part of the any-demand that is met by BPPs with r days residual shelf life. i_r^{A} clearly depends on $(d, \mathbf{x}, j, k, I)$ and the order in which the two types of demands occur,

and

- i^{Y} number of BPPs that are issued to meet the young-demand $= \sum_r i_r^{\mathrm{Y}}$,
- i^{A} number of BPPs that are issued to meet the any-demand $= \sum_r i_r^{\mathrm{A}}$.

Outdating, shortages and mismatching

With the above notation we get simple expressions for the number of outdated BPPs, for the number of shortages and for the number of BPPs mismatched:

- The number of outdated BPPs, which are removed from stock at the end of the day is $x_m - i_m^{\mathrm{Y}} - i_m^{\mathrm{A}}$,
- The total shortage is $(j - i^{\mathrm{Y}}) + (k - i^{\mathrm{A}})$ BPPs,
- When the young-demand is preferably met by BPPs that have a residual shelf life of l days or more, then the number of mismatches is $\sum_{r=1}^{l-1} i_r^{\mathrm{Y}}$.

Transition probability

The uncertainty in the PPP is solely set by the uncertainty in the demand volumes. For a chosen production volume a the transition probability $P^a_{(d,\mathbf{x}),(d+1,\mathbf{y})}$ from one state $(d, \mathbf{x})$

to a next state $(d+1, \mathbf{y})$ relates thus to the demand distributions of the young-demand and the any-demand. The probability of a demand for j 'young' BPPs and k BPPs of 'any' age on weekday d is denoted by $p_d^{\mathrm{Y}}(j)$ respectively $p_d^{\mathrm{A}}(k)$.

All combinations (j,k) resulting in the same inventory state $\mathbf{y}$, contribute $p_d^{\mathrm{Y}}(j) \cdot p_d^{\mathrm{A}}(k)$ to the transition probability $P^{a}_{(d,\mathbf{x}),(d+1,\mathbf{y})}$ as in Equation (3.1):

$$P^{a}_{(d,\mathbf{x}),(d+1,\mathbf{y})} = \prod_{(j,k)\,:\,y(\mathbf{x},j,k,I,a)=\mathbf{y}} p_d^{\mathrm{Y}}(j) \cdot p_d^{\mathrm{A}}(k). \tag{3.1}$$

3.2.4 Costs

The immediate costs include the following cost components:

- c^{H} holding costs per day per BPP in stock at the start of the day,
- c^{O} outdating costs per outdated BPP that is disposed,
- c^{S} shortage costs per BPP not in stock to meet the demand (independent of the demand category),
- c_r^{Y} (age or quality) mismatch costs per issued BPP with residual shelf life r days that violates the age-preference of the young-demand.

The immediate costs when j young BPPs and k BPPs of any age are demanded, is:

$$C(d,\mathbf{x},j,k,I) = \begin{cases} c^{\mathrm{O}} \cdot (x_m - i_m^{\mathrm{Y}} - i_m^{\mathrm{A}}) & \text{outdating costs,} \\ +c^{\mathrm{S}} \cdot (j+k-i^{\mathrm{Y}}-i^{\mathrm{A}}) & \text{shortage costs,} \\ +c^{\mathrm{H}} \cdot x & \text{holding costs,} \\ +\sum_{r=1}^{m} c_r^{\mathrm{Y}} \cdot i_r^{\mathrm{Y}} & \text{(quality) mismatch costs,} \end{cases} \tag{3.2}$$

with the total stock level $x = \sum_{r=1}^{m} x_r$. Note that $C(\cdot)$ depends on d and I via i^{Y} and i^{A}.

Given the issuing policy I, the expected immediate costs to incur in state $(d,\mathbf{x})$ are:

$$\mathbb{E}C^{I}(d,\mathbf{x}) = \sum_{j,\,k} p_d^{\mathrm{Y}}(j)\, p_d^{\mathrm{A}}(k)\; C(d,\mathbf{x},j,k,I). \tag{3.3}$$

Remark – The shortage costs may differ between the multiple demand categories. In addition, one may introduce c_r^{A} to capture any soft age-preference of the any-demand. Furthermore, $C(\cdot)$ might depend on the production volume a, when additional cost are made to realize the production of a BPPs, e.g. when the production volume requires working in overtime or when extra effort is needed to have enough donations to realize the targeted production volumes.

Fixed order costs per order can be included, hen it takes time or money to set-up a production run or when one has to pay a fixed amount for a delivery (next to proportional ordering costs). Since these do not apply to the PPP they are left out. Fixed order costs will be considered in the next chapter, in Section 4.5.

3.2.5 Successive approximation

The stochastic dynamic program for the PPP is solved by an SA algorithm, similar to the one introduced in Section 1.2.2. Instead of computing and storing all probabilities $P^{\,a}_{(d,\mathbf{x}),(d+1,\mathbf{y})}$, we save on RAM memory and only store $p_d^{\mathrm{Y}}(j)$ and $p_d^{\mathrm{A}}(k)$. The transition probabilities are computed using (3.1). SA is implemented as in Section 1.2.2, using the following recursive relation:

$\forall (d,\mathbf{x}) \in \mathcal{X}$:

$$V_n^I(d,\mathbf{x}) = \min_{a\in\mathcal{A}(d,\mathbf{x})} \left(\mathbb{E}C^I(d,\mathbf{x}) \;+\; \sum_{j,\,k} p_d^{\mathrm{Y}}(j)\; p_d^{\mathrm{A}}(k)\; V_{n-1}^I\left(d+1, y(\mathbf{x},j,k,I,a)\right) \right). \quad (3.4)$$

The SA algorithm start with $n = 1$ and $\mathbf{V}_0^I = \mathbf{0}$ and computes $\mathbf{V}_0^I$ for increasing values of n. The iterative scheme is stopped when $span(\,\mathbf{V}_n^I - \mathbf{V}_{n-7}^I\,)$ is smaller than a small value ϵ. Suppose this happens (at first) at iteration N.

Then the optimal costs level per week is approximated by $\mathbf{V}_N^I - \mathbf{V}_{N-7}^I$. The optimizing actions in the last iteration approximate the optimal policy. If optimal actions under issuing policy I are not stored, then an optimal production strategy π^I follows from Equation (3.5). The optimal production volume on weekday d in stock state $\mathbf{x}$ is:

$$\pi^I(d,\mathbf{x}) = \arg\min_{a\in\mathcal{A}(d,\mathbf{x})} \left(\mathbb{E}C^I(d,\mathbf{x}) \;+\; \sum_{j,\,k} p_d^{\mathrm{Y}}(j)\; p_d^{\mathrm{A}}(k)\; V_N^I\left(d+1, y(\mathbf{x},j,k,I,a)\right) \right). \quad (3.5)$$

The minimal long-run average costs per week related to this policy is approximated, by any element of $\mathbf{V}_N^I - \mathbf{V}_{N-7}^I$. This approximation is ϵ-accurate. The long-run average costs per day, g, follows from:

$$\left| g - \frac{V_N^I(d,\mathbf{x}) - V_{N-7}^I(d,\mathbf{x})}{7} \right| < \frac{\epsilon}{7}, \tag{3.6}$$

for an arbitrary state $(d, \mathbf{x})$.

3.2.6 Computational complexity

To do the computations in Equation (3.4) for all possible states $(d, \mathbf{x})$, in finite time the state and action space has to be finite. Both the action space and the state space are finite when we imposing an upper bound on the production volumes. This upper bound can be set to the present production capacity, or to an (artificial) finite production capacity for each day of the week. Even when the production capacity is limiting the number of states can be very large. When U is an upper bound on the (artificial) daily production capacity, the number of states is $\mathcal{O}((1+U)^m)$. Hence the number of states grows exponentially fast in the number of age categories m.

We discuss two ways for speeding-up the SA algorithm:

1. The number of states to consider can be reduced by imposing a (fictitious) storage capacity, such that states with an unrealistically-large, sub-optimal number of BPPs in stock are left out.

2. During each iteration it suffices to consider a single value of d: successive values of d may be considered in successive iterations as the PPP is periodic in d.

Imposing an (artificial) storage capacity – Assuming that it is suboptimal to have more than X BPPs in stock, the number of states to consider during each iteration can be reduced by imposing an (artificial) storage capacity of $\overline{x} < mU$ BPPs. All states with $x = \sum_{r=1}^{m} x_r > \overline{x}$ are not included in the state space $\mathcal{X}$. Consequently one should remodel transitions to a state that is outside the (truncated) state space. For example, when the initial total stock level $x = 20$ plus the production $a = 6$ minus the demand $j + k = 3$ exceeds $\overline{x} = 21$, we enforce to stay in the state space by removing the oldest $(x + a - (j + k) - \overline{x})^+ = 2$ BPPs from stock, such that one ends up in a state with $x = \overline{x}$.

We do not want this modeling trick to affect the optimal strategy too much, as the storage capacity is artificially introduced solely to reduce the state space. In a simulation model, we label such actions as *'overproduction'*, as the oldest BPPs would not be removed when the production volume a was set lower. By simulation we check whether overproduction happens very rarely, to justify the height of the chosen artificial capacity. We prefer to set $\overline{x}$ as small as possible, but large enough such that long-run average overproduction is (virtually) zero.

Iterate over one periodic class per iteration – In Equation (3.4) we iterate over all $\mathbf{x} \in \mathcal{X}$ for all days d, while it is more efficient to start with a specific d, say $d = 7$, and iterate only over all $\mathbf{x}$ in the *periodic class* $\mathcal{X}(d = 7)$. In the second iteration we choose $d = 6$ and iterate over all $\mathbf{x} \in \mathcal{X}(d = 6)$, etc. After 7 iterations d is reset to 7. This way the computational burden per iteration is reduced by a factor 7. Thus making use of the periodicity in d results in a more efficient implementation.

3.3 Step 2 – Scaling and Solving

Despite the suggested ways for speeding up an SA algorithm, the number of states to visit during a single iteration can be tremendous for the PPP with realistic data. A straight-forward computation of the optimal production policy is often not possible. In this chapter we use data gathered in 2003-2004 for one of the four Dutch blood banks, Sanquin's Division North-East. The number of BPPs to be kept in stock to meet the demanded is too large to solve the problem by SA. Therefore we suggest to scale the problem. We subsequently discuss, in this section, the data, the scaling, and the results for the scaled problem.

3.3.1 Data

The chosen parameter values are estimates based on data from, and communications with, the Logistics manager and the head Blood Processing of Sanquin's Division North-East (The Netherlands) amongst other [16, 66, 133].

Maximal shelf life

BPPs have a shelf life of at most 5 days, but in the near future the shelf life will be raised to 7 days, depending on improved processing techniques, clinical studies and legislation. The state space, including the weekday, is thus currently 6-dimensional, and is expected to be 8-dimensional in the near future.

Demands

The demand distributions are approximated by fitting the distribution on an estimated mean and standard deviation coefficient of variation. The mean demand is estimated using historical data and expectations regarding the future demand. The standard deviation of the demand is set as if demand is Poisson distributed.

The mean demands on Monday – Sunday are roughly 26, 21, 32, 21, 26, 8, and 10 BPPs. Blood bank managers estimate that (at most) 60-70% of the demand is *young-demand*, which prefers 'young' BPPs of say at most 3 days old (counted from the release date). Young-demand may be mismatched by issuing older pools, although is less favored. In the current practice, no strong distinction is made between young-demand and any-age: both types of demand are met by issuing the oldest BPPs in stock first. Nevertheless,

Table 3.1: Mean daily demand for BPPs at a Dutch blood bank.

Demand	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Weekly
'Young'	20	15	26	15	20	0	0	96
'Any-age'	6	6	6	6	6	8	10	48
Total	26	21	32	21	26	8	10	144

to increase their quality of service, blood bank managers may be interested in different issuing policies for the two types of demand.

In Table 3.1 the average demand of each demand category, as well as the total demand, is reported for each day of the week. Demand is assumed to be stationary but periodic, with period 7 days: e.g. all Mondays are stochastically the same. The average weekly demand is 144 pools and the average daily demand varies over the weekdays. Demand peaks on Wednesdays and continues during weekends although regular production of BPPs stops during weekends. The demand for 'young' BPPs is during weekend virtually zero as no transfusions are scheduled during the weekends. On Wednesday the ratio between young-demand and any-demand is 26 :6, thus more than 80% of the demand prefers 'young' BPPs. On average this ratio is 96 : 48, hence 67% of the demand prefers young BPPs.

Both types of demand are modeled as if demand is approximately Poisson distributed. The coefficients of variation for the total demand range from $1/\sqrt{32} \approx 0.18$ on Wednesday to $1/\sqrt{8} \approx 0.35$ on Saturday. Ledman and Groh [82] report similar values for the coefficients of variation for an empirical case study in region Washington (DC). Apparently the assumption of Poisson distributed demand is quite realistic.

Issuing policy

In the current practice, BPPs are issued in FIFO order, i.e. oldest BPPs are issued first. Since a great part of the demand prefers 'young' BPPs, alternative issuing policies are considered. Next to the common FIFO and LIFO issuing policies discussed in Section 2.2.1, we introduce a new issuing policy that we call FIFOR(l) for FIFO Restricted to age category l. In fact this new issuing policy is a mixture of FIFO and LIFO when mismatching is allowed. Under FIFOR(3) first BPPs with residual shelf life of 3 or higher are issued in the FIFO order, when not all demand can be met this way and if mismatching is allowed older BPPs are issued in a LIFO order. FIFOR(l) is primarily developed for meeting the young-demand.

The FIFOR(3) policy is illustrated in Figure 3.2, where the young-demand for 11 BPPs is met from stock state $\mathbf{x} = (1, 4, 2, 1, 5)$. Since not enough young BPPs are in stock to meet all young-demand, mismatching occurs. Three BPPs are of age category 2, i.e. these BPPs have a residual shelf life of 2 days, consequently one incurs a penalty of $3 \times c_2^{\mathrm{Y}}$.

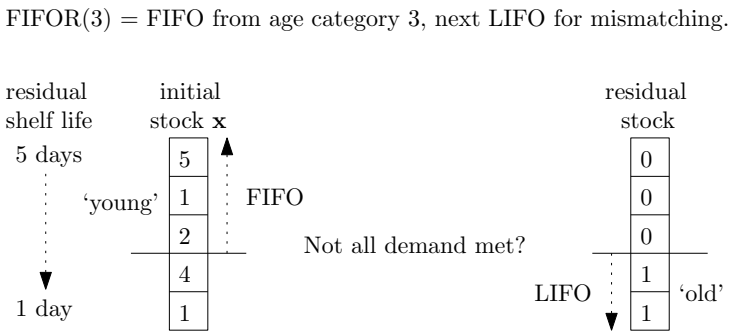


Figure 3.2: An illustration of issuing policy FIFOR(3).

Our FIFOR policy is similar to the upward and downward substitution rule as presented by Deniz, Scheller-Wolf and Karaesmen[35]. When mismatching is not allowed, FIFOR corresponds to the same issuing policy as presented in Pierskalla and Roach [114].

The issuing policy I describes how the two types of demands are met from stock. I is thus a composite issuing policy. In our notation we will first denote the issuing policy for the young-demand followed by the policy with respect to the any-demand. Thus $(\mathrm{LIFO}, \mathrm{FIFO})$ implies issuing the youngest pools to meet the young-demand and issue the oldest pools in stock to meet the any-demand. Combinations that we consider for I are:

- $(\mathrm{FIFO}, \mathrm{FIFO})$: meet all demand by the oldest BPPs in stock,
- $(\mathrm{LIFO}, \mathrm{LIFO})$: meet all demand by the youngest BPPs in stock,
- $(\mathrm{LIFO}, \mathrm{FIFO})$: LIFO for young-demand, FIFO for any-demand, and
- $(\mathrm{FIFOR(3)}, \mathrm{FIFO})$: FIFOR(3) for young-demand and age-mismatching is allowed.

Cost structure

The cost parameters may take fictitious values, since only their relative values are of importance for balancing the different criteria rather than their absolute value. The following cost figures are chosen:

- $c^{\mathrm{O}} = 150$ euro per BPP
  The major concern is to balance outdating and shortages. The cost of an outdated BPP is 150 euro, which is the variable production cost that are not paid off.

- $c^{\mathrm{S}} = 750$ per BPP short
  A shortage requires getting a BPP from another blood bank at high transhipment costs, further shortages imply a loss in goodwill. Therefore the variable shortage costs are set to a multiple of say five times the outdating costs.

- $c^{\mathrm{Y}} = (200, 200, 0, 0, 0)$
  Age or quality mismatch costs are incurred when young-demand is met by BPPs that have a residual shelf life of less than $l = 3$ days. In the current practice meeting the age-preferences is getting more-and-more important, although no preference for BPPs of a specific age category is formulated by doctors in the hospital. Instead we make a clear distinction between the two demand types by setting the mismatching cost quite high when BPPs with only 1 or 2 days residual shelf life: $c_1^{\mathrm{Y}} = c_2^{\mathrm{Y}} = 200$. The mismatch costs are higher than the outdating costs to reflect that the quality of service is more important than outdating.

- Proportional holding costs are small, say 0.1.

Capacity constraints

Although the show-up of donors, the production capacity, and the storage space is almost never limiting in The Netherlands, we need to set limits to reduce the state space. For the given demand figures we assume that, under an optimal ordering policy, the production stays well below $U = 80$ BPPs per day. On Saturday and Sunday the production capacity is set to zero. According to expression $\mathcal{O}((1+U)^m)$ obtained in Section 3.2, the number of states is then still in the order of billions when the maximal shelf life is 5 days. A reasonable bound on the storage capacity may be the average demand per week, hence $\overline{x} = 144$ BPPs. The total number of states is still very large. Therefore we suggest to scale data for the PPP.

3.3.2 Scaling

For the data of Sanquin's Division North-East, the MDP has billions of states to consider, whereas the optimal strategy is only tractable for cases with at most a few million states. A way to reduce the computational complexity is by scaling the problem parameters. Therefore we transform the data as if demand happens in batches of B pools. Consequently production volumes, and stock levels x_r are multiples of B pools. The problem can thus be solved at a more coarse or aggregated grid. The number of states reduces roughly by a factor B^m from $\mathcal{O}((1+U)^m)$ to $\mathcal{O}(1+\lceil\frac{U}{B}\rceil^m)$. In the numerical study of the PPP, we set the *'scaling factor'* B to 4.

Scaling the production and storage capacity – The production capacity at each weekday is then set to 20, and the storage capacity is 36 BPPs. Then the number of states in the MDP is 850,414. (In the scaled cases that we consider, the number of states is reduced a bit more by setting a day-dependent production capacity, and reducing the storage capacity even further.) Hence the scaled MDP can be solved in a reasonable amount of time. But first the demand distribution and the costs are to be scaled.

Scaling of demand distributions – The mean demand figures reported in Table 3.1 are divided by $B = 4$. To keep the same degree of uncertainty, the coefficients of variation (cv) are kept the same as those for the unscaled problem. In Table 3.2 the mean demand figures for the scaled MDP are reported, as well as the coefficient of variation.

So far the scaling is simple. A bit more complex is the scaling of the daily demand distributions, which should fit to the scaled means and coefficients of variations (of the original data). Fortunately, Adan, Van Eenige and Resing [2] have developed a procedure for fitting discrete distributions on the first two moments. We do not report their recipe here, but simply report on an example.

Table 3.2: Scaled mean daily demand figures (in batches of 4 pools), and coefficients of variation (cv).

Demand	Statistic	Mon	Tue	Wed	Thu	Fri	Sat	Sun
'Young'	mean	5.00	3.75	6.50	3.75	5.00	0	0
	cv	0.22	0.26	0.20	0.26	0.22	0	0
'Any'	mean	1.5	1.5	1.5	1.5	1.5	2.0	2.5
	cv	0.41	0.41	0.41	0.41	0.41	0.35	0.29

For example, consider the mean any-demand on Monday is $6/4 = 1.5$ batches. The cv is set to the original $cv = 1/\sqrt{6} \approx 0.41$. According to Adan et al., the demand distribution for the any-demand on Monday is approximated by a Binomial distribution: $\widetilde{p}_1^{\mathrm{A}} \sim \mathrm{Bin}(2, 0.75)$. The distribution of the young-demand on Mondays is a mixture of two Binomial distributions, conform Equation (3.7):

$$\widetilde{p}_1^{\mathrm{Y}} \sim 0.59 \cdot \mathrm{Bin}(6, 0.78) + 0.41 \cdot \mathrm{Bin}(7, 0.78) \tag{3.7}$$

One should be warned that for some combinations (μ, cv), no discrete demand distribution exists. We come back to this point in the discussion in Section 3.8.

All other demand distributions are derived in a similar way. To reduce the computational burden per iteration, we do truncate the probabilities at some level $\overline{j}$, such that $\sum_{j=0}^{\overline{j}} \widetilde{p}_d^{\mathrm{Y}}(j) < \theta$ for small θ. The probability distribution p_d^{Y} corresponds to $\widetilde{p}_d^{\mathrm{Y}}$, except for $p_d^{\mathrm{Y}}(\overline{j})$, which equals $\sum_{j=\overline{j}}^{\infty} \widetilde{p}_d^{\mathrm{Y}}(j)$.

Scaling of costs – Finally, the variable costs are multiplied by $B = 4$, since outdating, shortages and mismatching now occurs in batches of 4 BPPs.

3.3.3 Optimal scaled MDP solution for different issuing policies

The scaled MDP can now be solved by SA. The solution gives hopefully a good approximation of the results for the original unscaled problem. Furthermore, the structure of the scaled MDP solution may be visualized by simulation such that one may search for a simple rules that resembles the structure of the optimal policy. This will be checked in the next sections. In this section we investigate the typical results we can expect under an optimal strategy. Since the results strongly depend on the chosen issuing policy, four different issuing polices are considered.

Using the notations introduced in Section 3.2 the parameter values after scaling are as summarized in Table 3.3.

In Table 3.4 we report the annual costs and the related performance measures like the annual outdating and shortages figures and the mismatch of the demand for fresh. Outdating, shortages and mismatches are reported in absolute values (in batches of 4 pools) as well as in percentages of the annual demand. All results are obtained by multiple numerical evaluations of the Markov chain of the optimal strategy with only one costs component included in the cost structure at a time.

Table 3.3: MDP parameter values for scaled PPP (Sanquin, NE, 2003/2004).

m	=	5 (to 7)
c^{S}	=	3,000
c^{O}	=	600
c^{H}	=	0.4
c^{Y}	=	$(800, 800, 0, 0, 0)$
I	$\in$	$\{(\mathrm{FIFO}, \mathrm{FIFO}), (\mathrm{LIFO}, \mathrm{LIFO}), (\mathrm{LIFO}, \mathrm{FIFO}), (\mathrm{FIFOR}(3), \mathrm{FIFO})\}$
$p_d^{\mathrm{Y}}(j)$	:	fitted on first 2 moments that are reported in Table 3.2
$p_d^{\mathrm{A}}(k)$	:	fitted on first 2 moments that are reported in Table 3.2

In 2004, all demand was met by issuing the oldest BPPs first, $I = (\mathrm{FIFO}, \mathrm{FIFO})$, and outdating was about 10-20% against a shortage of about 1%. Mismatches were not registered. Under the optimal MDP strategy and $I = (\mathrm{FIFO}, \mathrm{FIFO})$ both outdating and shortages are less than 1% and about 6.3% of the young-demand is mismatched. The implied (fictitious) total costs are about $(14 \cdot 600 + 13.9 \cdot 3,000 + 78.9 \cdot 800 \approx)$ 115,727 euro per year.

Issuing the 'youngest' batches in stock results in less shortages and mismatches, but 156 batches outdate under the optimal MDP strategy with $I = (\mathrm{LIFO}, \mathrm{LIFO})$. The annual demand is 1872 batches, hence more than 8% of the production is not transfused because of outdating. The fictitious annual costs are about 125,304 and thus about 10,000 euro higher than under $I = (\mathrm{FIFO}, \mathrm{FIFO})$.

When different issuing policies are related to the two types of demand, the expected annual costs are much lower than under $(\mathrm{FIFO}, \mathrm{FIFO})$ or $(\mathrm{LIFO}, \mathrm{LIFO})$. The composite

Table 3.4: Mean performance figures for the MDP-optimal ordering policy under different issuing policies I. (Numerical evaluation by value iteration.)

Issuing policy	Outdating [a]		Shortage [a]		Quality mismatch [b]		Annual costs
$(\mathrm{FIFO}, \mathrm{FIFO})$	14	0.76%	13.9	0.75%	78.9	6.30%	115,727
$(\mathrm{LIFO}, \mathrm{LIFO})$	156	8.32%	8.4	0.45%	8.18	0.66%	125,304
$(\mathrm{LIFO}, \mathrm{FIFO})$	37	1.95%	4.9	0.26%	0.09	0.01%	36,605
$(\mathrm{FIFOR}(3), \mathrm{FIFO})$	35	1.89%	4.8	0.26%	0.005	0.00%	35,759

[a] in batches of 4 pools per year; % of total 1872 batches (7488 pools),
[b] in batches of 4 pools per year; % of young-demand of 1248 batches (4992 pools).

issuing policies (LIFO, FIFO) and the more complicated (FIFOR(3), FIFO) both result in very low outdating, shortage and mismatching figures. The cost optimal production policy related to a (LIFO, FIFO) issuing policy results in only 2% outdating, not even 0.3% shortages and virtually no mismatches.

The savings on the annual (fictitious) costs of using (LIFO, FIFO) instead of (FIFO, FIFO) is about 80,000 euro. The optimal production policy under (FIFOR(3), FIFO) performs almost equally well.

Preliminary conclusions – Based on the results for the scaled MDP, we may draw some preliminary conclusions:

- The annual outdating costs are under the MDP and $I = (\text{LIFO}, \text{FIFO})$ about 21,900 euro, whereas in the current practice these costs are $15\% \cdot 7488 \cdot 150 = 168,500$ euro per year. (This cost difference is even greater for $I = (\text{FIFO}, \text{FIFO})$).
- When shortages are 1% in the current practice, then the difference in the total annual shortages and outdating costs are about 187,400 euro: 224,000 ($= 15\% \cdot 7488 \cdot 150 + 1\% \cdot 7488 \cdot 750$) in the current practice, against 36,600 euro under the optimal MDP policy for $I = (\text{LIFO}, \text{FIFO})$.
- The cost difference is slightly bigger for $I = (\text{FIFOR}(3), \text{FIFO})$.

These conclusions are preliminary and require a more solid base

Average age of issued batches

Another relevant performance criterion is the age of the BPPs when they are taken from stock to meet the demand. Through simulation of the optimal policy for 100,000 weeks, the age distribution of the batches is derived. The age is registered for batches that are taken from stock to meet the demand and it is counted from the day of donation. Since the production lead time is one day, the age of a batch is always 1 or higher. Table 3.5 reports the average age of the batches upon meeting the demand. It appears that when all demand is met by FIFO, the mean age under the optimal MDP policy is 2.3 days, while this is only 1.6 days for $I = (\text{LIFO}, \text{LIFO})$.

$I = (\text{LIFO}, \text{LIFO})$ yields the lowest mean age of the batches, but, as we have seen it results in many outdated batches. Mixing an FIFO and LIFO policy gives an average age of 2 days; LIFO for young-demand and FIFO for any-demand seems to be a good

Table 3.5: Mean age of the platelets upon meeting the demand. Results are obtained through simulation of the optimal MDP policy for 100,000 weeks. Age is in days counted from day of donation: platelets are at least 1 day old.

Issuing policy	Mean age Young-demand	Mean age Any-demand	Mean age Total demand
(FIFO, FIFO)	2.3	2.4	2.3
(LIFO, LIFO)	1.5	1.8	1.6
(LIFO, FIFO)	1.4	3.2	2.0
(FIFOR(3), FIFO)	1.8	2.5	2.0

comprise between a low age and low outdating figures. $I = (\text{FIFOR}(3), \text{FIFO})$ yields the same average age of the batches, but the young-demand gets the youngest batches under $I = (\text{LIFO}, \text{FIFO})$.

3.3.4 Conclusions

By aggregating the demand as if it happens in batches of $B = 4$ pools the optimal stock-age-dependent ordering policy for the scaled PPP can be derived. Using realistic data for one of the four Dutch blood banks the optimal MDP policy is computed and evaluated numerically by evaluation of the average costs in Markov chains and by simulation. We may conclude that:

1. a great reduction of outdating from 15-20% in practice to less than 2% seems to be possible, even when two thirds of the demand gets the youngest BPPs in stock.
2. (LIFO, FIFO) is favored for yielding very low figures for outdating, shortages, and mismatching costs, as well as for providing the young-demand with young BPPs.
3. shortages are low under the optimal MDP policy with $I = (\text{LIFO}, \text{FIFO})$: on average about 0.3% of the demand requires the delivery of BPPs from another blood bank.

The economic cost saving by implementing an optimal MDP policy seems to be big: a rough estimate for the blood bank under consideration indicates that 147,000 euro per year can be saved by the reduction of outdating alone. However, the optimal policy is tractable only after scaling, and the optimal policy may be too complicated for practical use. Furthermore, a number of sensitivities, e.g. regarding the different blood groups, need to be investigated, which we will present in the Sections 3.6 and 3.7.

3.4 Step 3 – Search for simple rules in MDP policy

In the third step of our approach, the structure of the optimal stock-age-dependent ordering policy, as computed for the scaled PPP, is investigated. This may help us in re-scaling (nearly) optimal production volumes fro practical use. We limit the study to ordering policies under the (LIFO, FIFO) issuing policy. Although the structure depends on the problem data, we keep focussing on the case of Sanquin's Division North-East. Sensitivities regarding the data are investigated in the next sections and in Chapter 4.

In principle an optimal strategy can be very complicated, since optimal production volumes depend on the age of the BPPs in stock. If a simple rule exist that performs nearly optimal, a simple rule is favored for practical use. After a discussion of the optimal stock-age-dependent MDP strategy in Section 3.4.1, we investigate whether stock-*level*-dependent policies can be derived from it. Two stock-*age*-dependent policies are considered in Section 3.4.3.

An open question that we hope to answer is whether a single number gives enough information for setting nearly optimal production decisions for the complicated PPP. This may not be the case as a great part of the demand receives the 'youngest' products in stock and the maximal shelf life is only 5 days.

3.4.1 An optimal stock-age-dependent production policy (MDP)

The MDP solution for the scaled PPP was derived in the previous step of the approach. In Table 3.6 we report the optimal production volumes for a selection of states on a Wednesday morning. Although in all selected states the total inventory level x equals 14 batches, the production volume ranges from 3 to 7 batches. Apparently an order-up-to rule with a fixed order-up-to level does not apply, nor is the production volume fixed. Since the states in the table are only a selection of the so many states at which the system may be on Wednesday morning, we should investigate how likely each state occurs. Therefore we may look at the equilibrium distribution of states under the optimal policy.

Table 3.6: Optimal production volumes on Wednesday under $I = (\text{LIFO}, \text{FIFO})$ for a selection of stock states with in total $x = 14$ batches in stock. ($x_2 = x_3 = 0$ due to the production stop in the weekends).

Stock state $\mathbf{x}$					Opt. prod.	Order-up-to
x_1	x_2	x_3	x_4	x_5	volume a	level $S_3 = x + a$
0	0	0	3	11	4	18
1	0	0	5	8	3	17
2	0	0	4	8	4	18
2	0	0	5	7	3	17
3	0	0	1	10	5	19
3	0	0	5	6	5	19
4	0	0	1	9	6	20
4	0	0	4	6	6	20
5	0	0	1	8	7	21
6	0	0	2	6	7	21

Equilibrium distribution and State frequencies

Obtaining the equilibrium distribution $\mathbf{q}^\pi$ of the states under strategy π, can be done iteratively by value iteration. For increasing n, Equation (3.8) is computed starting with $n = 1$ and with initial distribution $q_0^\pi(d, \mathbf{x}) = I(\mathbf{x} = \mathbf{0})$ for all states $(d, \mathbf{x})$. The equilibrium distribution $\mathbf{q}^\pi$ is approximated by $\mathbf{q}_N^\pi$, for N sufficiently large, such that $span(\,\mathbf{q}_N^\pi - \mathbf{q}_{N-1}^\pi\,) \;<\; \epsilon$. The value of ϵ must be small compared to the probability $q_N^\pi(d, \mathbf{x})$ for the great majority of states $(d, \mathbf{x})$, say $\epsilon = 10^{-15}$.

$$\forall (d, \mathbf{x}) \in \mathcal{X} : \qquad q_n^\pi(d, \mathbf{x}) = \sum_{\mathbf{y}} q_{n-1}^\pi(d, \mathbf{x})\, P^{\,\pi}_{(d,\mathbf{x}),(d+1,\mathbf{y})}. \tag{3.8}$$

For details about the convergence, see the paper of Van der Wal and Schweitzer [148].

For studying the structure of the optimal strategy, we do not need an accurate equilibrium distribution as many states occur with a very small probability. Therefore, we suggest to simulate the optimal policy and consider only those states that occurred during the simulation. For each state that occurs we store its frequency, i.e. the number of visits to that state. After a long simulation run the relative frequencies are computed, which approximate the equilibrium distribution. In the next sections we discuss how the thus obtained simulation-based frequency tables help in approximating the optimal ordering strategy for the PPP under consideration.

3.4.2 Stock-level-dependent ordering rules

To investigate whether a simple stock-level-dependent rule fits closely to the optimal MDP policy, we visualize the structure of the optimal strategy through $(state, action)$-frequency tables in Table 3.7. The stock is aggregated into a single number $x = \sum_{r=1}^{m} x_r$. In the top row of the tables one reads the values that x has taken during the simulation; e.g. in Table 3.7(a) the total stock x ranges from 4 to 17 batches on the 100,000 simulated Mondays. The bottom row of each table shows how often each value of x has been observed during the simulation. Most-frequent (24,465 times out of 100,000) a stock state of 9 batches in total is observed on Monday morning.

The optimal actions $\pi(d, \mathbf{x})$ are translated into order-up-to levels S_d, which are simply the sum of x and $\pi(d, \mathbf{x})$. In Table 3.7(a) the first column shows that the S_d on the simulated Mondays ranges from 11 to 22 batches. When $\pi(d, \mathbf{x}) = 0$ the order-up-to level S_d is set to x to emphasize zero production, although any level $S_d \leq x$ relate to zero production. The last column shows how frequent each order-up-to level applies. The most-frequent order-up-to level on Mondays is 16 batches, which fits in 54.5% of the states visited.

When all (state,action)-frequencies on some day d are on a single row, the optimal ordering policy for that day is an order-up-to S strategy with fixed periodic order-up-to levels S_d. When all (state,action)-frequencies are on an upward diagonal, then a fixed order-quantity applies.

In Figure 3.3 we illustrate how three simple ordering rules for Mondays are read from the tables: the simple order-up-to S rule, a bounded order-up-to S rule, and the fixed order quantity rule. For each of the three rules we explain how nearly optimal parameter values are read and how good the rules resemble the optimal strategy. Therefore we define the *goodness-of-fit* of a rule as the (approximate) relative frequency of states in which the rule prescribes exactly the same order quantity as the optimal strategy.

Order-up-to S rule

For the order-up-to S rule one reads (nearly) optimal order-up-to levels S_d from the frequency tables in Table 3.7. Clearly, the order-up-to levels depend on the day of the week. When S_d is the most-frequent order-up-to level read from the table for weekday d then a (nearly) optimal production decision follows from Equation (3.9).

$$a = (S_d - x)^+. \tag{3.9}$$

Table 3.7: Simulation frequency tables of scaled MDP strategy under $I = (\text{LIFO}, \text{FIFO})$.

(a) (State, action)-frequency tables for 100,000 simulated Mondays.

Stock x	4	5	6	7	8	9	10	11	12	13	14	15	16	17	Freq(S_1)
Up-to S_1															
22														1	1
21													10		10
20											2	43			45
19									5	12	314				331
18								26	61	1366					1453
17							76	213	4383						4672
16						24465	19514	10558							54537
15					21002										21002
14				12498											12498
13			4551												4551
12		824													824
11	76														76
10															0
:															:
0															0
Freq(x)	76	824	4551	12498	21002	24465	19590	10797	4449	1378	316	43	10	1	100000

(b) (State, action)-frequency tables for 100,000 simulated Tuesdays.

Stock x	0 . . . 5	6	7	8	9	10	11	12	13	14	15	16	17	18	Freq(S_2)
Up-to S_2															
26														1	1
25													2		2
24												9			9
23										1	28				29
22								2	2	170	1				175
21						3	6	15	563	21	5				613
20					9	31	75	1479	209	35					1838
19				31	127	213	2968	913	326						4578
18			17963	20849	25014	19853	7626	1450							92755
17															0
:															:
0															0
Freq(x)	0	0	17963	20880	25150	20100	10675	3859	1100	227	34	9	2	1	100000

(c) (State, action)-frequency tables for 100,000 simulated Wednesdays.

Stock x	0 . . . 8	9	10	11	12	13	14	15	16	17	18	19	20	21	Freq(S_3)
Up-to S_3															
26														1	1
25											4	2			6
24										28	9	1	1		39
23									91	62	21	4			178
22								458	224	79	12	1			774
21							1301	753	243	34	1				2332
20						3129	1880	697	84	17	6				5813
19					4745	3740	2284	679	140	13					11601
18				12888	25368	23761	11559	790	112	20					74498
17							695	3199	598						4492
16								85							85
15															0
:															:
1															0
0									98[a]	81[a]	2[a]				181 [a]
Freq(x)	0	0	0	12888	30113	30630	17719	6661	1590	334	55	8	1	1	100000

(d) (State, action)-frequency tables for 100,000 simulated Thursdays

Stock x	0 . . . 3	4	5	6	7	8	9	10	11	12	13	14	15	16	Freq(S_4)
Up-to S_4															
17								399	46						445
16					4126	15584	26550	25562	1256	108	24	3			73213
15				155				314	15703	7305	2269	517			26263
14			3												3
13															0
:															:
0													69[a]	7[a]	76[a]
Freq(x)	0	0	3	155	4126	15584	26550	26275	17005	7413	2293	520	69	7	100000

(e) (State, action)-frequency tables for 100,000 simulated Fridays.

Stock x	0 . . . 3	4	5	6	7	8	9	10	11	12	13	14	15	16	Freq(S_5)
Up-to S_5															
23											4	2	2	1	9
22										56	46	34	8	4	148
21								44	345	831	634	944	88		2886
20						3591	17946	30836	27269	13380	3935				96957
19															0
:															:
0															0
Freq(x)	0	0	0	0	0	3591	17946	30880	27614	14267	4619	980	98	5	100000

[a] $S_d = 0$ to emphasize zero production

Stock x	4	5	6	7	8	9	10	11	12	13	14	15	16	17	Freq(S_d)
Up-to S_1															
22														1	1
21													10		10
20											2	43			45
19									5	12	314				331
18								26	61	1366					1453
17							76	213	4383						4672
16						24465	19514	10558							54537
15					21002										21002
14				12498											12498
13			4551												4551
12		824													824
11	76														76
10															0
:															:
0															0
Freq(x)	76	824	4551	12498	21002	24465	19590	10797	4449	1378	316	43	10	1	100000

(a) Reading an order-up-to S rule with a fixed order-up-to level.

Stock x	4	5	6	7	8	9	10	11	12	13	14	15	16	17	Freq(S_d)
Up-to S_1															
22														1	1
21													10		10
20											2	43			45
19									5	12	314				331
18								26	61	1366					1453
17							76	213	4383						4672
16						24465	19514	10558							54537
15					21002										21002
14				12498											12498
13			4551												4551
12		824													824
11	76														76
10															0
:															:
0															0
Freq(x)	76	824	4551	12498	21002	24465	19590	10797	4449	1378	316	43	10	1	100000

(b) Reading a bounded order-up-to S rule with bounded order quantities.

Stock x	4	5	6	7	8	9	10	11	12	13	14	15	16	17	Freq(S_d)
Up-to S_1															
22														1	1
21													10		10
20											2	43			45
19									5	12	314				331
18								26	61	1366					1453
17							76	213	4383						4672
16						24465	19514	10558							54537
15					21002										21002
14				12498											12498
13			4551												4551
12		824													824
11	76														76
10															0
:															:
0															0
Freq(x)	76	824	4551	12498	21002	24465	19590	10797	4449	1378	316	43	10	1	100000

(c) Reading a fixed order quantity.

Figure 3.3: Reading simple rules from $(state, action)$-frequency tables of optimal production policy (for Monday) under (LIFO, FIFO) issuing.

From the frequency tables we read most-frequent order-up-to levels $(S_1, S_2, \ldots, S_5) = (16, 18, 18, 16, 20)$. The goodness-of-fit on Tuesday to Friday is high: 92.7%, 74.5%, 73.2%, and 97% respectively. But as illustrated in Figure 3.3(a), the best order-up-to level on Monday fits to only 54.5% of the states visited during the simulation. On Tuesday and Wednesdays an order-up-to S rule with a fixed order-up-to level of 18 batches resembles the optimal production decisions for the most-frequently visited states, but still in about a quarter of the states visited, a higher production volume is optimal. Maybe a more refined order-up-to S rule approximates better the optimal ordering strategy.

Bounded order-up-to rule

Especially on Mondays one observes an upward diagonal that bends at order-up-to level 16, as highlighted in Figure 3.3(b). The fixed order-up-to level 16 applies only when the stock level x is between 9 and 11 batches. When the stock level is 9 or lower a fixed production quantity of 7 batches is optimal. When the stock level is 11 or higher, most often 5 batches are ordered. This suggests that the optimal production volumes on Monday can be approximated by what we call the *'bounded order-up-to S rule'* with a fixed order-up-to level $S_1 = 16$, minimum order quantity $\underline{a}_1 = 5$, and maximum quantity $\overline{a}_1 = 7$ batches.

The bounded order-up-to S rule has thus three parameters for each day of the week:

- S_d the fixed order-up-to level,
- $\underline{a}_d$ the minimum production volume,
- $\overline{a}_d$ the maximum production volume.

For given d and x the production volume a is computed as follows:

$$a = \begin{cases} \underline{a}_d & \text{, if } \quad S_d - x < \underline{a}_d, \\ S_d - x & \text{, if } \quad \underline{a}_d \leq S_d - x \leq \overline{a}_d, \\ \overline{a}_d & \text{, if } \quad \overline{a}_d < S_d - x, \end{cases} \tag{3.10}$$

Or in short: $a = \max\{\, \underline{a}_d \,,\, \min(S_d - x, \overline{a}_d)\}$.

On Mondays, the bounded order-up-to S rule with the given parameter values matches to the optimal production decisions in $(93,488 + 4,383 + 1,366 + 314 + 43 + 10 + 1 =)$ $99,605$ states out of the 100,000. The goodness-of-fit on Mondays is thus 99.6%. For the other weekdays we can read the best upper and lower bounds on the production volume, such that they maximize the fit to the optimal strategy. When the lower bound equals the upper bound a fixed order quantity rule applies.

Fixed-order-quantity rule

A very simple rule one can think of is to produce a fixed quantity every day, which may be week-day dependent. Q_d batches are to be ordered on weekday d, irrespectively of the stock level and the ages of the batches in stock. Especially on Mondays, given the diagonal shape of the elements in the frequency table, the fixed order quantity rule is of interest. In Figure 3.3(c), we illustrate that a fixed order quantity rule is read from the frequency tables by summing the frequencies along the upward diagonals. Each upward diagonal corresponds to a fixed production volume. The sum of the frequencies on the diagonal is the goodness-of-fit.

In the figure only three diagonals with non-zero figures can be drawn. A production volume of 5, 6 or 7 fits to respectively 16.7%, 19.8% and 63.5% of the states visited on the Mondays. Apparently, a fixed production volume of 7 batches is a reasonable approximation of the optimal strategy for Mondays. On the other days of the week, fixed production volumes are less appropriate: the best order quantities on Tuesdays – Fridays are optimal in only 25.3%, 32%, 26.9% respectively 31.2% of the states visited.

3.4.3 (Sub-optimal) Stock-age-dependent ordering policies

From the previous discussion we learn that the optimal ordering policy is stock-age-dependent. Although a bounded order-up-to S rule appears to fit closely to the optimal solution, we consider two stock-age-dependent ordering strategies. Especially the structure of the policy on Tuesdays and Wednesdays appears to be complicated. When many products are in stock, the age of the products appears to play an important role. We therefore propose two stock-age-dependent rules, for which stock is aggregated differently:

1. The *2D-order-up-to rule*, which has two order-up-to levels to acknowledge that a great part of the demand (strongly) prefers 'young' BPPs,

2. The *Final-stock rule* that aggregates the final stock minus any outdated BPPs, when average demand would happen over the day.

2D-order-up-to rule

The what-we-call 2D-order-up-to rule has two order-up-to levels:

- S_d an order-up-to level for the total stock level, and

- S_d^{Y} an order-up-to level for the 'young' stock.

The motivation for two order-up-to levels is that the rule should aim at having enough young BPPs in stock to meet the young-demand, next to the having enough BPPs in stock to meet the total demand. Therefore we propose a 2-Dimensional order-up-to S rule, which explains the name *2D-order-up-to rule* or in short *2D-rule*, by which actions follow from two (fixed) order-up-to levels according to:

$$a = \max\{(S_d - x)^+, S_d^{\mathrm{Y}} - x^{\mathrm{Y}}\}. \tag{3.11}$$

where x^{Y} is the total number of 'young' products in stock. When 'young' means having a residual shelf life of at least 3 days, then $x^{\mathrm{Y}} = \sum_{r=3}^{m} x_r$.

Search for good values (S_d, S_d^{Y}) – The fixed order-up-to levels for the total demand, S_d, are read directly from Table 3.7, as before for the ordinary order-up-to S rule. To obtain good values for S_d^{Y}, the optimal order quantities in states $\mathbf{x}$ are studied.

In Table 3.8, a selection is given of the 183 stock states that have occurred on the 100,000 simulated Wednesdays. The table contains the optimal order quantity $\pi(3, \mathbf{x})$ in each state and the decision prescribed by the ordinary order-up-to S rule with order-up-to level 18 batches. In the last column we report the values of S_3^{Y} under which the 2D-rule prescribes the same order volume as the optimal order quantity. By definition of the 2D-rule the set of values for S_3^{Y} is empty when $S_3 - x$ exceeds the optimal order-quantity. When the optimal production decision corresponds to $S_3 - x$ than multiple values for S_3^{Y} fit. In Table 3.8 it seems that order-up-to level $S_3^{\mathrm{Y}} = 16$ is most frequent, but the state frequencies are not reported in the table and only a selection of states is considered.
In Table 3.9, the state frequencies on Wednesday are reported for the same set of states. For each promising value of S_3^{Y}, the state frequency is copied in the related column. For example in stock state $(3, 0, 0, 1, 8)$ the best choice for S_3^{Y} is 16 and this state occurred 1638 times in the simulation. State $(1, 0, 0, 1, 9)$ was visited 776 times during the simulation. In that state, the 2D-rule fits to the optimal strategy whenever $S_3^{\mathrm{Y}} \leq 17$, but not to $S_3^{\mathrm{Y}} = 18$. Therefore the state frequency 776 is found in all columns except in the last column.

Table 3.8: Illustration of reading 2D-rule for Wednesdays: sets of S_3^{Y} under $I = (\mathrm{LIFO}, \mathrm{FIFO})$, given $S_3 = 18$. ($x_2 = x_3 = 0$, since no production in weekends).

Stock $\mathbf{x}$								1D-rule	S_3^{Y} for which 2D-rule is optimal
x_1	x_2	x_3	x_4	x_5	x	x^{Y}	$\pi(3, \mathbf{x})$	$(18 - x)^+$	$\{S_3^{\mathrm{Y}} : \max[(S_3 - x)^+, S_3^{\mathrm{Y}} - x^{\mathrm{Y}}] = \pi(3, \mathbf{x})\}$
0	0	0	0	11	11	11	7	7	$S_3^{\mathrm{Y}} \leq 18$
0	0	0	4	11	15	15	2	3	$\varnothing$
1	0	0	1	9	11	10	7	7	≤ 17
2	0	0	6	9	17	15	0	1	$\varnothing$
3	0	0	1	8	12	9	7	6	16
3	0	0	2	8	13	10	6	5	16
4	0	0	0	8	12	8	7	6	15
4	0	0	3	7	14	10	6	4	16
5	0	0	0	8	13	8	7	5	15
5	0	0	1	8	14	9	7	4	16
5	0	0	4	8	17	12	4	1	16
10	0	0	3	8	21	11	5	0	16
...	...	...	...	...	...	...	...	...	...

Table 3.9: Reading the best S_3^{Y} from the frequency distribution under $I = (\mathrm{LIFO}, \mathrm{FIFO})$, given $S_3 = 18$ is fixed. ($x_2 = x_3 = 0$, since no production in weekends).

Stock **x**					Frequency 2D-rule yields optimal decision				
x_1	x_2	x_3	x_4	x_5	$S_3^{\mathrm{Y}} \leq 14$	$S_3^{\mathrm{Y}} = 15$	$S_3^{\mathrm{Y}} = 16$	$S_3^{\mathrm{Y}} = 17$	$S_3^{\mathrm{Y}} = 18$
0	0	0	0	11	2442	2242	2242	2242	2242
0	0	0	4	11	-	-	-	-	-
1	0	0	1	9	776	776	776	776	-
2	0	0	6	9	-	-	-	-	-
3	0	0	1	8	-	-	1638	-	-
3	0	0	2	8	-	-	1706	-	-
4	0	0	0	8	-	829	-	-	-
4	0	0	3	7	-	-	575	-	-
5	0	0	0	8	-	442	-	-	-
5	0	0	1	8	-	-	788	-	-
5	0	0	4	8	-	-	34	-	-
10	0	0	3	8	-	-	1	-	-
...	...	...	...	...	...	...	...	...	...
				Total	74500	76924	91143	57234	40264
				Goodness-of-fit	75%	77%	91%	57%	40%

By summing all frequencies in a column, one computes the goodness-of-fit is of that value of S_3^{Y}. The table shows only a selection of states, but the next-to-last row reports over all states the frequency that a specific value of S_3^{Y} fits to the optimal MDP policy. On Wednesdays the most-frequent order-up-to level S_3^{Y} equals 16 batches. The goodness-of-fit of the 2D-rule on Wednesdays is 91%.

Similarly we read best order-up-to levels for the other weekdays. The resulting 2D-rule and the goodness-of-fit is reported in Table 3.10. On Thursdays all BPPs in stock are 'young' when $m = 5$, hence S_4^{Y} can be set equal to $S_4 = 16$ or lower.

Table 3.10: The 2D-rule under $I = (\mathrm{LIFO}, \mathrm{FIFO})$ with most-frequent order-up-to levels and their goodness-of-fit as read from the frequency tables.

2D-order-up-to S rule	Mon	Tue	Wed	Thu	Fri
Goodness-of-fit	61%	99%	91%	73%	97%
Total-Order-up-to level (**S**)	16	18	18	16	20
Young-Order-up-to level ($\mathbf{S}^{\mathrm{Y}}$)	5	13	16	-[a]	12

[a] $S_4^{\mathrm{Y}} \leq 16$, as all batches in stock on Thursday are 'young'.

Final-stock rule

A more refined approach to acknowledge the ageing of the stock is by looking one-day ahead. Let $\hat{y}(d, \mathbf{x})$ denote the final stock level, i.e. the next stock level before today's production is added and when average demand is met from. To compute $\hat{y}(d, \mathbf{x})$, one thus subtracts from $\mathbf{x}$ the (rounded) average demand volumes on day d and any batches that would outdate when average demand would happen during day d. The computation of $\hat{y}(d, \mathbf{x})$ thus requires a detailed registration of the outdating and the dispatching of products from stock.

For the *final-stock rule*, the production volume in stock state $\mathbf{x}$ on weekday d are easily computed from a, what-we-call, *target-stock level* F_d for weekday d:

$$a = (F_d - \hat{y}(d, \mathbf{x}))^+. \tag{3.12}$$

Under the final stock rule we thus aim at a fixed target level F_d for the number of products in stock at the start of the next morning. Therefore we may call the Final-stock rule a *target stock rule*. Since it anticipates outdating, it likely performs better than the ordinary order-up-to level, when outdating is an issue. A drawback of the final stock rule is that aggregating stock state $\mathbf{x}$ into a single figure $\hat{y}(d, \mathbf{x})$ easily results in error when done manually.

In Table 3.11, it is illustrated that a Final-stock rule fits better to the optimal MDP policy on Wednesday, than an ordinary order-up-to S rule. The goodness-of-fit of the Final-stock rule with target level $F_3 = 9$ on Wednesday is 91% and is much higher than the good ness-of-fit of 74.5% for the ordinary order-up-to S rule with order-up-to level $S_3 = 18$.

Similarly we generate frequency tables of the other weekdays, resulting in target levels for each working day: $\mathbf{F} = (9, 12, 9, 10, 13)$. On Thursday and Friday a target stock rule does not fit better than an ordinary order-up-to S rule, since outdating does not happen on Thursdays and Friday when the maximal shelf life is 5 days. Outdating happens primarily on Wednesdays: the BPPs produced on a Friday will expire at the end of next Wednesday.

Table 3.11: (State, action)-frequency tables for 100,000 simulated Wednesdays. A final stock rule fits better than an order-up-to rule.

(a) when the present stock **x** is aggregated in x.

Stock x	0...8	9	10	11	12	13	14	15	16	17	18	19	20	21	Freq(S_3)
Up-to S_3															
26														1	1
25											4	2			6
24										28	9	1	1		39
23									91	62	21	4			178
22								458	224	79	12	1			774
21							1301	753	243	34	1				2332
20						3129	1880	697	84	17	6				5813
19					4745	3740	2284	679	140	13					11601
18				12888	25368	23761	11559	790	112	20					74498
17							695	3199	598						4492
16								85							85
15															0
:															:
1															0
0									98[a]	81[a]	2[a]				181 [a]
Freq(x)	0	0	0	12888	30113	30630	17719	6661	1590	334	55	8	1	1	100000

(b) when the final stock as if average demand occurs is aggregated into $\hat{y}$.

Stock $\hat{y}$	0	1	2	3	4	5	6	7	8	Freq(F_3)
Up-to F_3										
9			18970	32038	27087	12373	672	3		91143
8		3674				972	3327	617		8590
7							86			86
6										0
:										:
1										0
0								102[b]	79[b]	181[b]
Freq($\hat{y}$)	0	3674	18970	32038	27087	13345	4085	722	79	100000

[a] $S_3 = 0$ to emphasize zero production
[b] $F_3 = 0$ to emphasize zero production

3.4.4 Selection – simple and good fit

In Table 3.12 we give an overview of the several rules, their (nearly) optimal parameter values and the goodness-of-fit for each day of the week. On Mondays a bounded order-up-to S rule resembles best the optimal strategy, on Tuesday the 2D-rule fits best. On Wednesday the final stock rule fits equally well as the 2D rule. On Thursday and Friday all rules are equally well, except the fixed order quantity which fits very badly. Only on Mondays, the Fixed-order-quantity rule resembles better the optimal strategy than the order-up to rule. The best order-up-to S rule fits to only 54.5% of the simulated Mondays, whereas a fixed production volume of 7 batches fits for 63.5%.

All results presented in Tables 3.7 – 3.12 are under the (LIFO, FIFO) issuing policy. The goodness-of-fit will be different for different issuing policies, as the corresponding optimal MDP policy changes with the issuing policy.

Table 3.12: Goodness-of-fit of some basic production rules under $I = (\text{LIFO}, \text{FIFO})$ as read from the frequency tables.

Rule		Mon	Tue	Wed	Thu	Fri
Fixed order quantity rule						
Goodness-of-fit		64%	25%	32%	27%	31%
Fixed-order-quantity [a]	$\mathbf{Q}$	7	9	6	7	10
Order-up-to S rule						
Goodness-of-fit		55%	93%	75%	73%	97%
Fixed order-up-to level[a]	$\mathbf{S}$	16	18	18	16	20
Bounded order-up-to rule						
Goodness-of-fit		100%	93%	75%	73%	98%
Fixed level[a]	$\mathbf{S}$	16	18	18	16	20
Maximum production[a]	$\overline{\mathbf{a}}$	7	-	-	9	-
Minimum production[a]	$\underline{\mathbf{a}}$	5	6	3	0	7
2D-order-up-to S rule						
Goodness-of-fit		61%	99%	91%	73%	97%
Fixed order-up-to level (Total)[a]	$\mathbf{S}$	16	18	18	16	20
Fixed order-up-to level (Young)[a]	$\mathbf{S}^{\mathrm{Y}}$	5	13	16	-[b]	12
Target level next morning						
Goodness-of-fit		55%	94%	91%	73%	97%
Fixed target level (Total)[a]	$\mathbf{F}$	9	12	9	10	13

[a] in batches of 4 pools.
[b] $S_4^{\mathrm{Y}} \leq 16$, as all batches in stock on Thursday morning are 'young'.

An ordering rule should be practical and easy to use in order to get adopted. Therefore we will not select different types of rule for the different days of the week. The goodness-of-fit of a rule gives an indication how well a policy resembles the optimal MDP policy, but to assess the quality of a rule we are interested the detailed performance statistics.

3.5 Step 4 – Performance of rules versus MDP policy

In the previous step, we have constructed different rules and found good parameter values by investigating the structure of an optimal MDP policy. The quality of these rules is to be checked and to be compared to the optimal MDP strategy. Note that only for the scaled case an exact numerical analysis of both the simple rule and the optimal MDP policy is possible by analyzing the corresponding Markov chains.

In Table 3.13 annual figures for outdating, shortage, and quality mismatches are reported, as well as the expected annual cost under the given cost structure. Next to the annual costs, one reads in the last column how far the costs levels are from the optimal MDP cost level.

All rules, except the Fixed-order-quantity rule, perform nearly optimal with respect to outdating and shortages. The outdating is about 2% of the production and shortages occur less than 0.5%. The difference between the different types of order-up-to rules is visible in the occurrence of mismatching. Although a Fixed-order-quantity rule yields virtually no shortages and mismatches, its implementation is not recommended given the large number or batches that outdate.

Under the simple order-up-to S rule mismatching is 2.17 batches. By specifying bounds on the production volumes, the average annual mismatching is reduced from 2.17 batches to 0.59 batches per year. Under all rules less than 0.2% of young-demand receives batches older than the preferred shelf life. Mismatching happens thus rarely under the (LIFO, FIFO) issuing policy.

Table 3.13: Annual performance statistics of some basic ordering rules under $I =$ (LIFO, FIFO) (numerical results from solving Markov chains).

Rule	Outdating [a]		Shortage [a]		Mismatch [b]		Annual costs	
MDP-optimal	37	1.95%	4.9	0.26%	0.09	0.01%	36,605	–
Fixed-order-quantity	157	8.41%	1.4	0.07%	0.00	0.00%	98,459	+168.9%[c]
Order-up-to S rule	36	1.95%	5.7	0.30%	2.17	0.17%	40,236	+9.9%[c]
Bounded order-up-to S	36	1.95%	5.6	0.30%	0.59	0.05%	39,077	+6.8%[c]
2D-order-up-to rule	36	1.92%	5.2	0.28%	2.16	0.17%	38,779	+5.9%[c]
Final stock rule	36	1.91%	5.1	0.27%	1.98	0.16%	38,389	+4.9%[c]

[a] in batches of 4 pools per year; % of total (1872 batches = 7488 pools),
[b] in batches of 4 pools per year; % of young-demand (1248 batches = 4992 pools),
[c] percentage above optimal cost level (MDP).

Considering the expected annual cost reported in the last (two) column(s) of Table 3.13, it appears that the simple order-up-to S rule is still 9.9% away from optimal. The bounded order-up-to S rule and the 2D-rule performs slightly better. The target stock rule is closest to the optimal annual cost level of 36,605 euros per year. Given the bad fit of the Fixed-order-quantity rule, it is not surprising that it gives extremely high annual costs.

Conclusions

Under a (LIFO, FIFO) issuing policy, the following conclusions are drawn for different ordering policies:

1. The simple order-up-to S rule, with order-up-to levels S_d that depend on the day of the week, results in an average cost level that is not far from the optimal cost level,
2. Bounding the production volumes improves the order-up-to S rule by reducing the occurrence of mismatching,
3. The 2D-rule reduces mismatching, since it has order-up-to levels for the 'young' stock, next to order-up-to levels for the total stock.
4. Even better results are obtained by Final-stock rule, which aggregates the stock that survives until the next day as if average demand happens during the day.

All rules of the order-up-to type perform almost equally well with outdating of about 2% and shortages less than 0.5%. The difference between the rules may change when another data set of the PPP is used. For practical use we propose to adopt the ordinary order-up-to S rule with only one parameter per day of the week. This rule is subject of further study to justify its use for solving the original unscaled PPP.

3.6 Step 5 – Re-scaling and validation of simple rule

In the last step of our approach, we validate the application of an ordinary order-up-to S rule to problems of realistic size. We return to the original, unscaled PPP, in which both demand and production decisions are in BPPs instead of batches. We consequently discuss:

1. a simulation model by which detailed performance statistics are obtained,
2. the disaggregation of the production volumes by re-scaling the order-up-to levels,
3. the impact of distinguishing different partly compatible blood groups instead of a single universal product, and
4. the robustness of the approach regarding the supply of buffy coats.

In Section 3.7 we report more sensitivity studies with respect to the problem data and we provide answers to several *what-if* questions.

3.6.1 Simulation model and data

Since exact results cannot be obtained for the full size problem, all results reported for the full size problem are obtained through simulation. The simulation model relies on the same assumptions as the MDP model.

Hence the simulation runs in discrete time with fixed time increments of 1 day. The following actions and events happening during the day in the chronological order:

1. the stock is observed at the very start of a day,
2. immediately thereafter the production volume is set and expected costs are incurred,
3. during the day BPPs are selected from stock and distributed to the hospitals to meet the demand: all young-demand is met prior to the any-demand,
4. at the end of the day the ageing of the BPPs left in stock is registered: any outdated pools are removed from stock
5. finally, today's production (if any) is added to stock as BPPs with a residual shelf life of m days.

In a first model, called the *Universal-group simulation model*, the existence of blood groups is not included and the supply by donors is not restrictive. In Section 3.6.3, this model is extended with blood groups and stochastic show-up of donors of different blood groups. We call the latter model the *Multi-group simulation model*.

Data – The unscaled data reported in Section 3.3.1 are used. Most notably, given probability distribution are used or these are fitted on the mean and the standard deviation. We assume, for the moment, that demand is modeled by a Poisson distribution with unscaled means as reported in Table 3.1.

Simulation horizon – To obtain accurate estimates of the several statistics that we are interested in all simulations cover a period of 100 million weeks. According to the following exercise, this is enough to get estimate that are accurate to 3 digits.

Justification of simulation length:

Let X be the number of BPPs that outdate in a week. Presume, X is Poisson distributed with mean μ outdates per week and standard deviation $\sqrt{\mu}$. Through simulation of n independent weeks, one may obtain n independent estimates of μ: $X_1, X_2, \ldots, X_n$. The 95% confidence interval for μ has half-width $2\sqrt{\mu/n}$. The interval has relative half-width $\frac{2\sqrt{\mu/n}}{\mu} = \frac{2}{\sqrt{\mu\, n}}$. The estimate $\mu \approx \frac{1}{n}\sum_{i=1}^{n} X_i$ is with 95% confidence accurate up to 3 digits, when $\frac{2}{\sqrt{\mu\, n}} < 10^{-3}$. Hence n must be at least $4\,10^6/\mu$. When outdating is in the order of 2 per week, then n must be at least $2\,10^6$.

Events that occur less frequent, require a longer simulation horizon, e.g. when μ is in the order of 0.04 per week then a simulation for 100 million is just enough to get estimates that are with 95% confidence accurate up to three digits. Ignoring any dependencies, we assume most of the estimates reported in the next sections for the full-size problem to be accurate up to 3 digits when simulating 100 million weeks.

3.6.2 Re-scaling – disaggregation of batches into BPPs

In the original unscaled or full-size problem the demand distribution is in pools. But in the previous sections the demand and the production happens in batches of four pools. Hence the order-up-to S rule and the corresponding order-up-to levels are also in batches. Below, we discuss two ways to use the frequency tables of the optimal scaled MDP solution in order to estimate the order-up-to levels for the unscaled problem.

Re-scaling the most-frequent order-up-to levels – Re-scaled parameter values of the order-up-to S rule are obtained by multiplying the most-frequent order-up-to levels as read from the frequency tables by the factor $B = 4$. Reasonable order-up-to levels for Monday to Friday for the original PPP of real size are thus: $4 \cdot (16, 18, 18, 16, 20) = (64, 72, 72, 64, 80)$.

Re-scaling average order-up-to levels – Instead of re-scaling the most-frequent order-up-to levels, one could re-scale average order-up-to levels. Average order-up-to levels for the scaled PPP are obtained from the frequency table by multiplying the first and the last columns of the tables in Table 3.7, and divide the result by the simulation length (100,000 weeks). For example the re-scaled average order-up-to level for Thursday morning is computed as follows:

$$S_4 = 4 \cdot \frac{17 \cdot 445 + 16 \cdot 73,213 + 15 \cdot 26,263 + 14 \cdot 3 + 0 \cdot 76}{100,000} \approx 63.$$

After re-scaling and rounding the average order-up-to levels, the following set of order-up-to levels is found: $(62, 72, 73, 63, 80)$. These values are close to the previously found levels.

Local search

To check whether re-scaling indeed results in nearly-optimal parameter values for the full size problem, the parameter values are given as input to a simulation-based search procedure. We have constructed an incremental search procedure for this purpose, which is presented Appendix C. The algorithm start with some set of order-up-to levels, e.g. with all $S_d = 0$, and successively determines which order-up-to levels to increment by one. The incremental search stops when improvements stay out for a number of iterations. Alternatively, one may start with very high levels and successively decreases the best levels. As the algorithm may not to all data sets, we simply use the algorithm to verify the re-scaled order-up-to levels.

Since the order-up-to S rule has five parameters there are many local optima at which the incremental search procedure might get stuck. The local search procedure is only used as a verification tool that is not a crucial part of the SDP-Simulation approach. The locally optimal order-up-to levels found by the algorithm are $(65, 74, 80, 64, 82)$ for Monday to Friday.

Evaluation of different sets of order-up-to levels

In Table 3.14 the performance of the order-up-to S rule for different order-up-to levels in the unscaled problem is reported.

- First of all, we observe that the order-up-to S rule is quite robust with respect to the chosen order-up-to levels.
- Secondly, we observe that the relative outdating, shortage and quality mismatch statistics are very close to the scaled case. Not even 2% of the production becomes outdated and less than a half percent of the demand is mismatched or cannot be delivered directly from stock.
- Thirdly, it appears that the annual costs differ from the annual costs for the scaled problem, which were reported in Table 3.12. The discrepancy could be due to the fact that the scaled demand distributions are not fitted on third or higher moments.

Simply re-scaling the most-frequent order-up-to levels suffices to find a nearly optimal order-up-to rule for the full-size PPP. Fine-tuning the order-up-to levels gives only a slight reduction of the cost level, but hardly influences the occurrences of outdating and shortages. As the simulation-based search approach may be time-consuming and could run into a bad local optimum, the SDP-Simulation approach is preferred as it also gives insight in the optimal policy.

Table 3.14: Performance of the order-up-to S rule under different order-up-to levels S. (simulation of 100 million weeks).

Re-scaled Order-up-to levels	Outdating [a]		Shortage [a]		Quality mismatch [b]		Mean age	Annual costs
Most-frequent levels								
$S = (64, 72, 72, 64, 80)$	142	1.9%	33	0.44%	9	0.18%	2.04	47,726
Average levels								
$S = (62, 72, 73, 63, 80)$	137	1.8%	33	0.44%	14	0.28%	2.03	47,677
Optimized levels								
$S = (65, 74, 80, 64, 82)$	191	2.5%	19	0.26%	11	0.21%	2.11	45,205

[a] in pools per year; % of total (7488 pools),
[b] in pools per year; % of young-demand (4992 pools).

3.6.3 Disaggregation of universal BPPs into blood groups

Thus far we ignored the eight different blood groups and the stochastic supply of buffy coats. In this section we show that the aggregation of the BPPs of different blood groups into a single universal product BPP is justified, at least for the Dutch case. Remember from the introduction of the PPP in Section 2.1, that in The Netherlands about a third to a half of the buffy coats is processed to BPPs.

The current practice

The donor management team calls donors by phone or mail to make an appointment for donating whole blood. As we have learned in Section 2.1 one prefers donors with the universal blood group O^-. Therefore one tends to call relatively many O^- donors, but O^- donors are scarce and donors normally donate only twice a year (on a voluntary basis). In Figure 3.4 the compatibility of the blood groups is shown once again.

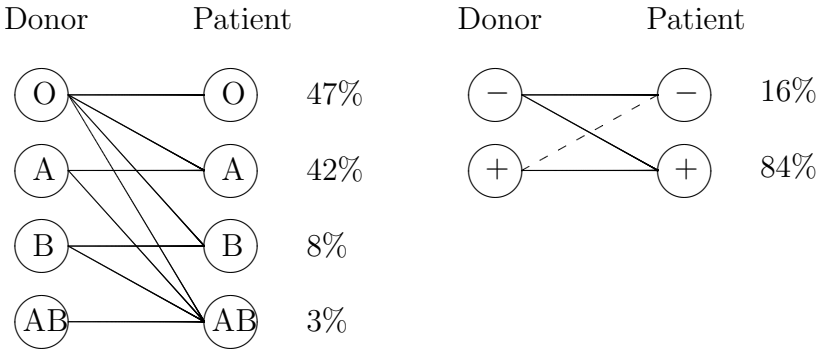


Figure 3.4: Compatibility of blood groups of donor and patient following the ABO-system and Rhesus-D system (+/−)

After a first processing step to separate red blood cells from a whole blood donation, the residual called the buffy coat is processed. To meet the demand for BPPs, on average only a third of the buffy coats needs to be processed. Hence, one may select the most favorite (=most-compatible) buffy coats for processing BPPs. Blood banks primarily produce BPPs from donor groups O and A, since compatible matching is the standard. In Figure 3.4, one reads that about 90% of the Western population has either blood group O or A. Patients of the B and AB type can be helped with O pools. Hence virtually all demand is met by pools of blood group O and A. With respect to the Rhesus-D factor we note that not even 50% of the RhD^- patients requires strict matching. This is only 8% of the patient population.

The Multi-group simulation model and data

In the what-we-call *Multi-group simulation model*, the number of donors of each of the eight different blood groups that show-up, is drawn from eight different Poisson distributions. Similarly eight different Poisson distribution are distinguished for each type of demand.

The model is an extension of the Universal-group simulation model. Below we explain in more detail, how is dealt with the distinction of blood groups.

Donor and patient population – We assume that the (relative) frequencies of the blood groups for the donor and patient population correspond to the Western population. Hence we do not over-represent the more compatible donor groups.

Donations – The number of donors to call is usually set primarily by the demand for RBCs. We assume that the average number of appointments made by the donor management team is the same from Monday to Thursday, but on average on Friday 10% more donors are scheduled to anticipate the production stop during the weekend. One may not always succeed in making enough appointments and some appointments are canceled by the donors on the day of donation. Modeling the number of whole-blood donations by a Poisson distribution may overestimate the uncertainty in the show-up of donors we accept to stay at the safe side.

The average weekly demand for 144 BPPs requires 720 buffy coats. When on average 2160 whole blood donations are taken per week, on average only a third of the buffy coats is used for processing BPPs. The mean number of donations on Monday – Thursday is thus $\frac{2160}{1+1+1+1+1.1} = 423.5$. On Friday the mean is $1.1 \cdot 423.5 \approx 466$ donations. We assume that no blood groups are over-represented compared to in the distribution of the blood groups in most Western countries. Hence, on Friday the number of whole blood donations of blood group O^- is Poisson distributed with mean $47\% \cdot 16\% \cdot 466 = 35$ donations. Thus on average only 7 pools of this blood group can be produced on Friday.

Production – The required aggregated production volume suggested by the order-up-to S rule is reached by producing as many pools as possible of the *most-compatible donations*. With most-compatible donation, we mean donations of a blood group compatible to the greatest part of the patient population. O^- donors can help all patient groups, O^+ is compatible to $84 + 0.5 \cdot 16 = 92\%$ of the patient population, etc. The order of most-compatible to least-compatible donation is O^-, O^+, A^-, A^+, etc.

Order of patient groups – Before distributing the BPPs to the hospitals, the stock is matched to the demand. One has some freedom in setting the order in which the different patient groups are considered. We meet the young-demand prior to meeting the any-demand. Upon meeting the demand for each of these groups, the demand set by the *least-compatible patients* is met first. A least-compatible patient has a blood group that is compatible to only a small fraction of the donations. O^- patients can receive only O^- which is offered by $47\% \cdot 16\% \approx 7.5\%$ of the donors, B^- patients rely on O^- and B^- donors, which is $(47\% + 8\%) \cdot 16\% = 8.8\%$ of the donor population. The order from least-compatible to most-compatible patient is: O^-, B^-, A^-, AB^-, O^+, B^+, A^+, AB^+. One could say the a least-compatible patient is least flexible in receiving BPPs. Since demand is met in this order, AB^+ patients seem to have the highest chance of experiencing a shortage. But a shortage to a AB^+ patients is most easily resolved, since AB^+ patients may receive BPPs of any blood group.

Detailed issuing policy – Now that we have specified an order in which demand is modeled, we need to specify the order in which BPPs are selected. In matching the stock to the demand, one tries to issue the least-compatible BPPs first. Hence in principle stock is issued in the order AB^+, AB^-, B^+, B^-, A^+, A^-, O^+, O^-, etc., provided that the BPPs are compatible with the blood group of the patient. If the issuing policy is (pure) FIFO, then first BPPs of the oldest categories are issued in the order of least-compatible blood group. Next younger pools are issued. Under LIFO the youngest pools in stock that are compatible to the patient's blood group, are issued before older pools are considered. In the PPP we deal with two demand categories; the young-demand receives the youngest compatible pools in stock (LIFO), the any-demand is met by the oldest compatible BPPs in stock (FIFO).

Through the above refined issuing policy, we expect to keep the occurrence of shortages and outdating low. We do assume that all demand is known upon distributing the pools to the hospitals, although in practice a portion of the demand occurs after the distribution early in the morning. Sharing this information with the production department is beneficial, but we consider a more difficult case where the production department has no information about the demand other than probability distributions obtained from historical data as in current practice.

In the remainder, we discuss the results obtained from the Multi-group simulation model. Besides showing how the production, outdating and shortages are spread over the blood groups, we will consider cases where buffy coats are more scarce.

Results – impact of distinction of blood groups

We study the impact of the distinction of blood groups by comparing the results obtained from the Multi-group simulation model against those obtained through the Universal-group simulation model. The results as presented in Table 3.15 are estimates accurate up to 2 or 3 digits. The annual costs in the Multi-group model are only 1% higher than in the Universal-group model. The differences in shortages, outdating and quality mismatches are virtually zero. It thus seems justified to ignore the blood groups in the study of nearly optimal production rules for the Dutch case.

Table 3.15: Annual performance of the order-up-to S rule under $I = (\text{LIFO}, \text{FIFO})$: the impact of blood groups in the Multi-group model (simulation for 100 million weeks).

	Multi-group model		Universal-group model	
Criterion	in pools	relative %	in pools	relative %
Production	7,597		7,597	
Outdating	143	1.9%	142	1.9%
Shortage	33	0.4%	33	0.4%
Quality mismatch	9	0.2%	9	0.2%
Annual costs	48,168		47,726	

Results – production, outdating and shortages per blood group

Although blood groups are not relevant when setting the (aggregated) production volume, it is of interest to study whether certain patient groups face an excessive shortage rate under the re-scaled order-up-to S rule and the (refined) (LIFO, FIFO) issuing policy. Table 3.16(a) reports on the annual production, outdating and shortages of pools of the eight different blood groups. Table 3.16(b) presents the relative outdating, shortage and quality mismatch figures. We conclude that when on average only a third of the buffy coats is actually used for the production of BPPs, virtually all BPPs are of blood group O. Since O^+ and O^- suits to 92%, respectively 100% of the patients, shortages and mismatches are very rare. The outdating percentage is with 5.5% the highest for the O^- pools, since BPPs of the universal blood group are issued only when no pools of the same age of other blood groups are in stock. Overall outdating is very low: only 1.9% overall blood groups.

Table 3.16: Production, outdating and shortages per blood group under the order-up-to S rule with re-scaled most-frequent order-up-to levels (over 100 simulation runs of 1 million weeks each) when on average a *third* of the buffy coats is processed into BPPs.

(a) Absolute volumes in pools per year

	Total	O^-	O^+	A^-	A^+	B^-	B^+	AB^-	AB^+
Production	7597	1582	5944	62	10	0	0	0	0
Outdating	142.8	86.5	56.3	0	0	0	0	0	0
Shortage	33.2	0.6	7.8	0.8	19.3	0.1	2.2	0.1	2.4
Quality mismatch	9.4	0	0.1	0.2	7.2	0	0.1	0	1.7

(b) Relative volumes

	Total	O^-	O^+	A^-	A^+	B^-	B^+	AB^-	AB^+
Outdating	1.9%	5.5%	0.9%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Shortage	0.4%	0.2%	0.2%	0.3%	0.7%	0.3%	0.4%	0.4%	1.1%
Quality mismatch	0.2%	0.0%	0.0%	0.1%	0.4%	0.0%	0.0%	0.2%	1.2%

Overall shortages and quality mismatches remain well below 0.5%. The AB^+ patients, who constitute less than 3% of the entire population, have the highest probability of experiencing a shortage (1.1%) or a mismatch (1.2%), since the AB^+ demand is met after all other demand is fulfilled.

3.6.4 Sensitivity – scarcity of buffy coats

When on average less donors show up, one uses relatively more of the buffy coats obtained after processing RBCs. As a consequence of the limited availability of O donors, one must use buffy coats of the less compatible blood groups (occasionally one may even not be able to produce the desired number of pools). In the last two rows of Table 3.17 we show that when on average two third of the buffy coats is used instead of only a third, a significant part of the production is of blood group A and on average 108 of the 7597 BPPs are of blood group B or AB.

The scarcity of buffy coats of group O affects the occurrence of shortages and outdating. In Table 3.18 we show that even when on average two third of the buffy coats are used, overall outdating, shortages and mismatches are still very low. Patients from blood group O^- and B^- experience a high shortage probability (2.5% respectively 4.4%), because on

Table 3.17: Production per blood group under the re-scaled order-up-to S rule (over 100 simulation runs of 1 million weeks each) when on average a third, a half or two third of the buffy coats is processed into BPPs.

Scenario	Total	O^-	O^+	A^-	A^+	B^-	B^+	AB^-	AB^+
33% of Bc used									
Annual production	7597	1582	5944	62	10	0	0	0	0
% of total production		20.8%	78.2%	0.8%	0.1%	0%	0%	0%	0%
50% of Bc used									
Annual production	7597	1018	5268	525	786	0	1	0	0
% of total production		13.4%	69.3%	6.9%	10.3%	0%	0%	0%	0
67% of Bc used									
Annual production	7597	739	4201	560	1991	8	86	0	14
% of total production		9.7%	55.3%	7.4%	26.2%	0.1%	1.1%	0.0%	0.2%

average only 9.8% of the produced BPPs is compatible to O^- or B^- patients. Relative outdating figures are high for BPPs of the less favorite blood groups B and AB, but the absolute volume is very low since BPPs of these blood groups are rarely produced.

Table 3.18: Production, outdating and shortages per blood group under the order-up-to S rule with re-scaled most-frequent order-up-to levels (over 100 simulation runs of 1 million weeks each) when on average *two thirds* of the buffy coats (Bc) is processed to BPPs.

(a) Absolute outdating, shortage and mismatching volumes in pools per year

	Total	O^-	O^+	A^-	A^+	B^-	B^+	AB^-	AB^+
Production	7597	739	4201	560	1991	8.0	86	0.0	14
Outdating	175	30.3	52.7	34.4	48.2	0.8	5.0	0.0	3.6
Shortage	64.7	7.1	27.0	1.8	15.4	2.1	9.3	0.2	1.9
Quality mismatch	10.9	1.2	0.4	0.5	5.7	0.6	0.9	0.1	1.4

(b) Relative outdating, shortage and mismatching

	Total	O^-	O^+	A^-	A^+	B^-	B^+	AB^-	AB^+
Outdating	2.3%	4.1%	1.3%	6.1%	2.4%	9.4%	5.8%	37.2%	26.8%
Shortage	0.9%	2.5%	0.8%	0.7%	0.5%	4.4%	1.7%	1.1%	0.9%
Quality mismatch	0.2%	0.6%	0.0%	0.3%	0.3%	1.9%	0.2%	0.7%	1.1%

Remark – We believe that it is not of interest to study under the same issuing and ordering policy a scenario where the fraction of buffy coats used is close to 1. When buffy coats are very scarce, one is not in the luxurious position to issue the youngest BPPs in stock to meet the young-demand. Furthermore, more advanced ordering strategies might perform better in that case. In the extreme case where on average all buffy coats are used to meet the demand, one may expect to process virtually all buffy coats into BPPs to anticipate the uncertain show-up of donors, causing that the desired production volumes are often not realized. We will not study this extreme case.

3.6.5 Conclusions for full-size PPP

The SDP-simulation approach is justified to solve the PPP at a Dutch blood bank. From a simulation study using realistic data we conclude:

1. The shortage, outdating and mismatch figures for the scaled PPP are a good approximation for the full-sized PPP,
2. An ordinary order-up-to S rule performs very well, even when patients and BPPs are of different blood groups,
3. Re-scaling the order-up-to levels yields nearly optimal order-up-to levels,
4. The order-up-to S rule appears to be robust with respect to minor changes in the order-up-to levels,

Consequently, when searching for good order-up-to levels, one can ignore the existence of blood groups and the uncertainty at the supply side. This observation is of practical importance, since it simplifies the search for good order-up-to levels and allows a fast evaluation of different scenarios by the Universal-group simulation model. In the next section we extend the sensitivity study by changing the problem data and we answer some prominent *what-if* questions.

3.7 Sensitivity analysis for the PPP

In the previous sections we have shown that an ordinary order-up-to S rule resembles the optimal ordering strategy for the PPP under consideration. The SDP-simulation approach yields after re-scaling nearly optimal order-up-to levels. In this section we put this rule and our approach for reading the best parameter values to the test. As argued in the previous section the study is done using the 'universal-group' simulation model. We are interested in the robustness and performance under various scenarios, we therefore answer the following questions:

- How robust is the rule when the estimates are far off from the realized mean demand?
- What if (part of) the demand is even more uncertain than Poisson?
- What if shortage costs are set lower or higher than in the base case?
- What if the maximal shelf life is extended to 6 or even 7 days?

The first two questions are interesting scenarios from a practical point of view: no matter how careful the average demand is predicted, the realized demand over a fixed period can be somewhat off. Further the demand may be somewhat more uncertain than Poisson distributed demands.

The latter two questions are more controlled. The cost structure is a matter of choice, that should resemble the preference of the stake holders: the blood bank managers and the doctors at the hospitals. When regulations by law and clinical test allow doing so, the maximal shelf life is in the current practice increased from 5 to 7 days.

We refer back to the previous section for a discussion of the impact of the issuing rule, the distinction of blood groups and the scarcity of donors.

3.7.1 Average demand estimates

Input to the optimization model are estimates for the mean demands, amongst other. These estimates might be a few percent off from what turns out to be the mean over the coming period, of 13 weeks (one quarter). When in the demand turns out to be higher than expected, shortages are more likely. When the average demand is below the expected value, then more BPPs become outdated.

Table 3.19: Performance of the order-up-to S rule with fixed levels $S = (64, 72, 72, 64, 80)$, when the average demand is on all weekdays 5 or 10% above (+) or below (−) the predicted mean demand. (simulation of 100 million weeks).

% off from estimate	Outdating [a]		Shortage [a]		Quality mismatch [b]		Mean age	Annual costs
−10% off	388	5.4%	6.0	0.09%	12.4	0.28%	2.19	65,127
−5% off	247	3.4%	14.1	0.20%	10.5	0.22%	2.12	49,820
0% off	142	1.9%	33	0.44%	9.2	0.18%	2.04	47,726
+5% off	73	0.93%	70	0.89%	8.1	0.16%	1.94	64,856
+10% off	34	0.41%	134	1.6%	7.2	0.13%	1.84	107,052

[a] in pools per year; % of total (7488 pools),
[b] in pools per year; % of young-demand (4992 pools).

In the unscaled case, the mean demand is Poisson distributed with mean 32 BPPs. The expected demand over 13 Wednesdays is thus $13 \cdot 32 = 416$ BPPs. The corresponding standard deviation is $\sqrt{416} \approx 20$ BPPs. A 67%-confidence interval of the demand over any 13 Wednesdays, is $[416 - 20, 416 + 20] = [396, 436]$. A 95%-confidence interval of the demand is $[376, 456]$. When one considers an arbitrary quarter, then with 2.5% chance the average demand is more than 2 standard deviation, or $\frac{40}{416} \approx 10\%$, higher than the true mean 416 BPPs. With probability $\frac{100\%-68\%}{2} = 16\%$, the demand over 13 weeks is $\frac{20}{416} \approx 5\%$ above the expected demand of 416 BPPs.

In Table 3.19, we consider a number of scenarios regarding the difference between the expected demand volumes and the actual demand volumes. Differences may arise by nature: not all periods of 13 weeks are the same, or they may be structural when trends are broken. Suppose the demand on all weekdays is 5% or even 10% below the expected mean demand, then the order-up-to levels are set too high. By simulation we check whether outdating then becomes a big issue. For the case where the mean demand on all weekdays is structurally 5% or 10% higher, we check whether this results in many shortages. Furthermore, we expect that the age of the pools drops when the demand is higher than expected.

The robustness of the order-up-to S rule, is checked for five scenarios: the average demands for each day of the week is consequently −10%, −5%, 0%, +5% and +10% off from the estimated mean demands. The order-up-to levels are fixed to $\mathbf{S} = (64, 72, 72, 64, 80)$. To show the robustness of the approach, we assume all deviations to be in the same direction on all weekdays.

The results show that deviations up to +5% or −5% do not cause many shortages and outdating: shortages are still less than 1% and average outdating ranges from 0.93 to 3.4 percent. In the extreme case where the mean demands deviate +10% or −10%, the outdating and shortages are still moderate compared to the current practice. On the one hand, when the demand is 5% lower than expected, the average number of outdated pools roughly doubles. On the other hand, the average number of shortages roughly doubles when demand on all days is 5% higher than expected. The shortages rate is 1.6%, when all demand is 10% higher than expected. As shortages are more critical than the outdating of BPPs, overestimating the demand is less critical than underestimation.

3.7.2 Demand uncertainty: higher coefficient of variation

In this section we investigate the sensitivity of an order-up-to S rule with respect to the uncertain in the demand. We increase the uncertainty in the any-demand; the young-demand is still Poisson distributed. Under Poisson demand the coefficient of variation, cv_d^{any}, of the any-demand on day d, would be $\frac{1}{\sqrt{\mu_d}}$ for weekday d, where μ_d is the average demand at day d. Now we set $cv_d^{\text{any}} = \frac{2}{\sqrt{\mu_d}}$. We expect to find higher order-up-to levels, such that the occurrence of shortages is kept low at the expense of a few more outdated BPPs.

By simulations of 100 million weeks, we evaluate the ordinary order-up-to S rule for three sets of order-up-to levels:

$\mathbf{S} = (64, 72, 72, 64, 80)$ as obtained by solving the scaled PPP with Poisson demand ($cv_d^{\text{any}} = \frac{1}{\sqrt{\mu_d}}$), and re-scaling the most-frequent order-up-to levels,

$\mathbf{S} = (72, 80, 80, 72, 88)$ as obtained by solving the scaled PPP with $cv_d^{\text{any}} = \frac{2}{\sqrt{\mu_d}}$, and re-scaling the most-frequent order-up-to levels,

$\mathbf{S} = (69, 79, 86, 67, 87)$ the locally optimal order-up-to levels obtained by local search for the unscaled PPP with $cv_d^{\text{any}} = \frac{2}{\sqrt{\mu_d}}$.

The second set of order-up-to levels are 8 pools higher than when demand was less uncertain as in the Poisson demands case. The third set is introduced to show that there is no need to fine-tune the re-scaled levels of the second set, as will become clear from Table 3.20.

In the table one finds the performance of the order-up-to S rule for the given choices of the order-up-to levels. As expected, underestimation of the degree of uncertainty results in

Table 3.20: Performance of the order-up-to S rule under different order-up-to levels S for the case where cv^{any} is doubled and I =(LIFO, FIFO). (simulation of 100 million weeks).

Order-up-to levels	Outdating [a]		Shortage [a]		Quality mismatch [b]		Mean age	Annual costs
Re-scaled, Poisson demand								
$S = (64, 72, 72, 64, 80)$	264	3.5%	129	1.7%	26	0.52%	1.98	141,686
Re-scaled, $cv_d^{\text{any}} = \frac{2}{\sqrt{\mu_d}}$								
$S = (72, 80, 80, 72, 88)$	524	6.6%	50	0.66%	18	0.37%	2.12	119,474
LS, double cv_d^{any}								
$S = (69, 79, 86, 67, 87)$	472	6.0%	53	0.70%	28	0.56%	2.12	116,002

[a] in pools per year; % of total (7488 pools),
[b] in pools per year; % of young-demand (4992 pools).

relatively many shortages. Using accurate estimates of the coefficient of variation results in lower shortages but higher outdating, in the order of 6%-6.6%. The local search reduces the outdating a bit, but this at the expense of more shortages and mismatching. To keep the number of outdated pools low, the order-up-to levels should be kept low, e.g. to those for the Poisson demand case, but than shortages become more prevalent and consequently the average cost level increases.

Return to Step 2 and 3 – investigate structure of optimal strategy

Fine-tuning the order-up-to levels improves the order-up-to S rule only slightly. This might raise the question whether the order-up-to S rule is far from optimal when demand is more uncertain and whether other rules are much better. This question can only be answered by returning to step 2 and 3 of our approach: we investigate the optimal strategy for the scaled PPP with $cv_d^{\text{any}} = \frac{2}{\mu_d}$.

Under the optimal MDP strategy, 7.1% of the produced batches become outdated and shortages and mismatches are on average less than 0.1%. The optimal production volumes depend heavily on the age of the pools in stock. In Table 3.21 we report the goodness-of-fit of some simple rules. These goodness-of-fit which are read from simulation-based frequency tables, which we do not report here.

We conclude that an order-up-to S rule fits in many states visited, but a bounded order-up-to S rule seems to fit much better. Due to the increased uncertainty in the demand, the goodness-of-fit figures are much lower than under Poisson demand. We do not elaborate on setting a better rule for this particular case, but we stress that the SDP-simulation

Table 3.21: Goodness-of-fit of some production rules as read from frequency tables for $I = (\text{LIFO}, \text{FIFO})$ and the any demand (=FIFO demand) is twice as uncertain.

Rule	Mon	Tue	Wed	Thu	Fri
Fixed production rule					
Goodness-of-fit	34%	32%	30%	25%	27%
Fixed production volume [a]	7	9	6	7	10
Order-up-to S rule					
Goodness-of-fit	34%	66%	41%	77%	59%
Fixed order-up-to level[a]	18	20	20	18	22
Bounded order-up-to rule					
Goodness-of-fit	64%	72%	48%	77%	68%
Fixed level[a]	18	20	20	18	22
Maximum production[a]	8	11	9	10	15
Minimum production[a]	6	7	5	0	9

[a] in batches of 4 pools.

approach can be used to find a good ordering rule that is a good compromise between quality and simplicity.

3.7.3 Cost ratios

In order to change the trade-off between outdating and shortages, we show what is the impact of major changes in the shortage costs. The outdating costs are fixed to 150 per outdated pool. Although the mismatch cost is also a kind of shortage cost, we keep it fixed to 200 per pool.

In the base case we have taken shortage costs equal to 5 times the outdating costs: a penalty of 750 euro applies for each pool demanded that cannot be met immediately from stock. In this section we show the impact of lowering and raising the shortage costs on the performance statistics.

In Table 3.22 we report results of a long simulation run of the order-up-to S rule with re-scaled order-up-to levels obtained from the scaled MDP which is solved for shortage costs set to 300, 750, and 1500 euros respectively. When shortage costs are high, one tends to keep many pools in stock. Consequently the available stock is old more BPPs outdate.

Table 3.22: Performance of the order-up-to S rule with re-scaled order-up-to levels for increasing shortage costs. (simulation of 100 million weeks).

Order-up-to levels	Outdating [a]		Shortage [a]		Quality mismatch [b]		Mean age
Shortage cost = 300							
$S = (60, 72, 72, 60, 80)$	127	1.7%	37	0.49%	23	0.46%	2.00
Shortage cost = 750							
$S = (64, 72, 72, 64, 80)$	142	1.9%	33	0.44%	9.2	0.18%	2.04
Shortage cost = 1500							
$S = (64, 72, 72, 64, 84)$	226	2.9%	19	0.26%	26	0.53%	2.05

[a] in pools per year; % of total (7488 pools),
[b] in pools per year; % of young-demand (4992 pools).

Remarkably, compared to the base case, where shortage costs are 750 euro per BPP short, mismatches happen more frequently both when the shortage cost are set higher or lower than 750. When shortage costs are higher, mismatching is higher since the pools in stock are on average older. When shortage costs are low the stock levels are set lower, which makes both shortages and mismatching more likely.

Other changes in the cost-structure are possible. For example, we could study more refined penalties on mismatching. Instead of fixing the mismatch costs to 200 euro per mismatched BPP irrespectively of the age of the BPP. More refined cost incentives for issuing 'young' BPPs can be modeled but are not considered throughout this thesis. Fixed order costs are studied in the next chapter.

3.7.4 Shelf life m

Blood banks are allowed to raise the maximal shelf life of the platelet pools, when clinical studies shows that the quality of a BPP upon transfusion is guaranteed to be high enough[1]. In some countries the shelf life is effectively only 4 days, whereas in most other countries the shelf life is 5 days and current efforts are focused on raising the shelf life to 6 or even 7 days. In this section we will study the importance of increasing the shelf life from an inventory management point of view.

[1] The quality of a pool (of 5 donor units) is sufficient when it contains more than 250 10^9 active platelets. Upon production immediately after donation this number is normally around 350 10^9, but decreases over time.

Table 3.23: The most-frequent order-up-to levels for the scaled case at varying maximal shelf life and young-demand prefers a residual shelf life of at least 3 days.

Maximal shelf life	Order-up-to levels as read from scaled MDP	Re-scaled order-up-to levels
$m = 4$	$(15, 17, 17, 16, 20)$	$(60, 68, 68, 64, 80)$
$m = 5$	$(16, 18, 18, 16, 20)$	$(64, 72, 72, 64, 80)$
$m = 6$	$(16, 18, 18, 16, 21)$	$(64, 72, 72, 64, 84)$
$m = 7$	$(16, 18, 18, 16, 22)$	$(64, 72, 72, 64, 88)$

When the maximal shelf life m is increased, one should redefine the 'young' demand. With $m = 5$ all young-demand strongly prefers pools with a residual shelf life of at least $l = 3$ days. The young-demand is thus mismatched when BPPs with a residual shelf life of 2 days or less are issued. We stick to this definition: independent of the maximal shelf life m, the age preference for the young-demand is a residual shelf life of $l = 3$ days. Alternative definitions are considered but not reported here. Mismatch costs are kept to 200 euro per mismatched pool (or 800 per batch of 4 pools).

Under this definition all young-demand on Monday will be mismatched when the maximal shelf life is only 4 days.

By executing the first four steps of our approach, for maximal shelf life varying from 4 to 7 days, we read simple order-up-to S rules with most-frequent levels for the scaled case. Since we will focus on the quality of the simple order-up-to S rule, we do not consider alternative production rules that might better fit to the optimal strategy. Table 3.23 display the several order-up-to levels for the scaled cases.

Apparently the maximal shelf life hardly affects the order-up-to levels. Most prominent are the higher order-up-to level for Friday when the maximal shelf life increases from 5 to 7 days. The higher order-up-to level on Friday reduces shortages on Monday, while the longer shelf life makes outdating less of a problem. These statements are verified by looking at the details from the simulation of the re-scaled order-up-to S rule for the full-size PPP.

Again the order-up-to S rules are simulated for 100 million weeks. The results are tabulated in Table 3.24. Raising the maximum shelf life from 4 to 5 days is most beneficial considering the reduction of outdating and shortages. As expected mismatches are very high when $m = 4$, since all any-demand on Monday must be mismatched under the given definition of the 'young' demand. Outdating and shortages roughly halves when the shelf life is increased by one day. Consequently also the cost figures halves when the shelf life

Table 3.24: Performance of the order-up-to S rule with re-scaled order-up-to levels for varying maximal shelf life and young-demand prefers BPPs with a residual shelf life of at least 3 days. (Simulation results over 100 million weeks).

	Outdating [a]		Shortage [a]		Quality mismatch [b]		Mean age	Annual costs
$m = 4$ days								
$S = (60, 68, 68, 64, 80)$	311	4.0%	62	0.82%	1056	21.15%	1.86	304,015
$m = 5$ days								
$S = (64, 72, 72, 64, 80)$	142	1.9%	33	0.44%	9.2	0.18%	2.04	47,726
$m = 6$ days								
$S = (64, 72, 72, 64, 84)$	88	1.2%	15	0.21%	6.7	0.13%	2.15	26,068
$m = 7$ days								
$S = (64, 72, 72, 64, 88)$	42	0.56%	8.1	0.11%	2.1	0.04%	2.25	12,760

[a] in pools per year; % of total (7488 pools),
[b] in pools per year; % of young-demand (4992 pools).

is lengthened by 1 day, except for $m = 4$: the costs when $m = 4$ are excessively high due to the mismatches on Monday. At maximal shelf life 7, both outdating and shortages are very low, hence lengthening the shelf life even more gives only a slight improvement.

A drawback of an increased shelf life is that the average age of the BPPs upon meeting the demand is somewhat higher. By definition the age of a BPP is at least one day, at $m = 7$ the mean age of the issued BPPs is 2.25 days, while this is only 1.86 days when $m = 4$. The mean age of a BPP is of secondary importance in our main study. In Kortbeek, Van der Wal, Van Dijk, Haijema, and De Kort [77] a number of ways is discussed to use the SDP-Simulation approach for finding good order-up-to levels when transfusing young pools is even more important.

3.8 Discussion and Conclusions

This section is split into two parts. First the SDP-Simulation approach is discussed, next its application to the PPP and the results are examined.

3.8.1 SDP-Simulation approach – Discussion and conclusions

Although the platelet production-inventory problem (PPP) is a high-dimensional problem, the problem exhibits enough structure to formulate an MDP problem and solve it. Solving the MDP requires the aggregation of states such that an SA algorithm can be executed in a reasonable amount of time. Aggregation is needed at several levels. First of all, the blood groups are to be aggregated as if blood platelet pools (BPP) are of a single universal blood group. Secondly, BPPs are aggregated in batches when the PPP plays at a scale that is too large. Finally, to come up with a simple rule, the stock consisting of batches of different age categories are aggregated into a single number. By simulation the structure of the optimal strategy is investigated and approximated by simple rules.

For the PPP with realistic data from a Dutch blood bank, aggregation at all three levels is possible and needed. For the given data, it appears that an order-up-to S rule roughly fits to the optimal ordering decisions. When other data, e.g. from a smaller blood bank or with more uncertainty in the demand, is used, more advanced stock-age-dependent rules may be needed.

The combined SDP-Simulation approach appears to be a very powerful approach to obtain insight in the structure of the optimal ordering strategy for perishable inventory problems, such as the PPP. It is a fast approach for:

- learning about simple rules that resemble the optimal strategy and
- obtaining their respective nearly optimal parameter values; e.g. nearly optimal order-up-to levels (and bounds) are easily read from frequency tables generated by simulation of the optimal strategy.

In this discussion section we elaborate on solutions to overcome some computational difficulties that one may experience when applying the approach to other perishable inventory problems. Next to the computational difficulties related to the available computation time, one may have difficulties in fitting a scaled demand distribution. In rare cases fitting a distribution on scaled demand figures is infeasible. We discuss these difficulties and ways

of resolving them. First, we discuss the computation times of the solution procedures to give an idea of the computational complexity and the necessity of scaling the problem. Finally we raise the question to which extend the approach can be extended to solve a finite horizon problem, such that it can deal with non-stationary problems.

Computation times

Core of the SDP-Simulation approach is solving the (scaled) MDP and simulating the thus obtained optimal strategy. The MDP is solved by an SA algorithm, an iterative scheme that is relatively easily coded in any programming language, i.e. in our case in Delphi-Pascal. The computational complexity of an SA algorithm is already discussed in Section 3.2.6. To give an indication of the computation time of the algorithm, we report in Table 3.25 the time it takes to solve the PPP for varying values of the maximal shelf life m by SA on a PC with a Pentium-D 2.8GHz (dual) processor with 1Gb RAM. The absolute running times serve as an indication (and can be reduced by streamlining the code and cutting out any flexibilities of the current program).

The total number of states of the scaled MDPs are significantly reduced by the production and storage capacity, such that states that are very unlikely to happen under an optimal strategy are not considered. The optimal MDP strategy can thus be computed in a reasonable amount of time. For example if the storage capacity was not imposed, the total number of states for $m = 7$ would be about 19 million, the computation time would be much higher and the storage of the value vectors would require about 300Mb. Clearly, the number of states and the running times grow more than linearly in the maximal shelf life.

Table 3.25: The computational complexity of solving the scaled PPP for varying maximal shelf live and production capacity limited to $(20, 20, 16, 16, 20)$ batches for Monday to Friday and the storage capacity set to 35 batches.

Maximal shelf life	# of states scaled PPP	Running time SA (in s.)
4	135,066	38
5	701,832	72
6	1,427,694	172
7	3,890,824	1860

Solving the MDP takes thus less than a minute for $m = 4$ to half an hour for $m = 7$.

The next step in the SDP-Simulation approach is to simulate the optimal strategy to generate frequency tables for identifying a simple rule. Since we are not interested in a very accurate estimation of the state-frequencies, a short simulation run of say 100,000 weeks suffices. Such a short simulation run takes only a couple of seconds.

The more detailed simulations for obtaining accurate estimates of the performance statistics should last much longer, say 100 million weeks. The simple Universal-group simulation model executes in 20-30 minutes, which is relatively fast given the number of statistics that are tracked. For the scaled problem one could also estimate the statistics by solving related Markov chains under different cost structures (as argued in Section 3.2). The evaluation of the average costs of the Markov chains is done in 5 seconds for $m = 4$ up to 216 seconds for $m = 7$. These figures can be lower when the required accuracy is smaller.

The Multi-group simulation model executes much slower than the Universal-group model, since much more stochastic processes have to be simulated and the issuing policy is more evolved. Given the eight different blood groups, random numbers are generated for the demand by 16 patient groups and for the supply by eight donor groups. For the full-size problem the Multi-group simulation of 100 million weeks last about 5 hours. The running time of both simulation programs grow only linearly when the maximal shelf life m is increased, whereas solving the MDP takes an in-m-exponentially growing running time.

A simulation-based search for optimal order-up-to levels using the Multi-group simulation program seems to be very time-consuming. Therefore aggregation of the blood groups and scaling the problem is favored for simulation as well as for solving the MDP.

MDP intractable for high values of m?

As argued above, and in Section 3.2.6, the running time of the SA algorithm strongly depends on the maximal shelf life m, since the state space is $(m + 1)$-dimensional. When considering the approach for solving other perishable inventory problems of products with a longer maximal shelf life, one needs to limit the time it takes to solve the MDP. We have shown that by scaling the PPP the optimal solution to the MDP can be solved (approximately) by SA for m up to 7 days. However when we consider an inventory problem for a perishable product with a higher fixed shelf life, of say 14 periods or even more, scaling the demand figures as if demand happens in batches may not resolve the computational problems (unless the demanded volumes are very low).

A simple solution is to approximate the MDP by an MDP with a shorter shelf life $m' < m$

for which the MDP is tractable (after scaling). Thus the dimensionality of the MDP is reduced from m to m', but this approximation is accurate only when in reality rarely products with residual shelf life $(m' + 1)$ days or older are in stock. This approach was first considered by Nahmias [99]. In [77], the approach is adopted as an alternative to aim at enough young BPPs in stock to meet the demand for young under an order-up-to S rule.

Another solution to reduce the dimension is to model the problem on a more coarse time scale by aggregating time, e.g. two periods are aggregated into one. When the problem is modeled as if production decisions are scheduled every two days instead of every single day, then the age of the products in stock can be expressed in multiples of two days. This modeling assumption slightly affects the accuracy in tracking outdating. The quality of such approximations are not considered in this thesis.

Infeasible fit of demand distribution?

After downsizing the average demand figures by a factor B, it might be impossible to fit a demand distribution on the first two moments. Not for all combinations of the mean demand and the coefficient of variation of the demand a fit of a discrete probability distribution is feasible, as shown in [2]. In particular when after scaling the mean demand μ is low, say less than 1, then a fit is only possible for 'high' coefficients of variation cv. As cv is copied from the full-size problem, the problem may arise at the scaled problem but not at the full-size problem.

A way to overcome such a fitting problem is choosing a smaller scaling factor B. Thus the computation time for solving the MDP will be somewhat higher due to an increase of the number of states (amongst other). Alternatively, one may modify the demand data slightly: one fits the distribution to a higher value of either cv or μ. One thus needs to accept such an modeling error. When the coefficient of variation, cv, of a particular demand category on a specific day is too low to get a fit on the first two moments, one may model the problem as if the cv is higher. Thus the uncertainty in the demand is increased, resulting in a (slight) overestimation of shortages and an underestimation of the number of outdated products.

When one chooses to increase μ for a specific patient group and day such that a fit becomes possible, then one may compensate by decreasing the mean demand of another patient group or the mean demand at some other day(s). For example consider a weekend day where only a small portion of the total demand requires 'young' BPPs. Instead of

discriminating between the two demand types all demand is modeled as demand for pools of any age. When only a small fraction of the total demand is modeled incorrectly, the impact on the overall mismatching is negligible. In the simulation of the full-sized problem the real mismatching figures can be computed to validate the results.

Non-stationary demand or production breaks

When modeling the PPP as an MDP we explicitly have assumed that the problem is periodic but stationary: every Monday is stochastically the same and production breaks happen only in weekends. In practice, blood bank managers need to anticipate irregular production breaks. For example on the two Christmas days production is hampered by a lack of voluntary donors, while demand for BPPs at hospitals continue (maybe at a somewhat lower level). The SDP-Simulation can be extended to deal with such periods, as will be discussed in the next chapter.

3.8.2 Application – Discussion and conclusions

Results for PPP

The SDP-Simulation approach is applied to the PPP introduced in Section 2.1 using data for one of the four Dutch blood banks.

- In 2003 outdating in The Netherlands was 15-20%, and shortages happened at division North-East roughly once every one or two weeks.
- We showed that outdating can be 2% (or even less) while shortages hardly happen, even when about two thirds of the demand gets the 'youngest' pools in stock.
- Just as in the current practice BPPs are produced primarily of the blood groups O and A, since compatible matching is the standard.
- A detailed sensitivity study is executed including practical issues, such as an increase of the maximal shelf life, 'biased' demand estimates, and different costs to trade-off between outdating and shortages.

Generalization

We believe that the results for the PPP at a Dutch blood bank gives an indication of what can be achieved at other blood banks of comparable size. The SDP-Simulation approach can also be applied to other blood banks and to hospitals. However, one has to be careful as the processes and the scale of the operations may differ between blood banks. Some differences one may observe in other regions and countries, and at hospitals are that:

- In some regions and countries BPPs are (also) stored in hospitals,
- The order problem at other blood banks and hospitals may play at a much smaller (or a much larger) scale,
- Delivery of fresh BPPs at hospitals may not happen daily: e.g. because fixed order costs are accounted.

We address these issues in the next chapter.

Chapter 4

Extended SDP-Simulation approach: production breaks and fixed order costs

4.1 Introduction

In this chapter we extend the SDP-Simulation approach, such that it solves non-stationary problems. Although the approach is more general we discuss the extension in the context of the platelet production problem (PPP). In practice the PPP may be virtually stationary, except for a number of periods during which additional production breaks occur because of holidays, e.g. Easter, Christmas and New Year's Day. This kind of short production breaks are known well in advance.

- Setting the right production volumes to anticipate near-future production breaks is in general difficult. Formal support through an extended SDP-Simulation approach is thus needed to support blood bank managers.

The distinction of young-demand from any-demand is not common to all blood banks. Therefore and to simplify the discussion we assume a single demand category. Another reason for doing so is the fact that when the maximal shelf life of BPPs is 5 days only, no 'young' BPPs are available after a production stop of 3 or more days. Mismatch costs are thus excluded from the model. Moreover, we simplify the discussion by meeting all demand with the oldest BPPs in stock first (FIFO), which is common to blood banks.

After applying the extended approach to one of the Dutch blood banks, we consider a case where demand volumes are 10 times as small. For the latter case one may think of a small blood bank or a medium-sized or large hospital. In The Netherlands only a few hospitals keep BPPs in stock. The lead time for regular orders is assumed to be one day.

Further, we provide answers to questions raised in the discussion at the end of the previous chapter:

- At a small blood bank or at a medium-large hospital, the coefficient of variation of the demand usually is higher. We expect that the optimal ordering policy is then more complicated. Therefore we investigate for a few cases whether an order-up-to S rule fits well or that other rules should be considered.
- When operating at a small scale fixed set-up costs may apply for each production run or fixed order costs may apply to hospitals. How do these fixed costs affect the optimal strategy? Is the SDP-Simulation approach also in this case helpful in deriving a nearly-optimal rule and the respective parameter values? In particular we are interested whether an (s, S)-policy fits well and whether nearly optimal thresholds s can be read from the frequency tables.

Answers to these questions are of interest to inventory managers at both blood banks and hospitals as well as to Operations Researchers.

Outline

In Section 4.2 we extend the SDP-Simulation approach such that it can deal with periods where production and demand may be non-stationary. Throughout the whole chapter, we assume that all demand is met by issuing the oldest BPPs in stock first, i.e. all demand is 'FIFO-demand'. Consequently mismatch costs are zero as no distinction is made between young-demand and any-demand. In Section 4.3 we present case study results for a Dutch blood bank that faces short production breaks during Christmas, New Year's Day and Easter.

Next, we consider in Section 4.4 a stationary order problem for a blood bank with much smaller average demand volumes. Alternatively one may think of a hospital that faces an average demand that is only a tenth of the demand at the blood bank. In Section 4.5 we add fixed order costs to the cost structure.

4.2 Extended SDP-Simulation approach

4.2.1 Problem: non-stationary production breaks

A practical question for production-inventory managers, and for blood bank managers in particular, that remained unanswered is: "How should one anticipate irregular production breaks like at Easter and Christmas?" When regular production stops for a few days, while demand continues (as usual) one should anticipate breaks by producing somewhat more some days before the break. Consequently the age distribution of the BPPs in stock is affected and thus the optimal production volumes immediately after a break might be different as well.

In the PPP most weeks are stochastically the same: the supply of whole blood (by voluntary donors) and the demand for BPPs are stationary processes. However, during a short holiday break production might be impossible, since donors do not show up.

As an example we consider, in Section 4.3, two cases of particular interest to blood bank managers:

1. A Christmas period (December 25 and 26) falling on Tuesday and Wednesday, followed by the New Year's Day (NYD) on the next Tuesday.

2. The four-days Easter period from Good Friday to Easter Monday. In The Netherlands the (regular) production is stopped for four consecutive days, while the maximal shelf life of BPPs is only 5 days.

In practice it is difficult to find nearly optimal production volumes on days around holidays. Due to the occasional nature of these events, it is not possible getting experienced. Formal support is thus needed. In the next section we extend the SDP-Simulation approach to include non-stationary production breaks and apply it to the BPP production-inventory management.

4.2.2 Approach

We discuss our approach by considering the breaks at a Christmas period and an Easter weekend. As the time between these breaks is very long compared the maximal shelf life of the products, one may analyze these periods independently of each other. In Figure 4.1 we illustrate our approach for the 4-days Easter weekend. The gray squared blocks on the time bar indicate production stops.

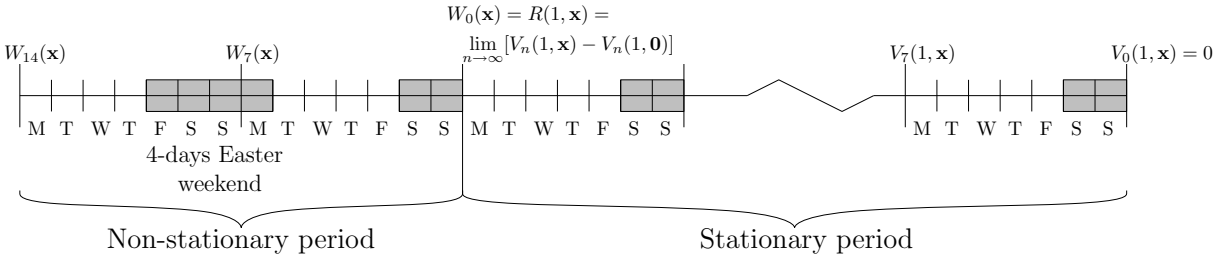


Figure 4.1: The SDP-Simulation approach splits the horizon in two parts, since the system behaves stationary a few days after a break

We model the problem as an infinite horizon problem, that consists of two parts. The first part is the irregular period containing the non-stationary production break(s). The second part relates to the stationary problem. In the example in Figure 4.1, the age distribution of the BPPs in stock on the Tuesday immediately after the break differs from an ordinary Tuesday, the optimal ordering policy may be different than on ordinary Tuesdays. But we may assume that the system returns to the stationary problem a few days after the irregular period has elapsed. We assume that from the next Monday onwards the system behaves indeed stationary.

By stochastic dynamic programming (SDP) the optimal ordering strategy can be computed for both the stationary and the non-stationary problems. We first apply successive approximation to solve the stationary problem and thus obtain the optimal ordering strategy for regular weekdays, Monday to Friday. As a by-product we compute relative values of each stock state that are used as terminal costs to solve in a backward fashion the non-stationary problem. In the next two sections we formalize the approach.

4.2.3 Stationary MDP formulation

The SDP-simulation approach starts with solving the MDP for the stationary problem, since its solution is input to the stochastic dynamic program for the non-stationary problem. The MDP formulation differs only slightly from the one formulated in the previous chapter, as we now deal with only a single demand category, which is met through a FIFO issuing policy.

The stock transition from weekday d to the next day, given that demand is met by a FIFO-issuing policy, follows from the stock transition function $y(\mathbf{x}, k, a)$ and depend on the initial stock $\mathbf{x}$, the demand k, and the production volume a. At the end of the day BPPs that have become outdated are disposed of. The transition probability relate to the demand distribution for weekday d is denoted by $p_d(k)$.

The immediate or direct costs to incur in a slot is now much easier than before, as we consider only a single demand category and no mismatch costs. Let $C(d, \mathbf{x}, k)$ denote the costs to incur in on weekday d, when the demand on that day is k and the initial stock is $\mathbf{x}$. With the total stock level denoted by $x = \sum_{r=1}^{m} x_r$, the direct cost $C(d, \mathbf{x}, k)$ are:

$$C(d, \mathbf{x}, k) = \begin{cases} c^{\mathrm{O}} \cdot (x_m - k)^+ & \text{outdating costs,} \\ +c^{\mathrm{S}} \cdot (k - \sum_{r=1}^{m} x_r)^+ & \text{shortage costs,} \\ +c^{\mathrm{H}} \cdot x & \text{holding costs.} \end{cases} \tag{4.1}$$

Note that in the definition of $C(\cdot)$ we have omitted the parameter I of the issuing policy as in this chapter we consider FIFO issuing only. The expected immediate costs to incur in state $(d, \mathbf{x})$ are simply:

$$\mathbb{E}C(d, \mathbf{x}) = \sum_{k} p_d(k) \cdot C(d, \mathbf{x}, k). \tag{4.2}$$

Using Equation (4.6), the MDP is tractable through a SA algorithm, unless the state space $\mathcal{X}$ is too large.

$$V_n(d,\mathbf{x}) = \min_{a\in\mathcal{A}(d,\mathbf{x})} \left(\mathbb{E}C(d,\mathbf{x}) \; + \; \sum_k p_d(k) \cdot \; V_{n-1}\left(d+1, y(\mathbf{x},k,a)\right) \right). \tag{4.3}$$

We start the SA algorithm by setting $V_0(1,\mathbf{x}) = 0$ for all states $\mathbf{x} \in \mathcal{X}$. The choice the start at a Monday ($d = 1$) is arbitrary, we could as well choose any other weekday to start with. Next $V_1(7,\mathbf{x})$, $V_2(6,\mathbf{x})$,...,$V_7(1,\mathbf{x})$,etc. are computed for all states $\mathbf{x}$.

The span$(\mathbf{V}_n - \mathbf{V}_{n-7})$ is checked every 7 iterations, hence at iteration $n = 7$, 14, 21, etc. Suppose $span(\mathbf{V}_n - \mathbf{V}_{n-7})$ is for the first time smaller than a pre-specified small value ϵ at iteration $N-7$, with N a multiple of 7 days. Optimal actions for each state $\mathbf{x}$ on Monday to Sunday are derived from Equation (4.4) from the last 7 iterations ($n = N-6$, $N-5,\ldots,N$). Hence after N iterations an optimal stationary strategy is approximated by:

$$\pi(d,\mathbf{x}) = \arg\min_{a\in\mathcal{A}(d,\mathbf{x})} \sum_k p_d(k)\, V_{N-d}(d+1, y(\mathbf{x},k,a)). \tag{4.4}$$

In the last 7 iterations the optimal production strategy π is stored. As N is a multiple of 7, the value vector $\mathbf{V}_N$ relates to a Monday. $\mathbf{V}_N$ is a relative value vector that can be used as terminal costs in solving the finite horizon problem of the non-stationary period.

4.2.4 Extension – Including non-stationary periods

In a very similar way the ordering decisions for each day of the non-stationary finite horizon problem are computed. For example, in Figure 4.1, the special period lasts two weeks (14 days) with the 4-days Easter weekend in the middle of the time interval. Note that we have chosen more or less arbitrarily that the finite horizon problem ends a few days after the break on a Monday morning.

For each of the 14 days of the special period the optimal production strategy is determined by Stochastic Dynamic Programming in a backward fashion, using the so-called relative values $R(1,\cdot)$ as terminal costs. The relative values $R(1,\mathbf{x})$ are used to compare stock states $\mathbf{x}$ within the same periodic class, namely the class related to Mondays, the value vector $V_N(1,\mathbf{x}$ can be used for this purpose. Instead of storing the optimal policy for each working day, we now store the policy for all 14 days of the non-stationary period.

The extended SDP-Simulation approach consists of the following five steps:

Step I. Compute relative values of the states for the stationary problem:

- First, the stationary problem is solved (maybe after scaling the problem), using Equations (4.6) and (4.4).
- Next, we choose an arbitrary reference state on Monday, say $(1, \mathbf{0})$, and compute for every possible stock state $\mathbf{x}$ on Monday the difference in expected future costs relative to this reference state:

$$R(1, \mathbf{x}) = \lim_{n \to \infty} [\, V_n(1, \mathbf{x}) - V_n(1, \mathbf{0}) \,] \approx V_N(1, \mathbf{x}) - V_N(1, \mathbf{0}) \tag{4.5}$$

These differences, $R(1, \cdot)$, are feasible relative values, when N is sufficiently large. $V_N(1, \mathbf{x})$ is finite when N is finite. Therefore setting $R(1, \mathbf{x}) = V_N(1, \mathbf{x})$ is also feasible.

Step II. Solving the non-stationary problem by SDP:

Let the irregular period last T days, with the last day being a Sunday. The days of this period are numbered backwards and denoted by t. $t = 1$ thus refers to the last day of the period (a Sunday) and day T is the first day of the finite horizon. Index t thus denotes the number of days to go until the end of the irregular period. In addition to the notation in Equations (4.6) and 4.4 we define:

$p_t^{\text{irr}}(k)$ = the probability of a (composite) demand k on day t.
If the demand remains stationary even during a break, then $p_t^{\text{irr}}(k)$ equals $p_d(k)$ for $d = 7 - (t - 1) \mod 7$.

$\mathcal{A}'_t(\mathbf{x})$ = action space at day t as bounded by the (artificial) production and storage capacity. Clearly the action space depends on the stock state $\mathbf{x}$. If production is not possible on day t, $\mathcal{A}'_t(\mathbf{x}) = \{0\}$.

$W_t(\mathbf{x})$ = the total expected costs under an optimal strategy from day t onwards when starting in inventory state $\mathbf{x}$ and at the end of the irregular period terminal cost are accounted.

$W_0(\mathbf{x}) \equiv R(1, \mathbf{x})$.

$\pi_t(\mathbf{x})$ = the optimal decision at day t given the inventory state is $\mathbf{x}$.

By stochastic dynamic programming one recursively computes and stores successively for $t = 1, 2, \ldots, T$, for all states $\mathbf{x}$ in the state space $\mathcal{X}'(t)$:

$$W_t(\mathbf{x}) = \min_{a \in \mathcal{A}'_t(\mathbf{x})} \left(\mathbb{E}C_t(\mathbf{x}) \; + \; \sum_k p_d(k) \cdot \; W_{t-1}(y(\mathbf{x}, k, a)) \right). \tag{4.6}$$

with $\mathbb{E}C_t(\mathbf{x}) = \sum_k p_t^{\text{irr}}(k) \cdot C(d, \mathbf{x}, k)$, in which d is the weekday related to t.

The optimal ordering quantity on day t follows from

$$\pi_t(\mathbf{x}) = \arg \min_{a \in \mathcal{A}'_t(\mathbf{x})} \sum_k p_t^{\text{irr}}(k) W_{t-1}(y(\mathbf{x}, k, a)). \tag{4.7}$$

Step III. Read simple rule from simulation-based frequency table
Again the optimal strategy may be fairly complex. Hence simulation is used to investigate the structure of the optimal strategy for each day of the special period. That is: frequency tables for each day t of the irregular period are generated and, if applicable, an order-up-to S rule is read for each day of the week or any other appropriate ordering rule.

Step IV. Finally, by a detailed simulation program the rule is put to the test and its performance, in terms of outdating and shortage figures, is compared to the figures for the optimal stock-age-dependent strategy.

Step V. In case the initial problem is scaled the simple rule is simulated for the full-size problem after re-scaling the parameters of the simple rule.

4.3 Case study – optimal policy around breaks

In this section we apply the extended SDP-Simulation approach using realistic data, as summarized in Section 4.3.1. The results for the problem around Christmas and New Year's Day are presented in Section 4.3.3. In section 4.3.4 we discuss the 4-days Easter weekend. The results are integrated in Section 4.3.5. We will show that a simple rule applies even around breaks, although we do not provide a detailed sensitivity study as we have done in the previous chapter. We limit the illustration to the first four steps of the SDP-simulation approach.

4.3.1 Data

The case study is constructed by modifying the data for one of the four Dutch Blood banks (Sanquin, division North-East). The demand is aggregated into a single category that will receive the oldest BPPs in stock (FIFO issuing). The demand for 144 BPPs a week is spread over Monday to Sunday as in Table 4.1. To make the MDP tractable we scale the problem by a factor 4, resulting in the mean demand figures in the last row of Table 4.1.

Table 4.1: Demand distributions: means and coefficients of variation (cv).

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
original mean	26	21	32	21	26	8	10
cv	0.28	0.31	0.25	0.31	0.28	0.50	0.45
scaled mean	6.5	5.25	8	5.25	6.5	2	2.5

The coefficient of variation (cv) of the demand is set to 1.4 times the cv when demand would be Poisson distributed. In fact, this assumption corresponds to the worst case of Section 3.7.2, where one demand category, comprising two thirds of the average demand, is Poisson distributed, but the cv of the remaining third of the demand is twice as high. The cv over all demand is then $\sqrt{2} \approx 1.4$ times[1] the cv for the Poisson case.

[1]Consider three stochastic variables X, X_1 and X_2. Let $X = X_1 + X_2$, with mean μ. X_1 is Poisson($\mu_1 = \frac{2}{3}\mu$) distributed, X_2 has mean $\mu_2 = \frac{1}{3}\mu$ and coefficient of variation $cv_2 = 2 \cdot \frac{1}{\sqrt{\mu_2}}$. Hence variance of X is $(cv_1 \cdot \mu_1)^2 + (cv_2 \cdot \mu_2)^2 = \mu_1 + 4\mu_2 = \frac{2}{3}\mu + \frac{4}{3}\mu = 2\mu$. Hence the coefficient of variation of X is $\sqrt{2}/\mu = \sqrt{2} \cdot cv(\text{Poisson}(\mu))$.

For each day of the week d we fit a discrete probability distribution $p_d(\cdot)$ on the mean demand and the reported cv. Conform [2] the unscaled demand distributions are mixtures of two negative binomial distributions. For the scaled problem, the demand distributions are fitted using a mixture of two binomial distributions.

The other problem data for the unscaled case remain unchanged:

Annual demand	7,488 BPPs or 1,872 batches
Costs	outdating 150 per outdated BPP, shortage costs 750 per BPP short, all other costs are zero,
Production	Monday – Friday, but no during breaks
Maximal shelf life	$m = 5$ days.

For a high accuracy performances statistics will be obtained from 100 detailed simulation runs of 1,000,000 weeks each.

4.3.2 Step I – Stationary problem

After scaling the MDP and solving the scaled problem, we simulate the resulting optimal strategy. Table 4.2 presents the simulation-based frequency tables for Monday to Friday. In the first column we read the order-up-to levels related to the optimal actions. From the last column we read which level is most frequent. On Mondays the most-frequent order-up-to level is 21 and it fits to 59% of the 1 million states visited. Similarly, we find most-frequent order-up-to levels $(21, 21, 21, 19, 24)$ batches (of 4 BPPs) for Monday to Friday.

In Table 4.3 we compare the order-up-to rule with order-up-to levels fixed to $(21, 21, 21, 19, 24)$ against the optimal (age-dependent) strategy. The upper half of the table shows the characteristics of the optimal MDP strategy, the lower half shows the performance of a fixed replenishment rule. As expected from the results and the discussion in the previous chapter, the order-up-to S rule appears to perform nearly optimal. The absolute outdating and shortage figures per week should be related to a weekly demand of 36 batches of 4 pools. The annual results relate to an average annual demand for 1872 batches.

Table 4.2: Simulation frequency tables of scaled MDP strategy with FIFO demand only.

(a) (State, action)-frequency tables for 1,000,000 simulated Mondays.

Stock x	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	Freq(S_1)
Up-to S_1																				
24													4	6	12	153	101	250	27	553
23												4	46	2906	17596	5508	1262			27322
22										5	1156	121403	79791	39802						242157
21								116497	152240	167708	154513									590958
20						39450	74336													113786
19					17209															17209
18				5976																5976
17			1657																	1657
16		332																		332
15	50																			50
14																				0
:																				:
0																				0
Freq(x)	50	332	1657	5976	17209	39450	74336	116497	152240	167713	155669	121407	79841	42714	17608	5661	1363	250	27	1000000

(b) (State, action)-frequency tables for 1,000,000 simulated Tuesdays.

Stock x	0 . . . 7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	Freq(S_2)
Up-to S_2																		
26														1				1
25														1	1	3		5
24												2	6	42	33	22	1	106
23											5	277	622	898	247	16		2065
22									40	653	5055	15270	10982	2259	215			34474
21			786	9588	44321	83285	143525	191312	198489	159515	95150	32597	4316	427				963311
20		38																38
19																		0
:																		:
0																		0
Freq(x)	0	38	786	9588	44321	83285	143525	191312	198529	160168	100210	48146	15926	3628	496	41	1	1000000

(c) (State, action)-frequency tables for 1,000,000 simulated Wednesdays.

Stock x	0 . . . 9	10	11	12	13	14	15	16	17	18	19	20	21	22	Freq(S_3)
Up-to S_3															
25													3		3
24									2		72	103	87	2	266
23								9	173	1314	4637	1865	592		8590
22						9	988	6708	49510	99883	31271	5334			193703
21				16565	59327	136626	214317	231213	136611						794659
20		167	2612												2779
19															0
:															:
0															0
Freq(x)	0	167	2612	16565	59327	136635	215305	237930	186296	101197	35980	7302	682	2	1000000

(d) (State, action)-frequency tables for 1,000,000 simulated Thursdays

Stock x	0 . . . 4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	Freq(S_4)
Up-to S_4																			
22																	3	6	9
21															106	753	204		1063
20											1156	7482	44639	30494	8368	931			93070
19					7114	24181	58816	110748	164500	196650	184235	129167	28890						904301
18			145	1401															1546
17		7																	7
16																			0
:																			:
1																			0
0																		4	4
Freq(x)	0	7	145	1401	7114	24181	58816	110748	164500	196650	185391	136649	73529	30494	8474	1684	207	10	1000000

(e) (State, action)-frequency tables for 1,000,000 simulated Fridays.

Stock x	0 . . . 7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	Freq(S_5)
Up-to S_5																
28														1	2	3
27														2		2
26											30	262	223	70		585
25							1	378	2670	17755	20380	9471	1110			51765
24		141	2406	15686	57196	132318	210586	235857	185682	87884	19889					947645
23																0
:																:
0																0
Freq(x)	0	141	2406	15686	57196	132318	210587	236235	188352	105639	40299	9733	1333	73	2	1000000

Table 4.3: Order-up-to S rule vs MDP policy for a regular (stationary) week (statistics in batches of 4 pools obtained by simulation for 100 million weeks).

Weekday	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Weekly	Annual
MDP policy									
# batches outdated	0	0.01	0.06	$-^a$	$-^a$	0	0.01	0.08	4.30 (0.23%)
# batches short	0.01	0	0	0	0	0	0	0.01	0.71 (0.04%)
Annual costs									4,698 euro
Order-up-to S rule									
Levels S_d	21	21	21	19	24	–	–		
Goodness-of-fit	59%	96%	79%	90%	95%	–	–		
# batches outdated	0	0.01	0.06	$-^a$	$-^a$	0	0.01	0.08	4.18 (0.22%)
# batches short	0.01	0	0	0	0	0	0	0.01	0.74 (0.04%)
Annual costs									4,724 euro

[a]Outdating must be zero on Thursday and Friday, as $m = 5$ and production stops in the weekends.

From the scaled MDP results we conclude that

- Compared to the current practice, it seems that the annual outdating can be reduced from about 15% to 0.2%, even when demand is more stochastic than Poisson.
- Shortages occur virtually never: only 0.04% of the total annual demand cannot be met immediately from stock.
- The order-up-to S policy as read from the frequency tables closely approximates the structure of the optimal strategy and is only 0.6% off from the optimal cost level.

Remark For the unscaled case, the replenishment levels are re-scaled by multiplication with a factor 4. In the previous chapter we have already shown that the resulting order-up-to levels are nearly optimal. More results for cases where all demand is aggregated into a single category are found in [56], [57], [58], [149], [77].

4.3.3 Steps II to IV – Christmas and New Year's Day

For example, we consider a year in which Christmas falls on Tuesday and Wednesday, and New Year's Day (NYD) on the next Tuesday. On these days (regular) production is stopped because of a lack of donors. As depicted on the timeline in Figure 4.2 there is only a single day between the weekend and the two Christmas holidays. On this Monday one has to produce additional BPPs to anticipate the production stop for the next two days. In this section we consider a worst case where demand for platelet pools continues as usual.

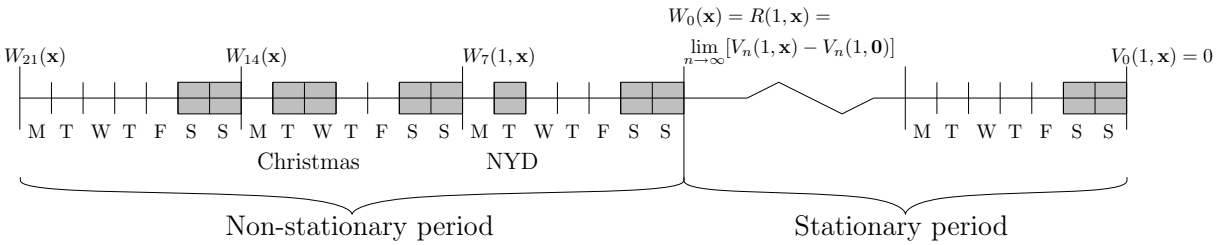


Figure 4.2: The SDP-Simulation approach splits the horizon into two parts, as if the system behaves stationary a few days after a break

What to expect?

When the production capacity on Monday is not restrictive, as in our case, then the production on Monday has to be set high enough to anticipate the production stop on the next two days. When the production capacity on Monday is normally high enough but too restrictive to anticipate the two-day production stop, then the production volume on Friday is raised as well. When no holding costs and quality mismatch costs apply, the capacity restrictions will hardly affect the cost-level. Compared to the stationary case the optimal cost level does increase, since outdating and shortages are likely more prevalent. Shortages mostly occur on the Thursday after Christmas. Excessive production on the Monday before Christmas outdates the next Saturday.

We assume that the production and storage capacity are not restrictive. (Artificial bounds, which are set to make the state space finite, are set high enough to ensure that the optimal policy is not affected.) The length of the non-stationary period can then be reduced from 3 weeks (as in Figure 4.2) to 2 weeks, as the problem in the first week maybe still stationary. When the production capacity on Wednesdays, January 2^{nd}, is not restrictive then the production strategy on Thursday January 3^{rd}, may differ only slightly from that on an ordinary Thursday. Moreover, given $m = 5$, no BPPs will expire on Thursday, as no BPPs in stock on Thursdays are older than three days.

Results for optimal strategy

Table 4.4 shows the impact of the irregular production breaks on the optimal strategy over a ten-days period from Monday (December, 24^{th}) to Wednesday (January, 2^{nd}). The results are presented in batches of 4 pools, since the MDP is scaled to reduce the state space. The average demand over the 10-days period is almost 56 batches (223 pools).

Table 4.4: Impact of production breaks around Christmas and New Year's Day.

(a) Impact of breaks on outdating and shortages (in batches) under MDP policy.

10-days period	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Total
Stationary case											
# batches outdated	0	0.01	0.06	$-^a$	$-^a$	0	0.01	0	0.01	0.06	0.15
# batches short	0.01	0	0	0	0	0	0	0.01	0	0	0.03
Irregular breaks		Dec-25	Dec-26						Jan-1		
# batches outdated	0	0.01	0.11	$-^a$	$-^a$	0.59	0	$-^a$	$-^a$	0.05	0.77
# batches short	0	0	0	0.06	0	0	0	0.01	0	0.01	0.08

[a]No outdating $m = 5$ days after weekends and holidays, since production is zero.

(b) Impact of breaks on order-up-to levels S_t as read from the optimal MDP policy.

10-days period	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed
Stationary case										
Order-up-to S_t	21	21	21	19	24	–	–	21	21	21
Goodness-of-fit	59%	96%	79%	90%	95%	–	–	59%	96%	79%
Irregular breaks		Dec-25	Dec-26						Jan-1	
Order-up-to S_t	31	–	–	20	24	–	–	27	–	21
Goodness-of-fit	98%	–	–	49%	44%	–	–	93%	–	48%

Table 4.4(a) shows that outdating for this ten-days period has increased from 0.15 batches (0.6 pools) to 0.77 batches (3.1 pools). Expected relative outdating thus is about 1.4%. Since the results relate to the optimal strategy one has to accept this increase, which is primarily due to the outdating of pools produced on the Monday before Christmas. Although not reported in the table, we have observed that only 0.55% of the BPPs (0.08 batch) produced on the Monday before News Year's Day becomes outdated on the next Saturday.

Order-up-to rule vs Optimal strategy

Since in practice one prefers a simple rule, we hope that an order-up-to S rule resembles the structure of the optimal policy. By simulation we generate (state, action)-frequency tables for each day of the special period in a similar way as for the stationary case. In Table 4.4(b) we report the most-frequent order-up-to levels for the ten-days period and compare them against the stationary ones. As expected the order-up-to levels on the Mondays before the two breaks are considerably higher than in the stationary case: e.g. 31 vs 21 before Christmas. Remarkably, an order-up-to S rule fits even better on Mondays just before a break, than on a regular Monday: 98% before Christmas versus 59% on the regular Mondays. On the days after a break the order-up-to S rule fits to almost 50% of the states visited, whereas in the stationary case this figure falls in 79%-95%.

We evaluate the order-up-to S rule with the most-frequent order-up-to levels from the

Table 4.5: Order-up-to S vs MDP policy around Christmas and New Year's Day.

10-days period	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Total
Irregular breaks		Dec-25	Dec-26						Jan-1		
MDP policy											
# batches outdated	0	0.01	0.11	$-^a$	$-^a$	0.59	$-^a$	$-^a$	0	0.05	0.77
# batches short	0	0	0	0.06	0	0	0	0.01	0	0.01	0.08
Order-up-to S rule											
Levels S_t	31	–	–	20	24	–	–	27	–	21	
# batches outdated	0	0.01	0.11	$-^a$	$-^a$	0.59	$-^a$	$-^a$	0.01	0.04	0.75
# batches short	0	0	0	0.06	0	0	0	0.02	0	0.01	0.10

[a]No outdating $m = 5$ days after weekends and holidays, since production is zero.

frequency tables by a long simulation (100 million replications) and compare its performance against the optimal strategy. (The results could be computed in an exact way by solving the underlying Markov chains under modified cost structures.) Although the order-up-to S rule does not fit for 100% at each day, its performance is very close to optimal as reported in Table 4.5. Over the given ten-days period on average only 0.1 batch (out of the 56 batches demanded) cannot be met from stock: the shortage rate is thus less than 0.2%. Outdating is in the order of 1.3%.

Re-scaling After re-scaling the order-up-to levels, one obtains a nearly optimal order-up-to S rule for the real-sized case. Given the robustness of the rule with respect to minor changes in the order-up-to levels as shown in the previous chapter we skip step V of our extended SDP-Simulation approach.

Conclusions – Results breaks during Christmas and New Year's Day

It appears that:

- an order-up-to S policy with increased order-up-to levels for the Monday before Christmas and the Monday before New Year's Day is nearly optimal,
- age plays an more important role after Christmas and New Year's Day, but the absolute impact on outdating and shortages is relatively small, and
- outdating over a ten-days period, including the breaks, is as low as 1.4%, while shortages are less than 0.2%.

4.3.4 Steps II to IV – Four-days Easter weekend

The second example of a non-stationary problem with irregular production breaks is the 4-days Easter weekend, as depicted before in Figure 4.1. The long weekend from Good Friday to Easter Monday is considerably more difficult, since the production is stopped for four consecutive days, while the shelf life is only 5 days and demand remains stationary. We assume that the production and storage capacity are not restrictive.

What to expect?

Again, we expect that primarily the production volumes one day before the break, in this case on the Thursday before Good Friday, are increased dramatically to anticipate the production stops for the next four days. The available stock plus the production on Thursday should be enough to survive until the next Wednesday morning when new stock is released. The order-up-to level on Thursday before the break can be derived by the Newsboy model, since no BPPs will survive until Wednesday.

The marginal return of ordering z batches instead of $z-1$ batches is the savings on shortages. The marginal costs are an increase in the expected outdating costs. Let the stochastic variable Z denote the demand in batches over the six-days period from Thursday to Tuesday, and $P(\cdot)$ is the cumulative distribution of Z. $P(\cdot)$ is obtained from the convolution of the six demand distributions. The optimal order-up-to level on Thursday is the greatest value of z for which the expected marginal savings on shortage costs is still larger than the expected marginal outdating costs, as reflected in Equation (4.8).

$$c^{\mathrm{S}} \cdot P(Z \geq z) \;\geq\; c^{\mathrm{O}} \cdot P(Z \leq z-1) \tag{4.8}$$

Hence the best order-up-to level is the greatest z for which holds:

$$P(Z \leq z-1) \;\leq\; \frac{c^{\mathrm{S}}}{c^{\mathrm{S}} + c^{\mathrm{O}}}. \tag{4.9}$$

For the given demand distributions and the cost figures, the order-up-to level on the Thursday before the break is according to the Newsboy equation 31 batches. The production volume just before the break is thus likely much higher than on a regular Thursday. Consequently, outdating mostly happens five days later on Tuesday, just after the break.

Since production stops from Friday to Monday, shortages primarily happen on Tuesday given that the production lead time is one day. We expect shortages and outdating to be far more prevalent than over the Christmas period.

On Tuesday morning after Easter Monday all products in stock, if any, are of the same age. The optimal production strategy is thus not stock-age-dependent. No BPPs produced before the break will survive until Wednesday morning, hence one may expect that the production volume on Tuesday is fixed to a target inventory level on Wednesday morning. Consequently, the optimal production volume on Wednesday is also fixed, as Tuesdays production becomes available only at the start of Wednesday morning and all products in stock are of the same age. From Thursday onwards the optimal production strategy is again stock-age-dependent.

Results of optimal strategy

After scaling and solving the MDP, we can check our expectations. Through simulation we generate frequency tables from which we read the structure of the optimal strategy. A selection of them is found in Table 4.6. As expected and argued before, according to the upward diagonal in Table 4.6(a), a fixed production volume applies on the Tuesday just after Easter Monday. All batches produced before the break will not survive until the Wednesday morning after Easter.

Since all stock present on Tuesday morning after Easter Monday will perish the same day, the initial stock on Wednesday morning consists only of the 15 batches produced on Tuesday. From Table 4.6(a) we observe that a fixed order-up-to level of 21 batches applies on Wednesday, which implies a fixed production volume of 6 batches. Note that an order-up-to level of 21 batches corresponds to the stationary order-up-to level reported in Table 4.4(b).

The order-up-to level on Thursday after the break equals 19, which corresponds to the stationary order-up-to level. This illustrates that the stationary order-up-to levels apply from Wednesday onwards. Although not reported the outdating and shortages figures from the Thursday after the break onwards do not differ significantly from those on regular days. One positive exception, not visible in the table, is that outdating on the Sunday after the break is significantly lower than usual.

The optimal production policy on Thursday prior to Good Friday resembles for virtually 100% an order-up-to S rule with fixed order-up-to level 32. This is inline with our expectations: the newsboy model suggest an order-up-to level of 31 batches. As the initial stock

Table 4.6: Frequency tables from 1 million simulations of MDP policy around Easter.

(a) (State, action)-frequency table for a Tuesday after Easter Monday.

Stock x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	Freq(S)
Up-to S																							
36																						1	1
35																					20		20
34																				176			176
33																			933				933
32																		3578					3578
31																	10375						10375
30																23749							23749
29															43775								43775
28														68301									68301
27													91544										91544
26												110320											110320
25											116201												116201
24										118217													118217
23									110287														110287
22								94938															94938
21							74669																74669
20						53775																	53775
19					35596																		35596
18				21466																			21466
17			11893																				11893
16		5758																					5758
15	4428																						4428
14																							0
⋮																							⋮
0																							0
Freq(x)	4428	5758	11893	21466	35596	53775	74669	94938	110287	118217	116201	110320	91544	68301	43775	23749	10375	3578	933	176	20	1	1000000

(b) (State, action)-frequency table for a Wednesday after Easter Monday.

Stock x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	Freq(S)
Up-to S																							
21																1000000							1000000
20																							0
⋮																							⋮
0																							0
Freq(x)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1000000

(c) (State, action)-frequency table for a Thursday after Easter Monday.

Stock x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	Freq(S)
Up-to S																							
21																					245		245
20																		27771	8545	1830			38146
19								8547	27770	66651	122191	174559	196384	174561	122189	66652							959504
18							260	1830															2090
17																							0
⋮																							⋮
1																						15	0
0																							15
Freq(x)	0	0	0	0	0	0	260	1830	8547	27770	66651	122191	174559	196384	174561	122189	66652	27771	8545	1830	245	15	1000000

may not survive till the next ordering moment, the best order-up-to level is 1 higher. The order-up-to level 32 is 13 batches higher than on a regular Thursday (see Section 4.3.2).

Just as for the Christmas and New Year's Day period, on the day before a production break an order-up-to S rule (with increased levels) fits even better than in the stationary case. Apparently the age-distribution of the stock is less relevant on a day prior to a production break.

Order-up-to rule vs Optimal strategy

The structure of the optimal policy seems thus, again, to be very well presented by an order-up-to S rule. In Table 4.7 we report over a ten-days period, including the Easter weekend, the outdating and shortage volumes around Easter under both the optimal production policies and the order-up-to S rule. In the table Good Friday and Easter Monday are abbreviated by GF respectively EM.

In the last column of Table 4.7 we observe that the order-up-to S rule performs almost equally well as the optimal MDP policy. As expected, outdating and shortages happen mostly on the Tuesday after Easter Monday. Under both strategies on average approximately 4.3 batches will outdate and stock falls on average 0.28 batches short. The average demand over the ten-days period is on average almost 56 batches and the average production to cover the demand over this period is roughly 60 batches. Relative outdating over the ten-days period is thus $\frac{4.3}{60} = 7.2\%$. The shortage rate over the ten-days period is $\frac{0.28}{56} = 0.5\%$.

Table 4.7: Order-up-to S rule vs MDP policy around Good Friday and Easter Monday.

10-days period Irregular breaks	Mon	Tue	Wed	Thu	Fri GF	Sat	Sun	Mon EM	Tue	Wed	Total
MDP policy											
# batches outdated	0	0.01	0.06	$-^a$	$-^a$	0	0.03	0.07	4.16	$-^a$	4.34
# batches short	0.01	0	0	0	0	0	0	0	0.26	0	0.28
Order-up-to S rule											
Levels S_t	23	22	24	32	–	–	–	–	15^b	21	
Goodness-of-fit	44%	75%	64%	100%	–	–	–	–	$100\%^b$	100%	
# batches outdated	0	0.01	0.06	$-^a$	$-^a$	0	0.03	0.04	4.19	$-^a$	4.33
# batches short	0.01	0	0	0	0	0	0	0	0.26	0	0.28

[a] No outdating 5(= m) days after weekends and holidays, since production is zero.
[b] A fixed production volume applies since all batches in stock are outdated at the end of the day.

Conclusions – Results four-day Easter weekend

The major conclusions over a ten-days period that includes the four-days Easter weekend are:

- A simple replenishment rule performs nearly optimal and results in:
  - an outdating figure of only 7.2%,
  - shortages to be less than 1%.
- Outdating and shortages happen mostly on the Tuesday after Easter Monday, indicating the difficulty in anticipating a four-days production stop when the maximal shelf life is only 5 days.
- The production levels on the Thursday before and the Tuesday after the weekend are considerably higher than in the stationary case.
  - On the Tuesday after Easter Monday a fixed production quantity applies, since pools produced before Good Friday will not survive until Wednesday.
  - Shortly after the break (from Wednesday onwards) the stationary order-up-to S rule resumes as a very good approximation of the optimal MDP policy.

4.3.5 Conclusions – Extended SDP-Simulation Approach

Extended SDP-Simulation Approach

The SDP-Simulation approach can be applied to solve the order problem of perishables with a (short) fixed shelf life. We have extended the approach such that it can deal with (non-stationary) production breaks. It appears to be a powerful approach for deriving an optimal stock-age-dependent (scaled) policy and for investigating the structure such that a practical rule can be derived from it.

Results over a year including Easter and Christmas

We have tested the approach on a PPP using realistic data for a single category of demand. For the PPP under consideration even around breaks simple order-up-to S rules apply and optimal order-up-to levels are easily read from simulation-based frequency tables.

By combining the results from the previous sections, the following conclusions can be drawn concerning the performance of the SDP-Simulation approach over a year which includes the breaks during Christmas, New Year's Day and the 4-day Easter weekend:

- Average annual shortage = 1.1 batch = 4.2 pools $< 0.1\%$
- Average annual outdating = 9.0 batches = 36 pools $< 1\%$

Compared to the current practice the potential savings are substantial: it seems that the current outdating figure of 15-20% can be reduced to less than 1%, while shortages arise only a few times per year.

4.4 The hospital case with no order costs

The SDP-Simulation approach provides insights into the structure of the optimal strategy. It shows whether an order-up-to S is appropriate, and if so, it suggests nearly optimal order-up-to levels for each weekday. The 'relative uncertainty' in the demand, as measured by the coefficient of variation, depends on the scale at which the problem plays. Therefore we may expect that an order-up-to S rule fits less well to an production-inventory problem that plays at a much smaller scale: i.e. at a much smaller blood bank or at a medium-large hospital.

Then fixed set-up costs per production run or fixed order costs may apply, as we will discuss in the next section. In this section we leave any fixed order costs out of the model, and focus on the impact of the demand uncertainty on the structure of the optimal policy, and the resulting outdating and shortage figures.

4.4.1 Problem and data

Thus far we have studied the BPP inventory problem using data of one of the four Dutch blood banks. Since other (foreign) blood banks might operate at a smaller scale, and surely do most hospitals do, we now consider the order problem at a much smaller scale. We refer to the problem in this section as the *Hospital case*, although we do not change the modeling of the order problem. Again, we assume a fixed lead time of one day: regular replenishment orders are placed early in the morning and these do arrive at the end of the day, say early the next morning.

We investigate the optimal stationary ordering policy for a hospital with an average demand that is 10 times as low as that for the blood bank considered in the previous section. The average weekly demand is thus 14.4 pools. Emergency orders to resolve shortages are delivered from the blood banks more or less instantaneously. The costs of emergency deliveries to resolve any shortages are fixed to 750 per BPP.

Again, we assume that the demand is Poisson distributed. On Mondays the mean demand is 2.6 pools and the coefficient of variation (cv) is thus $1/\sqrt{2.6} \approx 0.62$. Table 4.8 summarizes the daily demand characteristics. Since demands happens occasionally during weekends, the cv on some days can well exceed 1, but is on most days around 0.65. The cv of the demand over an entire week is $1/\sqrt{14.4} \approx 0.26$.

Since demands are Poisson distributed with low means, the coefficients of variation are very high. Therefore we expect an increase in the relative outdating and shortage figures

Table 4.8: Poisson demand distributions: means and coefficients of variation (cv).

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
original mean	2.6	2.1	3.2	2.1	2.6	0.8	1.0
cv	0.62	0.69	0.56	0.69	0.62	1.12	1.00

and the optimal policy to be 'more' stock-age-dependent.

- An open question to answer in the next sections is whether a simple order-up-to S rule is still close to optimal for such a small-scale problem.

Given that BPPs have to be transported from a blood bank to the hospital, daily ordering by the hospital may be inefficient. Instead of daily ordering, one may restrict the number of decision epochs to say Mondays, Wednesdays, and Fridays. An interesting question is thus:

- How does the optimal ordering strategy look like, when orders can be placed only on Mondays, Wednesdays and Fridays.

Hence additional production breaks are introduced on Tuesdays and Thursdays. Alternatively one may incur fixed order costs to reflect the costs involved with the transport of the pools from the blood bank to the hospital.

In Table 4.9, an overview is given of the several studies for the 'Hospital case'. Although, in The Netherlands, these costs are not (directly) billed to the hospital, the effect of fixed order costs is studied in Section 4.5.

Table 4.9: Overview of study of small scale problem.

Hospital case or Small blood bank 'FIFO-demand' only : 14.4 BPPs per week			
No fixed order costs ($c^{\mathrm{F}} = 0$)		With fixed order costs c^{F}	
Order daily (Mon to Fri) Section 4.4.2	Order only on Mon, Wed, Fri Section 4.4.3	low $c^{\mathrm{F}} = 75$ Section 4.5.2	high $c^{\mathrm{F}} = 150$ or 500 Section 4.5.3

4.4.2 Daily ordering

The formulation of the MDP is similar to that in Section 4.2.3. Since this problem plays at a small scale direct computation of an optimal strategy is possible without scaling the problem. Still we investigate the structure of the resulting optimal strategy in search for a practical rule. From simulation-based frequency tables (see Table 4.10) we read an ordinary order-up-to S rule with order-up-to levels $(11, 10, 11, 9, 11)$ for Monday to Friday. The order-up-to S rule fits to 47-66% of the states visited during the simulation. The performance of the rule and the optimal MDP strategy is evaluated by a long simulation for 100 million weeks. The main results and conclusions are:

1. Even under the optimal MDP policy, the relative outdating and shortage figures for the hospital case are much higher than those reported for the blood bank (in Table 4.3), due to the high coefficient of variation of the demand and the small scale at which the hospital problem plays:
   - outdating = 8.5%,
   - shortages = 1%.
2. An order-up-to S rule rule appears to perform nearly optimal with slightly more shortages:
   - outdating = 8.5%,
   - shortages = 1.3%,
   - The order-up-to S rule shows a 10% increase in costs compared to the cost-optimal MDP policy:
     - weekly shortage and outdating costs under MDP: 303,
     - weekly costs of order-up-to S rule: 337

     .

For now, we conclude that, even for the small scale problem under consideration, the order-up-to S rule performs quite well.

Table 4.10: Frequency tables from 1 million simulations of MDP policy with daily ordering independent of the fixed order costs.

(a) (State, action)-frequency table for Monday.

Stock x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	Freq(S_1)
Up-to S_1															
14												55	121		176
13										194	6543	4242			10979
12								226	44582	104321	22168				171297
11							31536	252641	185848						470025
10					59645	101872	134222								295739
9			13301	30744											44045
8		5276													5276
7	2463														2463
6															0
:															:
0															0
Freq(x)	2463	5276	13301	30744	59645	101872	165758	252867	230430	104515	28711	4297	121	0	1000000

(b) (State, action)-frequency table for Tuesday.

Stock x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	Freq(S_2)
Up-to S_2															
15														11	11
14												113	413	10	536
13											260	2744	1256		4260
12										204	10805	17370			28379
11								25	7719	184917	108229				300890
10					13789	39903	103890	167811	217563	119774					662730
9			8	1131											1139
8															0
:															:
1															0
0												1764	291		2055
Freq(x)	0	0	8	1131	13789	39903	103890	167836	225282	304895	119294	21991	1960	21	1000000

(c) (State, action)-frequency table for Wednesday.

Stock x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	Freq(S_3)
Up-to S_3															
14												4	15		19
13											599	3001			3600
12										21291	45764	829			67884
11								249	127232	323870	45411				496762
10						843	79943	156663	148673						386122
9				2819	10810	31400									45029
8			538												538
7		46													46
6															0
:															:
0															0
Freq(x)	0	46	538	2819	10810	32243	79943	156912	275905	345161	91774	3834	15	0	1000000

(d) (State, action)-frequency table for Thursday.

Stock x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	Freq(S_4)
Up-to S_4															
12												379			379
11										3378	18643				22021
10								8315	263097	77619					349031
9				17646	48379	86116	150772	319554							622467
8			4613												4613
7		76													76
6															0
:															:
1															0
0												1413			1413
Freq(x)	0	76	4613	17646	48379	86116	150772	327869	263097	80997	18643	1792	0	0	1000000

(e) (State, action)-frequency table for Friday.

Stock x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	Freq(S_5)
Up-to S_5															
16													48		48
15												584			584
14											2305	2197			4502
13									283	20966	43832	265			65346
12							67	17357	246413	150981	2899				417717
11					30246	78344	158916	220214	12319						500039
10			2382	9345											11727
9		37													37
8															0
:															:
0															0
Freq(x)	0	37	2382	9345	30246	78344	158983	237571	259015	171947	49036	3046	48	0	1000000

4.4.3 Order only on Mondays, Wednesdays and Fridays

When orders can be placed only on Monday, Wednesday and Friday, one can expect to order more at a time to anticipate the additional stationary ordering breaks on Tuesday and Thursday. Consequently one expect to observe an increase in both outdating and shortages. The frequency tables in Table 4.11 show the structure of the optimal strategy. Again a simple order-up-to S rule seems to fit reasonably well to the optimal MDP policy, with goodness-of-fit percentages close to 50%.

Both the optimal MDP strategy and the order-up-to S rules are simulated for 100 million weeks. In Table 4.12(a) and Table 4.12(b) we compare daily ordering on Monday to Friday against ordering on Monday, Wednesday and Thursday only.

We observe:

- Under the optimal MDP policy with ordering on Monday, Wednesday and Friday only the outdating and shortages rates are respectively 11.7% and 1.2%. Compared to daily ordering the additional breaks on Tuesday and Thursday thus have significant negative effect on the outdating of pools.

- Under the order-up-to S rule, with order-up-to levels $(13, 0, 12, 0, 12)$ for Monday to Friday, outdating and shortages are respectively 11.7% and 2.0%. An order-up-to S rule yields thus only 0.8% more shortages than the optimal stock-age-dependent strategy, but at the same amount of outdated BPPs.

- The resulting cost level of the order-up-to S rule is about $\frac{500-416}{416} = 20\%$ above the cost level of the optimal MDP strategy with ordering on Mondays, Wednesdays, and Fridays only.

The main conclusion is that the costs under daily ordering are much lower than when ordering on Tuesday and Thursday is prohibited: when applying an order-up-to S rule the gap is about 163 euros per week. When ordering would costs 75 euros per order, then daily ordering is $163 - 2 \cdot 75 = 13$ euros per week cheaper. But when the fixed order costs are twice as high, 150 euro per order, then not ordering on Tuesdays and Thursdays implies a cost saving of $163 - 2 \cdot 150 = 137$ compared to daily ordering.

Table 4.11: Frequency tables from 1 million simulations of MDP policy with ordering every Monday, Wednesday and Friday independent of the fixed order costs.

(a) (State, action)-frequency table for Monday.

Stock x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	Freq(S_1)
Up-to S_1															
18												44	78		122
17											1624	1984	258		3866
16										4860	8519	2065	426		15870
15									5845	31922	31769	5329			74865
14								4125	162516	114021	5688				286350
13							16628	153762	224439	80156					474985
12					18923	43547	71030								133500
11		2348	7184												9532
10	910														910
9															0
:															:
0															0
Freq(x)	910	2348	7184	18923	43547	87658	153762	228564	248517	150803	47600	9422	762	0	1000000

(b) (State, action)-frequency table for Tuesday: zero production.

Stock x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	Freq(S_2)
Up-to S_2															
0						625	6171	29367	67457	129870	374060	309674	72958	9818	1000000
Freq(x)	0	0	0	0	0	625	6171	29367	67457	129870	374060	309674	72958	9818	1000000

(c) (State, action)-frequency table for Wednesday.

Stock x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	Freq(S_3)
Up-to S_3															
17														306	306
16													2150	279	2429
15												15339	8300	644	24283
14										2729	39526	43900	1103		87258
13									50893	208093	107255				366241
12							55810	101847	160175	158537					476369
11					10953	26371									37324
10				3987											3987
9		1276													1276
8	527														527
7															0
:															:
0															0
Freq(x)	527	1276	3987	10953	26371	55810	101847	160175	209430	210822	146781	59239	11553	1229	1000000

(d) (State, action)-frequency table for Thursday: zero production.

Stock x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	Freq(S_4)
Up-to S_4															
0			1	216	4424	16363	41376	95387	153179	330485	268262	78406	11637	264	1000000
Freq(x)	0	0	1	216	4424	16363	41376	95387	153179	330485	268262	78406	11637	264	1000000

(e) (State, action)-frequency table for Friday.

Stock x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	Freq(S_5)
Up-to S_5															
17													119	32	151
16												872	446		1318
15											6817	4741	966		12524
14									379	22776	37499	7176			67830
13								427	120672	109750	12028				242877
12							141	163203	198773	70900					433017
11					35466	67646	112238								215350
10				16588											16588
9		6850													6850
8	3495														3495
7															0
:															:
0															0
Freq(x)	3495	6850	16588	35466	67646	112379	163203	199200	191951	132526	56344	12789	1531	32	1000000

Table 4.12: Impact of ordering on Monday, Wednesday and Friday only, compared to daily ordering on Monday to Friday.

(a) Order-up-to S rule.

Order-up-to S rule	S_1	S_2	S_3	S_4	S_5	Outdating	Shortage	Weekly costs
Daily	11	10	11	9	11	8.5%	1.3%	337
Only on Mon-Wed-Fri	13	0	12	0	12	11.7%	2.0%	500

(b) Optimal stock-age-dependent policy (MDP).

MDP	Outdating	Shortage	Weekly costs
Daily	8.5%	1.0%	303
Only on Mon-Wed-Fri	11.7%	1.2%	416

4.5 The hospital case with fixed order costs

In this section we will investigate the impact of making the order costs explicit in the MDP model. The modification of the cost structure in the MDP model is discussed in Section 4.5.1. We investigate the structure of the optimal ordering policy for two scenarios: one with fixed order costs of 75 euro per order, another with fixed order costs of 150 euro per order. Numerical results are presented in Sections 4.5.2 respectively Section 4.5.3.

4.5.1 Optimal control under fixed order costs

An optimal ordering policy can be computed in a similar way as described in Section 4.2.3. The SA algorithm is unaltered except that $C(d, \mathbf{x}, k)$ is to be replaced by $C(d, \mathbf{x}, a, k)$:

$$C(d, \mathbf{x}, k) = \begin{cases} c^{\mathrm{F}} \cdot \mathrm{I}(a > 0) & \text{fixed order costs,} \\ c^{\mathrm{O}} \cdot (x_m - k)^+ & \text{outdating costs,} \\ +c^{\mathrm{S}} \cdot (k - \sum_{r=1}^{m} x_r)^+ & \text{shortage costs,} \\ +c^{\mathrm{H}} \cdot x & \text{holding costs.} \end{cases} \tag{4.10}$$

After a sufficiently large number of iterations, say N, an the optimal strategy with fixed order costs is known, or, if optimal actions are not stored, is to be computed from:

$$\pi(d, \mathbf{x}) = \arg \min_{a \in \mathcal{A}(d,\mathbf{x})} \left(c^{\mathrm{F}} \cdot \mathrm{I}(a > 0) + \sum_{k} p_d(k)\, V_{N-d}(d+1, y(\mathbf{x}, k, a)) \right), \tag{4.11}$$

with $V_N(d, \mathbf{x})$, the expected costs over an horizon of $N - d$ days when the the horizon starts in state $(d, \mathbf{x})$.

After simulation of the resulting optimal policy, one aims in the SDP-Simulation approach at finding a more simple nearly optimal rule. From the literature review, we may expect that an (s, S) policy could be nearly optimal. This is to be checked, and further one need to find for each working day d nearly optimal parameter values s_d and S_d. Hopefully these can be read from the simulation-based frequency tables.

4.5.2 Structure optimal policy when $c^{\mathbf{F}} = 75$

For numerical results we use the following costs figures:

- c^{F} = fixed order costs: 75 per order,
- c^{O} = proportional outdating costs: 150 per outdated pool,
- c^{S} = proportional shortage costs: 750 per pool short.

The fixed order costs of $c^{\mathrm{F}} = 75$ euro is a reasonable estimate of the costs involved in the (scheduled) transport of one or more pools from a blood bank to the hospital. Since multiple deliveries at hospitals can be combined in a route by the same vehicle, the fixed order costs do not relate to a single trip from a blood bank to a hospital. When the hospitals are geographically more dispersed the transportation costs should be set higher. Higher fixed order costs are considered in the next section.

An optimal stock-age-dependent strategy is computed and simulated to investigate its structure. In a simulation of 1 million weeks we count for each observed total stock level x, how often the MDP policy implies a certain order-up-to level. The results are tabulated in the frequency tables in Table 4.13. Optimal order size 0 is made explicit by translating it into order-up-to level 0, as reported in the next-to-last row of each table.

At first sight the tables look similar to those in Table 4.10, but the great difference is read in the next-to-last row of each table from which we read how often the optimal strategy implies zero ordering. Especially on Tuesday and Thursday we observe that in more than 75% respectively 51% of the states visited the optimal order size is zero.

Reading an (s, S)-rule:

An (s, S)-policy seems to apply, since zero ordering is prevalent when the stock level x exceeds some threshold value s. The thresholds are visualized in the Table 4.13(a) to Table 4.13(e) by the additional vertical line. For example, the threshold on Thursday is $s_4 = 7$: when $x > 7$ no orders are placed. Positive order quantities on Thursdays are found only when $x \leq 7$; the optimal order quantity can be approximated by an order-up-to level $S_4 = 9$, which is the most-frequent positive order-up-to level. Apparently for the given cost structure the minimum order-quantity on Thursday is $S_4 - s_4 = 2$ BPPs.

Similarly one reads thresholds s_d and order-up-to levels S_d for the other weekdays. On Friday no threshold is found. On Monday, Tuesday and Wednesday, the optimal strategy

Table 4.13: Frequency tables from 1 million simulations of MDP policy with $c^{\mathrm{F}} = 75$.

(a) (State, action)-frequency table for Monday.

Stock x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Freq(S_1)
Up-to S_1																
18									1			14	32			47
17								10			34	90				134
16							8			59	368	18				453
15						2			161	5251	375					5789
14					1			912	32311	57378						90602
13				1			133	139106	195198							334438
12				24140	51679	97652	176127	110056								459654
11		3541	9682													13223
10	1539															1539
:																0
0										52174	35584	6018	345			94121
Freq(x)	1539	3541	9682	24141	51680	97654	176268	250084	227671	114862	36361	6140	377	0	0	1000000

(b) (State, action)-frequency table for Tuesday.

Stock x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Freq(S_2)
Up-to S_2																
16											88	7				95
15										830	229					1059
14									3706	3012						6718
13								8861	8680							17541
12							27055	20073								47128
11						12065	985		67639							80689
10				2131	5189	3030	16968	44963	27564							99845
9			773													773
8	82	228														310
:																0
0										167138	306259	204034	59062	9326	23	745842
Freq(x)	82	228	773	2131	5189	15095	45008	73897	107589	170980	306576	204041	59062	9326	23	1000000

(c) (State, action)-frequency table for Wednesday.

Stock x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Freq(S_3)
Up-to S_3																
16									19							19
15								82								82
14							19									19
13						6				85540						85546
12					1				84241	109089						193331
11				4494			71595	156108	148289							380486
10			1128		14066	37276	14488									66958
9		279														279
8	72															72
:																0
0										42389	180333	44803	5503	180		273208
Freq(x)	72	279	1128	4494	14067	37282	86102	156190	232549	237018	180333	44803	5503	180	0	1000000

(d) (State, action)-frequency table for Thursday.

Stock x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Freq(S_4)
Up-to S_4																
10							93309	29467								122776
9				11265	32605	133097	71791	109159								357917
8			3407													3407
7		1281														1281
6	561															561
:																0
0									167515	287990	49481	8763	309			514058
Freq(x)	561	1281	3407	11265	32605	133097	165100	138626	167515	287990	49481	8763	309	0	0	1000000

(e) (State, action)-frequency table for Friday.

Stock x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Freq(S_5)
Up-to S_5																
17											4		42			46
16										157	23	456				636
15									927	778	2684	343				4732
14								1519	3833	14525	12191	344				32412
13							1530	5962	33791	61866	8473					111622
12						518	4741	104397	196390	196975	47995					551016
11				14453	47594	99474	83962	48177								293660
10			4409													4409
9		1092														1092
8	352															352
:																0
0												23				23
Freq(x)	352	1092	4409	14453	48112	104215	189889	252048	235526	125321	23375	1166	42	0	0	1000000

Table 4.14: Goodness-of-fit of (s, S)-policy at order costs $c^{\mathrm{F}} = 75$.

(s, S)-policy	Mon	Tue	Wed	Thu	Fri
Threshold s_d	9	8	9	7	S_5
Order-up-to levels S_d	12	10	11	9	12
Goodness-of-fit	55%	85%	65%	87%	55%
Freq. $x \leq s_d$ (do order)	96%	25%	77%	49%	100%

is more stock-age-dependent such that a threshold might not hold uniformly. In some states the order quantity is positive even when the total stock level exceeds the threshold. The threshold s_d on day d is chosen such that at column $x = s_d + 1$ in the majority of states the production is zero and at column $x = s_d$ production is positive at the majority of states. For determining the most-frequent order-up-to level one should restrict to the states where $x \leq s_d$. (Alternatively one may compute the average order-up-to level over all positive (non-zero) order-up-to levels.)

The goodness-of-fit of the (s, S)-ordering strategy is the frequency of states in which the (s, S)-policy prescribes the same order quantity as the optimal MDP policy. The (s, S)-policy on Thursday fits to $357,917 + 514,058 = 871,975$ out of the 1 million states. The goodness-of-fit of the (s, S)-rule on Thursday with $(s_4 = 7, S_4 = 9)$ is thus 87%.

In Table 4.14 we summarize the best parameter values as read from the frequency tables, when the fixed order costs are 75 euro. On Friday there seems to apply no threshold s_5, to emphasize this we report $s_5 = S_5$. Considering the goodness-of-fit values of the (s, S)-policy we conclude that it resembles for a great part the optimal MDP strategy. The last line in the table shows how frequent orders are placed. Virtually every Monday and Friday an order is placed. Only on 25% of the Tuesday BPPs are ordered. On 77% of the Wednesdays and 49% of the Thursdays an order is placed.

In Table 4.15 we compare three policies with respect to their shortage and outdating rates and the average number of orders placed per week. The cost optimal MDP policy gives the best trade-off between these three criteria for the given cost structure. The (s, S)-policy and order-up-to S rule performs nearly-optimal with respect to the shortage and outdating rate. The (s, S)-policy results on average in 1 order more over a period of $\frac{1}{0.2} = 5$ weeks. Ignoring the thresholds s_d results in much more frequent ordering: 4.3 orders per week against 3.4 orders under the optimal strategy.

Compared to the results for ordering on Mondays, Wednesdays, and Fridays only, as reported in Table 4.12, the savings on outdating and shortage is significant. Compared to daily ordering and ignoring any fixed order costs, outdating and shortages have increased, but the number of orders placed per week is much lower: for the optimal stock-age-dependent policy the average number of orders drops form 5 to 3.4 times per week.

Table 4.15: Performance of different policies when $c^{\mathrm{F}} = 75$

$c^{\mathrm{F}} = 75$	Outdating	Shortage	# orders per week	Total weekly costs
Optimal MDP policy	10%	1.2%	3.4	601
(s, S)-policy:				
$s = (9, 8, 8, 7, S_5)$				
$S = (12, 10, 11, 9, 12)$	10%	1.2%	3.6	633
Order-up-to S				
$S = (12, 10, 11, 9, 12)$	10%	1.1%	4.3	660

Conclusions

- The SDP-Simulation approach with fixed order costs successfully solves the MDP and finds nearly optimal (s, S)- policies.
- Order frequencies on Tuesday and Thursday are (much) lower than on the other weekdays.
- On Fridays an ordinary order-up-to S rule applies.

In the next section we investigate the impact of higher fixed order costs on the structure of the optimal strategy.

4.5.3 Higher fixed order costs

We consider two case in which the fixed order costs are increased to $c^{\mathrm{F}} = 150$, respectively $c^{\mathrm{F}} = 500$ euro per order.

Doubled fixed order costs: $c^{\mathbf{F}} = 150$

When the fixed order costs are doubled, one tends to order less frequently and thus more at a time. Consequently one may expect the BPPs in stock to be somewhat older and that outdating thus becomes more prevalent. The optimal strategy likely is thus more stock-age-dependent than when the fixed order costs are low.

In terms of the (s, S)-policy, the thresholds s are expected to be lower and the order-up-to levels S are expected to be higher than for the case with $c^{\mathrm{F}} = 75$. The minimum order quantity $S_d - s_d$ is thus higher when the order costs are increased.

The above expectations are supported by numerical results obtained by the SDP-Simulation approach with the fixed order costs equal to 150 euros. The nearly optimal parameter values that one reads for each working day are summarized in Table 4.16. Compared to Table 4.14 where the fixed order costs were 75, the thresholds s of the (s, S)-policy are indeed somewhat lower and the order-up-to levels S are slightly raised, but not dramatically. Although the optimal strategy is a bit more stock-age-dependent, given that in relatively more states where $x > s_d$ the order-size is positive, the (s, S)-policy fits even better to the optimal strategy when $c^{\mathrm{F}} = 150$ than for $c^{\mathrm{F}} = 75$.

In Table 4.17 we compare the performance of the (s, S)-rule against that of the optimal stock-age-dependent policy. The (s, S)-rule performs nearly optimal, besides an additional 0.3% of the demand (or on average 2 BPPs per year) cannot be met from stock. The average number of orders per week is only 2.9.

Table 4.16: Goodness-of-fit of (s, S)-policy at order costs $c^{\mathrm{F}} = 150$.

(s, S)-policy	Mon	Tue	Wed	Thu	Fri
Threshold s_d	8	7	8	6	S_5
Order-up-to levels S_d	13	12	11	9	12
Goodness-of-fit	85%	94%	78%	81%	52%
Freq. $x <= s_d$ (do order)	87%	15%	50%	41%	99.9%

Table 4.17: Performance of different policies when $c^{\mathrm{F}} = 150$.

$c^{\mathrm{F}} = 150$	Outdating	Shortage	# orders per week	Total weekly costs
Optimal MDP policy	11%	1.2%	2.9	836
(s, S)-policy:				
$s = (8, 7, 8, 6, S_5)$				
$S = (13, 12, 11, 9, 12)$	11%	1.5%	2.9	865

Very high fixed order costs: $c^{\mathbf{F}} = 500$

When the fixed order costs are extremely high, say 500 euro, we observe that the unnatural (s, Q) policy then performs equally well as the more natural (s, S)-policy. In Table 4.18 we report on the performance when the order costs are extremely high (500 euro per order). In general, shortages increase when fixed order costs are raised. The optimal strategy limit shortages since its order quantities depend on the ages of the BPPs in stock. When ages are ignored as in the stock-level-dependent policies (e.g. the (s, Q) and (s, S)- policies), then shortages are much more prevalent. The resulting average weekly costs of the stock-level-dependent policies are more than 20% above the optimal cost level. If this is unacceptable one should search for better rules by developing simple rules that are stock-age-dependent, similarly as we did in Section 3.4.

Table 4.18: Performance of different policies when $c^{\mathrm{F}} = 500$

$c^{\mathrm{F}} = 500$	Outdating	Shortage	# orders per week	Total weekly costs
Optimal MDP policy	17%	1.8%	2.0	1,600
(s, S)-policy:				
$s = (7, 6, 6, 6, 8)$				
$S = (14, 14, 14, 13, 13)$	18%	4.8%	1.9	1,943
(s, Q)-policy:				
$s = (7, 6, 6, 6, 8)$				
$Q = (10, 8, 11, 9, 8)$	19%	4.9%	1.9	1,952

4.6 Summary and conclusions

The contribution of this chapter is two-fold:

1. The SDP-Simulation approach is extended such that it can deal with non-stationary periods (e.g. additional production breaks during holidays), in a further stationary horizon.

2. It is shown how the SDP-Simulation approach derives efficient ordering rules when average demand volumes are much smaller than in the previous chapter. In particular, the problem is studied with fixed order costs.

We have studied the PPP in a setting where no distinction is made between young-demand and any-demand: any age-preference regarding the issued BPPs is ignored. All demand is met by issuing the oldest BPPs first (FIFO), and no mismatch costs apply. In the next two subsections, we summarize the main conclusions for a number of cases.

Production breaks at blood banks

We have investigated the impact on the ordering strategy of some additional production breaks that occur during a year: i.e. on Good Friday, Easter Monday, the two Christmas days and New Years Day. Therefore, the mean demand figures from one of the Dutch blood banks are used. In fact, a bit more difficult case is studied as we assume the demand uncertainty to be somewhat higher than under Poisson distributed demands. We draw the following conclusions:

- Additional outdating and shortages on the day(s) after a production break are to be accepted even under the truly-optimal MDP policy.

- An order-up-to S rule resembles the optimal policy even better on the day prior to an additional production break than on ordinary weekdays.

- When the production break last $m-1$ days, the order problem on the day before the break is a single period problems. The best order-up-to level S just before the break is then a bit higher than the one suggested by a Newsboy equation, as the initial stock will not survive until the next order moment.

- Simulation results over a year, including the production breaks during Christmas, New Year's Day and the 4-days Easter weekend, indicate that compared to the current practice overall outdating and shortage figures can be reduced significantly:
  - outdating from 15-20% to less than 1%,
  - shortages from about 1% to less than 0.1%).

Fixed order costs for hospitals

For Poisson distributed demands the coefficient of variation (cv) of the demand is much higher when the mean demand figures are much lower. Consequently, the trade-off between shortages and outdating is then much more difficult to make. We have executed a numerical study for a case where the demand is only 14.4 BPPs per week, or on average only 2 per day. The cv is then about 0.7. Typically, one may think of a hospital that keeps BPPs in stock, or a small blood bank.

When any fixed ordering costs are neglected, the SDP-Simulation shows that the optimal ordering policy is still well structured: an ordinary order-up-to S rule performs nearly optimal. Nevertheless, for the given data the outdating is substantial: 8.5% under both the optimal policy and the order-up-to S rule. When operating at this scale, the outdating cannot be reduced more through the ordering policy, unless one accepts more shortages.

As daily ordering may be not efficient, regarding the efforts to set-up an order and to arrange the transportation of BPPs to an hospital, ordering on Monday, Wednesday and Friday only will reduce the number of orders to place. Alternatively, we can add any set-up and transportation costs in the form of a fixed order costs per order. The fixed order costs can be high when the hospital is quite isolated from other hospitals and blood banks.

The SDP-Simulation approach successfully solves the PPP with fixed order costs. From the simulation-based frequency tables (s, S) policies can be read. On day d a threshold value s_d applies for ordering: if the stock level is above s_d, no orders are placed. When the total stock level x is on or below the threshold value $S_d - x$ BPPs are ordered.

1. Incorporating the fixed order costs in the MDP model results in cost optimal balancing of shortage, outdating and order costs,
2. An (s, S)-ordering strategy appears to be nearly optimal even for the given relatively short fixed shelf life and the high uncertainty in the demand,

3. Optimal parameter values for the seven thresholds s_d and seven order-up-to levels S_d are easily generated by the SDP-Simulation approach,

4. When the fixed order costs are increased the optimal ordering strategy becomes more stock-age-dependent: the stock-level-dependent (s, S)-policy results in many shortages and is far from optimal when the order costs are extremely high.

Part II

Dynamic control of Traffic Lights

Chapter 5

Introduction and outline of Part II

In 1868, about 30 years before the motor car was invented, the first traffic lights were installed. At that time, traffic lights were introduced in front of the British House of Commons to control the flow of horse buggies and pedestrians. The invention came from the British railroad signal engineer J. P. Knight. The lights consisted of a gas lantern with red and green signals. Unfortunately, these gas lanterns appeared to explode easily.

In the beginning of the twentieth century more traffic signals were developed as motor traffic grew rapidly these days. From 1910 a traffic control system consisted of signals showing 'Stop' and 'Proceed' (or 'Move'), which had to be operated manually. In 1917 automated signals were introduced using two electric lights: red and green. In 1920 the yellow (or amber) light was added by a Detroit policeman. In these days automatic traffic lights were introduced to regulate the priority of the different traffic participants with no intervention of a policeman.[1]

From the early fifties the control of traffic lights is studied from the perspective of car drivers who wish to minimize their delay due to waiting at intersections. Since then several delay models and control policies are developed.

In this part of the thesis we study the optimal dynamic control of traffic lights. In the first section (Section 5.1) we define the scope of our research and we introduce the problems that are studied in the next chapters. In Section 5.2 we report on the most relevant studies on the control of traffic lights. Before we present our approach in the next chapter we report in Section 5.3 on similar approaches to solve other high dimensional problems.

Those who are less interested in the literature should at least read Section 5.1, before they proceed to Chapters 6, 7 and 8.

[1] This introduction into the history of traffic lights is mostly based on the information from [161].

5.1 Three traffic light problems of interest

5.1.1 Single intersection in isolation

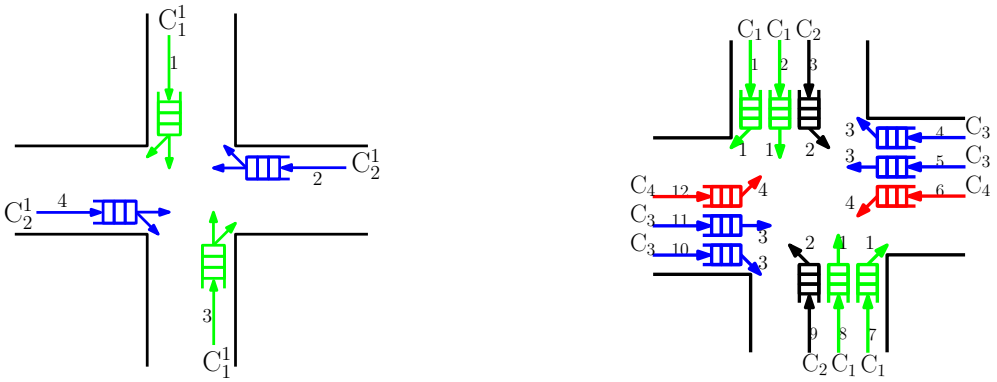
Traffic lights are introduced in the early start of the previous century to make road traffic safer, at places where traffic from different directions cross the same road segment, called the intersection or crossing. By giving right of way to traffic in some direction(s), cars approaching from other directions need to wait before they get priority. By sophisticated control of the traffic lights the overall delay or waiting time of the cars can be kept to a minimum. In practice, traffic engineers aim to set a good (hopefully nearly optimal) control scheme using simulation software, delay formulas and experience. The underlying optimization problem is not always clearly defined by policy makers. Several objectives are possible: minimize the delay, minimize pollution by cars or a combination. In addition, (political) constraints may apply such as public transport traveling at a separate lane has the highest priority over all other traffic flows.

We formulate throughout the next three chapters the control problem as a clean mathematical problem of minimizing the long-run average waiting time, dealing with cars only. We acknowledge that changing priority takes a switching time during which green lights turn into yellow and next an all red phase applies to clear the intersection. Cars continue passing the stopping line as long as the light shows green or yellow. Our definition of waiting time is the time a car spends in the queue, no matter the color of the light: waiting time is thus the time between joining a queue and passing the (downstream) stopping line.

By induction loops, that are cut into the surface of the roads, or by using cameras, one may count (or estimate) the number of cars waiting in each queue. The thus obtained information on the queue lengths can be used in reducing the waiting time at intersections. In the next chapter we formulate the optimization problem for a single intersection in isolation as a multi-dimensional Markov decision problem (MDP). The number of cars waiting in each queue is included in the state description.

Example – Some notations and the computational complexity

For example, consider the infrastructures in Figure 5.1. Figure 6.1(a) depicts a simple intersection with only $F = 4$ traffic flows (or streams) leading to 4 queues. The flows and queues (f) are numbered clockwise: 1–4. Stream 1 and 3 receive green simultaneously and are grouped in combination 1 (C_1). Combination 2, C_2, consists of the streams 2



(a) "F4C2" serve 4 flows in 2 symmetric combinations

(b) "F12C2" serve 12 flows in 4 asymmetric combinations

Figure 5.1: Two typical infrastructures.

and 4. At most one combination at a time has right of way (when its lights are green or yellow). Suppose many more cars are waiting at C_2 than at C_1, but C_1 is currently getting served, a typical question is: 'When should we turn the lights of C_1 into yellow?' Should we wait until both queues of C_1 are empty or should we switch as soon as one of the queues is empty or does an optimal action depend on the precise numbers of cars waiting at each queue? A similar question arise when only one car is waiting at C_2 and the queues of C_1 are almost exhausted: should we keep the lights green to C_1 until more cars are waiting at C_2?

Figure 6.1(b) shows a more complex intersection where $F = 12$ flows are grouped into 4 combinations C_1 to C_4. C_1 and C_3 consist of twice as many flows than C_2 and C_4. The grouping of the streams is also indicated by the numbers in the front of each queue. The asymmetry in the number of queues in the four combinations, complicates the decision making. As long as some cars are waiting at all queues, giving priority to C_1 or C_3 results in more departures per unit time than giving priority to C_2 and C_4.

The decisions *'when-to-end-a-green-period'* as well as *'which-combination-to-serve-next'*, should depend on the number of cars waiting at each queue: $\mathbf{q} = (q_1, q_2, ..., q_{12})$. An optimal policy prescribes for each possible state, i.e. each value of the vector $\mathbf{q}$, the action to take given the current state of the lights. Such an optimal policy can be obtained from the solution of the underlying Markov decision problem. When each element q_f can take only 10 possible values (0 to 9), then there are already 10^{12} possible vectors $\mathbf{q}$ to consider. Thus the state space of the MDP easily exceeds the computationally acceptable limit of say 10 million states.

Solving the MDP is only possible for simple infrastructures with a 'small' number of traffic streams and under non-saturated conditions such that the queue lengths can be bounded. Large MDPs cannot be solved due to the large number of states. Therefore we develop an approximate solution, based on a one-step policy improvement algorithm. Such an approach has been successful to other types of problems, which are discussed in Section 5.3. In Chapter 6 the new approach is presented for the control of a single intersection in isolation. As we will see, these new policies greatly improve existing policies that we have evaluated in a simulation model.

5.1.2 Isolated intersection with arrival information

With todays technology one can observe the number of cars waiting in each queue, as well as the individual cars driving on each lane ([93], [65]): hence the arrival times to the queue of nearby cars can be predicted. This information can be used in a control policies that anticipates near-future arrivals. In Chapter 7, we extend the new policies that are introduced in Chapter 6, by including arrival information. In a detailed (microscopic) simulation model we show that the average waiting time can be reduced by taking only a limited amount of arrival information into account.

5.1.3 Networks of intersections

Finally, in Chapter 8, we consider several networks of intersections. We suggest to control the traffic lights in the network for each intersection in isolation of all others. By using some arrival information, we hope to realize a desirable degree of synchronization of the intersections. The resulting policies are evaluated by an extended simulation model, in which we may simulate an arbitrary number of intersections at different workloads and where car speeds may differ. Main criterion remains minimizing the long-run average waiting time that a car experiences at an arbitrary intersection.

5.1.4 Outline

Before we present the MDP approach for the above three problems in the next three chapters, we successively discuss in this chapter relevant studies on:

- the control of traffic lights, and
- the application of a one-step policy improvement approach to high-dimensional MDPs.

5.2 Studies on control of traffic lights

The optimal control of traffic lights is studied by researchers from different disciplines using different techniques:

- Operations Research-ers study the problem using mathematical techniques to evaluate and optimize the control of the lights,
- Computer scientist use techniques from Artificial Intelligence (AI) to develop heuristics, local search procedures and learning algorithms, and
- Simulation experts develop genuine microscopic simulation models to carefully evaluate different control schemes.

There is a rich amount of literature making it impossible to give an extensive overview. Instead we address some relevant articles. The studies on the control of traffic lights can be divided into two categories:

1. studies on the static control of the lights, in which the control is fixed and does not depend on the actual situation at the intersection,
2. studies of the dynamic control of the lights that use information on the actual situation at the intersection.

In the next sections we discuss a number of control schemes that are well studied.

- FC or pre-timed control,
- Exhaustive control,
- Vehicle actuated control, and
- Adaptive control.

5.2.1 Fixed cycle

The fast majority of the Operations Research literature on traffic control is on fixed cycle control (FC), since under FC the control has a simple structure that allows mathematical analysis. FC is a static control policy: it does not take into account any information that changes dynamically over time. Under FC the order in which queues get green as well as the duration of each green period is fixed. Hence the decisions about when to switch are predetermined and do not depend on the actual number of cars waiting at the queues. Therefore FC is also called fixed-time or pre-timed control.

A traffic responsive version of FC allows to choose between several predetermined FCs: at times at which it is very busy one applies an FC with a long cycle length. Hence during rush hours a longer cycle length applies than during the rest of the day. During the night the lights might be even turned off.

One of the first and most cited studies on FC is that of Webster in 1958 at the Road Research Laboratory in London [162]. By applying analytical and numerical techniques (simulation and curve fitting) he derived his famous delay formula based on the M/D/1 queueing system:

$$\mathbb{E}W_f \approx \frac{(D-d_f)^2}{2D(1-\lambda_f/\mu_f)} + \frac{\lambda_f/\mu_f D^2}{2d_f(\mu_f d_f - \lambda_f D)} - 0.65\left(\frac{D}{d_f^2}\right)^{1/3}\left(\frac{\lambda_f D}{\mu_f d_f}\right)^{2+5d_f/D} \tag{5.1}$$

which estimates the expected delay $\mathbb{E}W_f$ of a car in flow f, as a function of the duration of a cycle D, the effective green time d_f during which departures may happen from queue f, and the arrival and departure rates λ_f and μ_f.

Shortly after the introduction of Webster's delay formula, other approximations were developed; see the work of Miller [94], Newell [103] and McNeill [90]. McNeill's delay formula is exact when one would have an exact expression for the average queue length at the start of a red period. In 1964 such an expression was found by Darroch.

In [33] Darroch presents a formal solution for the stationary distribution of the queue length under FC at discrete points in the cycle. Herewith Darroch provides an expression for the mean number of cars queued upon the start of a red period. However, the expression requires extensive numerical computations, for this and other reasons his results was not much used. In 1978 Darroch's approximations were reformulated by Ohno [106]; he made Darroch's analysis more accessible and compared the different approximations available at that time. Nowadays the computational problems to evaluate Darroch's exact

expression are resolved, since the numerical schemes are clarified and computing power has increased. Nevertheless, approximations are still very useful in providing insight into the performance of FC and to heuristically optimize FC [145].

Recently, in 2006, Van den Broek, Van Leeuwaarden, Adan and Boxma provided new bounds and approximations, see [146]. They provide efficient numerical procedures for some of their results, and they have derived also some closed-form expression that do not require numerical procedures.

Van Leeuwaarden extends the work of Darroch on the queue length distribution. He derives in [153] the probability generating function of the queue length distribution and the delay distribution for more general discrete arrival distributions. (Darroch focussed on (compound) Poisson arrivals in a discrete time model.) Van Leeuwaarden argues that the numerical efforts for finding the distribution are not a big issue any more, since numerical search procedures can be structured as FC is well-structured. Once the distributions are computed, one easily computes the performance measures such as the mean, the variance and the percentiles of the waiting time.

So far, all articles focus on a single intersection in isolation. The control of the traffic lights of successive intersections of an arterial is considerably more difficult, since one has to specify the start of the green periods, next to the length of each green period. Setting green waves in 1 or 2 directions might contribute to the minimization of the waiting time. In the master thesis of Van den Broek [145], the developed approximations are used to optimize FC at some realistic intersections. The approximate delay functions are transformed into spline functions, such that a Mixed Integer linear Programming model (MIP) can be formulated and solved to determine (nearly) optimal lengths and starting times of green periods. The approach is also used to control an arterial of two intersections.

Another technique for finding a good synchronized FC is by using a genetic algorithm. In TRANSYT-7F hill climbing techniques and genetic algorithms are implemented to heuristically optimize FC for a network of intersections ([122], [84]). Park et al. [109] discuss a genetic algorithm that optimizes over the start and length of the green periods as well as over the sequencing of the phases of FC: a phase is a period of time during which a specific combination receives green. We do not discuss all software that is available for this purpose. For an overview we refer to surveys like the one of Papageorgiou et al. [108] and the survey published on the web site of the ITS group of the University of Leeds (see [160]).

5.2.2 Exhaustive control

Under exhaustive control a (effective) green period of a combination is ended as soon as all queues related to that combination become empty. Based on the visiting order, one distinguishes cyclic and acyclic exhaustive control policies. Under cyclic control the order in which the combinations are served is fixed, but a combination may be skipped in a cycle when no cars are waiting upon the intended start of the green period. Under acyclic control we need to specify which queue to serve after an all-red phase. For example, this could be the combination with the highest number of waiting cars or the combination that contains the longest queue. We distinguish two classes of exhaustive control policies based on the definition of 'exhaustion': *pure-exhaustive control* and *anticipative-exhaustive control*.

Pure-exhaustive control

Under *pure-exhaustive control* signals stay green until all queues that have right of way are empty, say 'exhausted'. We abbreviate the cyclic pure-exhaustive control policy by XC; XA is the acyclic variant. Note that XC does not require detailed information concerning the number of cars queued: next to the state of the traffic lights, XC needs for each queue only information on whether cars are present or not.

Anticipative-exhaustive control

Cars leave a queue as long as its lights show green or yellow. A policy may anticipate the departures during a yellow slot by switching from green to yellow before the queues that have right of way are exhausted. Therefore we introduce, what-we-call, *anticipative-exhaustive control* policies. Under anticipative-exhaustive control, departures during the yellow phase are anticipated by switching already when at each of the queues that have right of way one or two cars are present. These polices are denoted by XC-1 (or XA-1) respectively XC-2 (or XA-2). Compared to pure-exhaustive control, anticipative-exhaustive control needs information about the actual number of cars waiting at each of the queues but ignores any information on near-future arrivals.

Remark – Polling models: For intersections with only a single queue per combination, the analysis of exhaustive control can be done by modeling the service of the queues by a polling model. Polling models have been extensively studied in Queueing theory: in

a polling model a single server visits one queue at a time to provide service to jobs or customers in that queue only. For a few problems, it has been proven that exhaustive control is optimal under specific conditions. When switching costs or switching times apply, switching thresholds may apply: one switches only when the queues that get service become empty and 'enough' jobs, say cars, are waiting at the other queues. Most of the articles deal with production or service networks rather than signalized intersections.

A rich amount of articles have been published on polling systems. We do not discuss polling models here, since at most infrastructures a combination consists of multiple queues. For an overview of such queueing models we refer to surveys of Takagi ([136], [137],[138],[139]), and Adan, Boxma and Resing [1]. Specific polling models that acknowledge a positive switching time are discussed by Hofri and Ross in [67], by Boxma, Levy and Weststrate in [20], by Reiman and Wein in [121], by Van der Mei in [91], and by Winands in [167].

5.2.3 Vehicle actuated control

In practice, one often finds a mixture of FC and exhaustive control: lights stay green until the queues are exhausted or a maximum green time has elapsed. An induction loop cut in the pavement upstream at a fixed distance of the stopping line detects when a car is approaching the stopping line. When no car has passed this induction loop for β seconds, the green period of that stream is ended. The green phase is also ended when some predetermined maximum green time has elapsed. Next to a maximum green time often a minimum green time applies. Such a policy is called *vehicle actuated control*, since the actions are triggered by the presence of vehicles.

Two of the first articles on vehicle actuated control appeared in 1964 by Darroch, Newell and Morris [34], and by Dunne and Potts [39] respectively. Dunne and Pott relate their algorithm to a kind of vehicle actuated control: the control functions they propose allow dynamic control based on the actual number of cars queued and thresholds apply in determining when to switch. Darroch et al. study the impact of the choice of β on the average delay. In their paper they provide formulas for the mean and variance of the (variable) cycle length, for the delay per vehicle and for the optimal value of β_f for each flow f.

Some later studies on vehicle actuated control are by Cowan [32], and Akcelik [3]. In 2002 Taale [135] compares different methods to optimize vehicle actuated control.

5.2.4 Adaptive control

Under adaptive control the lights are adjusted based on the actual conditions of the traffic flows. According to this definition vehicle actuated control is adaptive. Pre-timed control can be adaptive when from a set of off-line calculated FCs a best one is selected, e.g. based on the workload over the last say 5-15 minutes. Existing control systems, like SCOOT and SCAT ([87], [70], [86]), are based on this idea. Under adaptive control, we thus assume new information to be fed to the controller on a regular basis, say every T seconds. An adaptive control policy that gets frequent updates of the state of the system has, what we call, a high resolution.

In this subsection, we restrict adaptive control to strategies that take into account at least the information on the actual number of cars waiting. According to the definition by Van Katwijk, De Schutter and Hellendoorn [151], adaptive control strategies do optimize online over the state of the traffic lights. Therefore an adaptive control considers the actual numbers of queued cars and, if available, estimated arrival times of cars on their way to the intersection(s).

Based on the system's state, a number of actions is evaluated over some planning horizon and a best one is selected for implementation. Commonly a rolling planning horizon approach is used, since new information becomes available, say every T seconds. The optimization process to find the best (set of) decision(s) to implement for the next T seconds should be done in real time. All necessary calculations have to be done in a short time-window, at least before new information becomes available. The computation times can be kept low, when the evaluation of a decision is based on a short planning horizon. However, an accurate evaluation needs an horizon of sufficient length.

For an accurate evaluation of a decision one commonly takes future arrivals and future decisions into account. From the decision tree in Figure 5.2, it becomes clear that the number of sequences of decisions until the end of the planning horizon is huge. At each square in the decision tree a decision concerning the lights has to be taken: in the illustration it is assumed that there are two possible decisions every decision epoch. Each circle represents the event of arrivals and departures, which is modeled by a stochastic experiment when the arrival times of cars are unknown. The potential number of states to consider in the last layer, can be very large, since the number of events is exponential in the number of flows F. In Figure 5.2 it is assumed that the arrival times of cars are known for the first two time intervals, this reduces the number of nodes significantly.

Commonly the potential number of paths to evaluate is thus very large. Fortunately

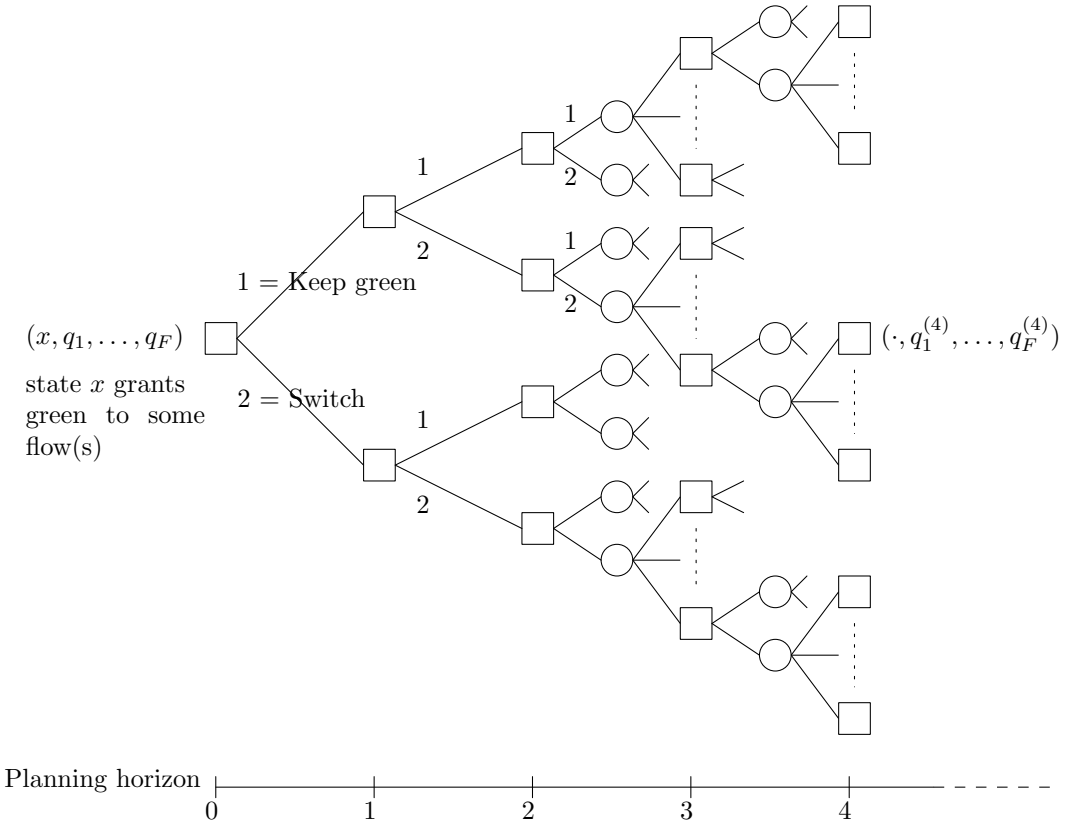


Figure 5.2: Decision tree under adaptive control.

standard OR techniques will skip the evaluation of many sub-optimal paths. By making use of Bellman's principle of optimality the computations can be done efficiently by forward Dynamic Programming (DP)[14]. DP is applied in existing control systems OPAC, PRODYN and RHODOS, see [48], [63], and [127] respectively. Alternatively, one uses a Branch and Bound (B&B) technique to find an optimal sequence of decisions. B&B techniques are used in ALLONS and in UTOPIA, and in RHODOS in combination with DP (see [116], [88], [127]). Even when applying DP or B&B algorithms, the computation time exceeds the available time when the depth of search tree is too large.

In the traffic engineering literature common ways to limit the computations are:

1. leaving out unknown future arrivals and evaluating waiting time until all present cars have left; this way the number of branches in the tree is reduced significantly,

2. decreasing the resolution by which decisions are taken, such that the number of layers in the tree reduces, and

3. shortening the planning horizon.

The first option is clearly inaccurate, since the impact of the decision on unknown future car arrivals is ignored. Hence, when the departure process is deterministic, only deterministic components of the waiting time are considered. This approach is implemented in OPAC, PRODYN and UTOPIA, see [48], [63] and [88] respectively. The resolution at which decisions are updated is 3 seconds for UTOPIA and 5 seconds for PRODYN and OPAC.

The latter two options do consider future arrivals, but ignore any delay experienced by cars that stay behind at the end of the planning horizon. Therefore one often applies terminal costs to value the impact of a decision on cars that depart (or arrive and depart) after the planning horizon, see Newell [104] and Shelby [128]. Bell and Brookes [13] state that the rolling planning horizon should be 2 minutes to compute the short term delay, and the effects beyond the horizon are estimated by state dependent terminal costs. From Figure 5.2 it becomes clear that the evaluation of the short term delay over 2 minutes may be too time consuming: when the decisions are taken every 5 seconds, then there are 60 layers in the search tree.

In ALLONS the fixed time intervals can be up to 15 seconds: every 5-15 seconds an optimal (sequence of) decision(s) is updated. In the online evaluation of a decision only deterministic arrivals are assumed, based on the predicted arrival times of all cars driving up stream. Van Katwijk et al. [152] suggest to split the planning horizon in time periods of unequal length. The estimation of the delay over a time interval requires a delay model. Therefore simulation can be used, but Van Katwijk et al. argue that a detailed microscopic simulation program may not fit when the update-frequency is high, due to the high computation times of such an program. Nevertheless more and more simulation models are used that take into account the physical length of queued cars. The phenomenon that a queue may grow upstream is called in the traffic engineering literature *queue spillback*.

Stochastic car arrivals are considered by Bell [11] using Markov chain techniques that require the storage of the expected delays for all possible state vectors. In PRODYN and RHODOS states are aggregated according to some approximate state equivalence relation. Yu and Recker formulate in [168] an MDP and transform the queue state description by imposing a threshold for each queue: when the number of cars queued is above the threshold the state is called saturated otherwise the state is non-saturated. This way they reduce the number of states in the MDP significantly. The transformation of the transition probabilities is however not clearly presented, which makes it hard to asses the approach.

When arrival information is not available, the number of sample paths can be very large, as illustrated in the right part of the decision tree in Figure 5.2. By simulation a decision can be evaluated over a number of arrival patterns (or sample paths). Since simulation requires less computations, a longer planning horizon can be considered. When only a small fraction of all sample paths is evaluated, the accuracy of the estimated delay may be poor.

Adaptive control policies are popular subjects of study for computer scientists in the field of Artificial Intelligence (AI). Commonly used techniques are learning algorithms and neural networks. Neural Networks are used to predict the delay that may be input to a dynamic program to find an optimal sequence of decisions. See for example the work of Theodorović et al. ([142], [143]).

An interesting application of a learning algorithm is that of Wiering et al. [164]: Q-learning is applied to heuristically minimize the overall average delay per car by dynamically adjusting the lights and routes traveled by cars having specific destinations in a network. The prediction of the delay is made in a microscopic simulation environment through reinforcement learning, so no prior known distributions of car arrivals are used. By dynamically computing routes to the known destinations of the cars traveling in the network, the algorithm aims at spreading the traffic over the different intersection such that the overall waiting time is reduced.

Tan et al. [140] and Lee et al. [83] use fuzzy logic. There are many more interesting articles published in the domain of AI. We do not go into the details here. For a review of the advances in traffic signal control we refer to [12], [37], and [108].

5.2.5 Simulation

Simulation plays an important role in testing the different control strategies. Several simulation programs are developed: most of them are microscopic simulation models, in which individual cars are generated such that the processes can be modeled at a realistic level. At the ITS group of the University of Leeds (UK) a thorough review [160] is executed of more than 30 (microscopic) simulation programs, like VISSIM, MIXIC, MICROSIM, and CORSIM. Most of them are developed by the scientific community to test new control policies against existing ones: nine of them are commercially available.

5.3 Studies on Decomposition and One-step policy improvement to solve high dimensional systems

Despite the fact that dynamic programming principles are known to and used by computer scientists, there appears to be no article in which a one-step policy improvement algorithm is used in combination with decomposition of the state space. Nevertheless, such an approach is proven to be beneficial to solve high-dimensional MDPs as we will learn in this section. In this section an overview is given of some applications for which a high-dimensional MDPs is approximately solved by a one-step policy improvement algorithm after decomposition of the state space. We skip the technical details and simple report the modeling assumptions that allow the decomposition of the state space, such that a one-step policy improvement can be executed. The idea to approximately solve a high-dimensional MDP by a one-step policy improvement algorithm based on decomposition of the state space dates back to Norman [105]. As will become clear, the way to decompose the state space is problem specific.

5.3.1 Ordering in multi-item production-inventory systems

In 1979 Wijngaard [165] discusses a production-inventory problem in which production capacity is shared for the production of multiple products. Sequential decisions concern the production volumes for each product. Since production capacity is to be shared, the decisions are dependent and should be taken integrally. Since the state description is the number of items in stock of each product, the total number of states can be very large. However, when the shared capacity is sufficiently large, the underlying MDP, in which capacity is finite and to be shared, can be approximated by an MDP model in which capacity is infinite.

The MDP with infinite capacity can be decomposed into subproblems for each product. Hence an order decision for one product can be evaluated separate of the other products. To come to an approximate solution one makes an integral ordering decision based on a finite capacity for the current decision epoch, but future decisions are evaluated as if the production capacity is not shared. The approach is thus a one-step policy improvement of the optimal policy for the case of infinite capacity.

Note that Wijngaard assumes that the products are no substitutes to each other and do not share order costs: the sole correlation between the products is through the shared capacity. A similar approach is suggested in [165] to solve the problem of ordering products in a two-echelon inventory system.

Federgruen, Groenevelt and Tijms [45] study a more complex multi-item inventory system: a fixed order cost K applies when an order is placed for any of n items (next to fixed order costs k_i for each item i). They study the common (s_i, c_i, S_i) ordering strategy (with $s_i \leq c_i < S_i$): orders are placed as soon as the inventory position of one or more items drops below their critical value s_i. For each item i one orders $S_i - x_i$ products whenever the inventory position x_i is at or below c_i. In their paper they make use of a decomposition scheme introduced by Silver [129]. By a policy improvement algorithm they search for the best values of (s_i, c_i, S_i) for each item i, independently of the other items. Through an iterative scheme the values of (s_i, c_i, S_i), for all items i, are successively adjusted to obtain a coordinated replenishment rule of the (s_i, c_i, S_i) type. Although the thus-obtained coordinated ordering policy is sub-optimal, numerical results indicate a great cost saving compared to uncoordinated ordering.

5.3.2 Routing telephone calls through a network

Another application of a one-step policy improvement algorithm is that of routing telephone calls through a network of switches as studied by Ott and Krishnan in [107]. They consider a network of n nodes or switches through which a call request between any node pair $k = (a, b)$ can be routed, with $a, b \in \{1, 2, \ldots, n\}$. The total number of node pairs is $K = n(n-1)/2$. Between node pair k maximal S_k (direct) channels are available, when all channels are occupied no call can be directed over channel k. When all pairs are connected directly, the number of possible routes from node a to b is exponential in n.

The criterion for selecting a route is to minimize the long-run average number of blocked calls for the entire system. For call requests between a node pair k, one thus needs to specify the best route to select, given the number of occupied channels for all K node pairs: $(x_1, x_2, \ldots, x_K)$. Although the problem can be formulated as an MDP, its state and action space are far too large to compute an optimal routing policy. To keep the number of occupied channels low, one wishes to make use of a low number of channels. When a direct connection is available, that connection can be used. Alternatively, the call is directed via an intermediate node. In the latter case the authors speak of a 2-hop call, since two serial channels are used.

The 1-hop routing policy that allows only direct routing has two possible actions: to block a call or to accept the call when at least one direct channel is available. The decision does not affect the acceptance or rejection of calls between other pairs, and thus the blocking rate for node pair k can be evaluated in isolation of all other node pairs. The blocking rate of the entire system is then simply the sum of the blocking rates over all node pairs. The long-run average cost function $v(x_1, x_2, \ldots, x_K)$ under the 1-hop routing policy, when starting in a given state $(x_1, x_2, \ldots, x_K)$, is separable in the cost functions $v_k(x_k)$ for each node pair k: $v(x_1, x_2, \ldots, x_K) = \sum_k v_k(x_k)$.

The separable routing policy that Ott and Krishnan formulate comes from a single policy improvement step over the 1-hop routing policy. For any call a number of possible routes (including indirect routes) is available. The evaluation of a route is done by assuming future calls either to be routed through a direct channel or to be blocked. The separable state-dependent routing policy shows a number of practical advantages over classical routing schemes, and has lead to a number of patents.

5.3.3 Assigning customers to queues

A different problem arising at telecommunication networks is addressed by Bhulai [18]. He considers a multi-skill call center, at which incoming calls have to be directed to a service group. The destination of a call is thus not fixed. Each call requires a specific skill. The agents in the call-center are grouped according to the different skill sets they possess: agents in a single-skill agent group possess a single specific skill only, more flexible agents possess multiple skills. A call can thus be handled by several agent groups, when the agents in the group possess the right skill(s).

Bhulai treats two scenarios. In the first scenario the system has no buffers, hence if a call cannot be directed to any available agent that possesses the right skill(s), the call is blocked and lost. The routing has to be such that the blocking rate is minimized. In a second scenario each agent, or group of agents with the same skill set, has an finite buffer in which calls are held when all agents are occupied. Incoming calls have to be routed directly and cannot be rerouted. The objective in this case is to route incoming calls to a queue of an appropriate agent group, such that the long-run average number of calls in the system is minimized. We deliberate a bit more on this last scenario.

For the optimal routing, one considers the state of each agent group: the number of calls either in the buffer or in service. The system's state space has as many dimensions as there are agent groups, consequently the number of possible states can easily become

very large. To come to an approximate solution, Bhulai suggests to execute a policy improvement step over a policy that allows the decomposition of the system into its agent groups. When an incoming call cannot be directed to a single-skill agent group (because its buffer is full), the call is directed to a second layer of more flexible agent groups. In the computation of the relative value function he assumes that the overflow can be approximated by a Poisson process and that the routing at the second layer happens according to fixed routing probabilities for all groups that possesses the required skill. Hence under the initial policy, the routing of a call that requires skill k depends only on the number of calls buffered at or being served by the single-skill agent group k.

Under this initial routing policy the state of an agent group is statistically independent of the state of the other agent groups. Therefore the average number of calls present at a group can be analyzed in isolation of all other groups. Relative value functions are derived based on this decomposition and under the assumption of Poisson arrivals and exponentially distributed service times. Next, these relative value functions are used in a single policy improvement algorithm: each incoming call is assigned to an agent group based on the actual number of calls in each agent group, but all future calls are assumed to be routed randomly, according to the initial policy. Numerical results demonstrate that the resulting dynamic routing policy performs quite well.

The latter problem is related to the problem of routing customers to parallel queues as studied before by Sassen, Tijms and Nobel (See Sassen et al. [124]). In their paper they extend the results of Krishnan and Ott [80] and Krishnan [79], by allowing the service times to be generally distributed. Sassen et al. assign an incoming customer to a queue according to the Bernoulli-splitting rule. With a fixed probability, p_i, a customer is assigned to queue i. First, optimal values of p_i, for $i = 1$ to n, are determined such that they minimize the long-run average sojourn time per customer. Next, the relative value of states under this splitting rule are derived in an analytical way. Finally, the separable heuristic algorithm is formulated. Numerical experiments show that the separable rule combines good performance with minor computing time.

5.4 Conclusion

The dynamic control of traffic lights at road intersections is a high-dimensional decision problems, when the number of cars waiting at each queue is taken into account. Therefore one relies on heuristics and approximate solutions to minimize the average waiting time per car at an intersection. The static control of traffic lights by a so-called fixed cycle control policy (FC) is well studied. For the dynamic control vehicle actuated control schemes are considered but hard to optimize. A number of adaptive control schemes rely on dynamic programming over a finite horizon, but the dimensionality of the problem hampers an exact approach.

Solving or circumventing the curse of dimensionality in high dimensional control systems requires a problem specific approach. For a number of applications decomposition of the state space is possible, such that a one-step policy improvement algorithm can be executed. In the examples given in Section 5.3, decomposition of the state space is possible either by modifying the problem assumptions or by imposing a special well-structured policy. Whereas the decomposition of the call routing and customer assignment problems rely on a randomized routing strategy, such a strategy may not be applicable in the control of traffic lights.

There is no universal way of finding a special strategy that allows the decomposition of the state space. Instead, it is a problem-specific approach that requires intuition and insight in the problem under consideration. If a policy does exists that allows the decomposition of the state space, a one-step policy improvement does not necessarily yields a great improvement. Nevertheless, it seems to be worth to investigate whether a good approximate policy may be obtained by a one-step policy improvement approach.

Chapter 6

Single intersection in isolation

In this chapter we focus on the dynamic control of a signalized intersection in isolation. We introduce some new control policies based on an MDP formulation. By simulation the new policies are tested and compared to other control policies for a number of isolated intersections. Some signalized intersections are part of a network, but often the control of the lights can be done for each intersection in isolation. The approach is extended in the next chapters. In Chapter 8 we consider networks of intersections.

6.1 Modeling the problem at a single intersection

In Section 5.1.1 we have introduced the problem of minimizing the overall average waiting time at a single intersection in isolation. Our definition of waiting time is the time spent in the queue, hence the time between joining a queue and passing the stopping line. Throughout this (first) chapter we assume cars to have zero length to ease the analysis. This assumption corresponds to vertical queueing. Our definition of waiting time corresponds then to the definition of waiting time that is common in queueing theory. In the Chapters 7 and 8, we do acknowledge queued cars to take space. As we will see this slightly affects the waiting times reported by the simulation model since then cars sooner hit the tail of a queue, when the queue length increases.

In this section we formulate the problem in more detail, introduce the notations that we use in this and the next chapters and we discuss some modeling assumptions.

6.1.1 Infrastructure

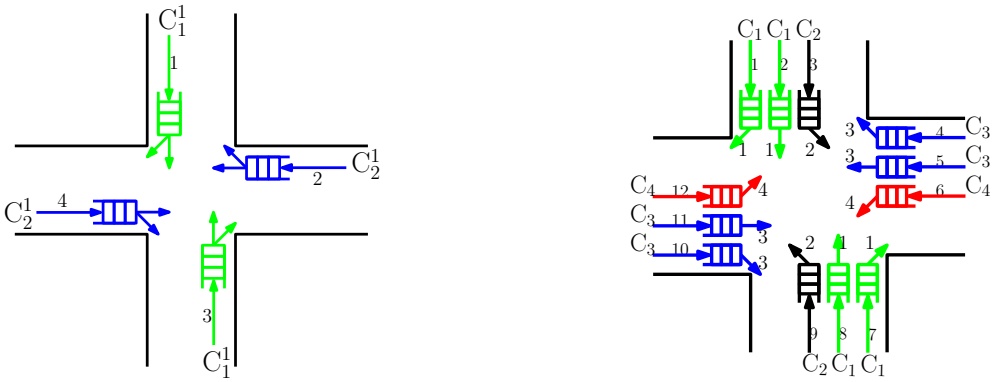
Figure 8.4 shows two typical cases that we are interested in. We refer to these cases by code names: F4C2 for an intersection of $F = 4$ traffic flows grouped in $C = 2$ combinations and F12C4 an intersection with $F = 12$ flows grouped into $C = 4$ combinations. The set of F flows is thus divided in C disjoint subsets $\mathcal{C}(1), \mathcal{C}(2), \ldots, \mathcal{C}(C)$, called *combinations*. In We abbreviate specific queues and combinations: Q_f refers to the queue of cars in traffic flow f and C_s abbreviates *combination* s consisting of flows $\mathcal{C}(s)$. The flows, queues, and combinations are numbered clockwise (rather than using the more detailed notation that is common in traffic engineering). Both F4C2 and F12C4 consist of four *approaches* or directions from which traffic comes. We refer to these directions by North, East, South, West, as if the top of each figure faces North. North and South are *opposite approaches*, and so are East and West.

At F4C2 each approach consists of a single lane. Cars are not allowed to turn left and thus flows 1 and 3 are not in conflict and constitute C_1, flows 2 and 4 constitute C_2. Ongoing or through-traffic is referred to as *'thru' traffic*. At F12C4 each of the four approaches consists of three separate lanes for right-turning, thru, and left-turning traffic. The left-turning traffic of two opposite approaches are served at the same time. The right-turning and thru traffic of two opposite approaches are served together in a combination of four flows. The 12 flows are thus grouped in 4 combinations (C_1 to C_4). C_1 and C_3 are composed of four flows each, whereas C_2 and C_4 have only two flows:

$$\mathcal{C}(1) = \{1, 2, 7, 8\}, \quad \mathcal{C}(2) = \{3, 9\}, \quad \mathcal{C}(3) = \{4, 5, 10, 11\}, \quad \mathcal{C}(4) = \{6, 12\}.$$

We assume that these combinations are given and fixed. The order in which the combinations are served (=get right of way) maybe cyclic or acyclic. Under cyclic control the combinations are visited in the order: 1, 2, 3, 4, 1, etcetera.

Summary of notations – An intersection consists of F flows of car arrivals; each flow f has a single lane and a single queue Q_f. The vector $\mathbf{q} = (q_1, \ldots, q_F)$ stores the numbers of cars waiting in each queue. The set of all flows $\mathcal{F} = \{1, \ldots, F\}$ is partitioned into C disjoint subsets $\mathcal{C}(1) - \mathcal{C}(C)$ of combinations of non-conflicting flows that will receive green, yellow and red together. We call a subset of flows a combination (of flows). The set of the C combinations, $\mathcal{S} = \{1, \ldots, C\}$, is given and hence not part of the optimization problem. Flows in the same combination will always receive green, yellow and red together. If one combination has green or yellow, all other lights show red. We assume that *conflicting traffic flows* are not part of the same combination.



(a) "F4C2" serve 4 flows in 2 symmetric combinations

(b) "F12C4" serve 12 flows in 4 asymmetric combinations

Figure 6.1: Two base-case infrastructures.

6.1.2 Discrete time modeling assumptions

The (automated) controller of the traffic lights periodically takes decisions about how to adjust the lights. When switching from green for one set of flows to green for another set, a fixed switching time is to be acknowledged. This suggests to model the problem in *discrete time* which calls for some simplifications of the processes.

Discrete time

The time unit or slot is taken to be the (average) time a car needs to pass the intersection when the light is green. We fix the slot length to two seconds, which corresponds also to a safe driving distance between cars at different speeds. When car drivers keep all the time a safe driving distance of exactly 2 seconds, then either 0 or 1 cars passes the stopping line within a slot.

Signal process

Changing from green for one combination to green for another combination of flows takes ($\gamma =$) 3 slots (6 seconds): ($\beta =$) 2 slots of yellow and ($\gamma - \beta =$) 1 slot in which all lights show red to clear the intersection. This *switching time* is assumed to be independent of the two sets of flows involved. In addition a minimum green time, of say 1 slot, applies such when a combination is served it signals shows green for at least 1 slot. (In our model all slots are of identical length. Although notations become more involved, the approach can be extended to different slot lengths and to different, flow-dependent switch-over times.)

Arrival process

In the basic model of this chapter cars are not observed until they arrive at the stopping line at which they are queued vertically. The position of the tail of a queue corresponds thus with the stopping line, as if cars take zero length. It is assumed that the arrival of cars at the queues occur uniformly over time rather than in platoons that may be formed at any upstream signalized intersection. Arrivals at different queues and in different slots are independent. Per flow the number of car arrivals in a slot is either 0 or 1. The probability of an arrival in flow f is λ_f: the arrival rate at queue f. (Instead of a Bernoulli arrival process one could assume Poisson arrivals but having more than 1 car arrival within a slot at a single queue does not happen when cars keep a safe traveling distance of one-slot.)

Departure process

When a signal shows green or yellow and as long as cars are present, every slot (exactly) one car passes the stopping line in a deterministic fashion. Remember car drivers acknowledge a safe driving distance of 1 slot. In countries where yellow officially means 'stop-when-safe-to-do-so' it might be more realistic to say that during the first yellow slot the probability that a car passes is still (close to) 1, but during the second yellow slot it is less than one [1]. The model allows for this, as it allows for any Markovian stochastic departure process, but notations would become more involved. Assuming deterministic departure processes with an inter-departure time of 1 slot (2 seconds) is quite common in the practice of traffic engineering ([119]).

In the cases that are studied, conflicting traffic flows do not get simultaneously right of way. In practice conflicting flows might have green at the same time, in which case (regular) basic traffic rules apply. As will be discussed at the end of this chapter, the models we present can be extended at this point.

Queueing process

A car arriving at an empty queue that has right of way passes the stopping line without delay. Therefore we assume new arrivals to take place at the beginning of the slot (just after observing the state of the queues and the lights). When one or more cars are queued a new arrival joins the end of the queue. Departures from the queues take place at the end of the slot prior to the observation of the new state. In Figure 6.2 we depict the modeling of the course of events and actions.

[1] In most countries, the sequence is red (stop), green (go), amber (prepare to stop). In the UK and Canada, amber officially means 'stop' (unless it would cause an accident to do so), but in practice amber is treated as 'prepare to stop'. [Source: http://en.wikipedia.org/wiki/Traffic_light.]

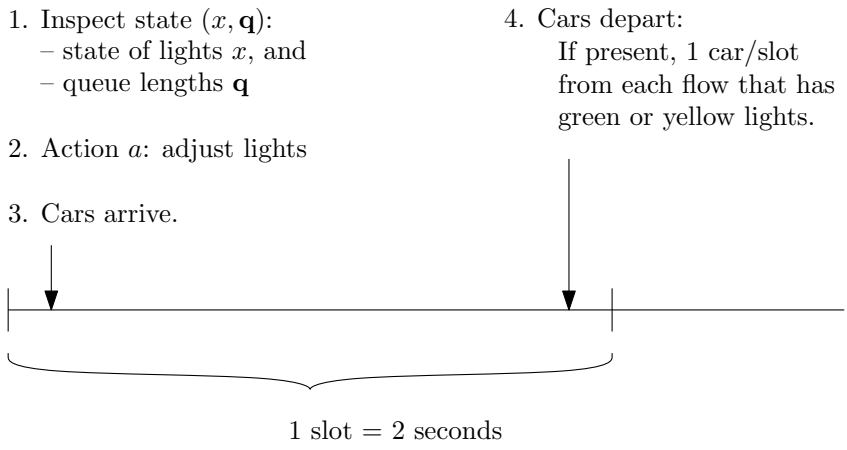


Figure 6.2: The queueing process: modeling the course of events in the MDP model

Workload

The workload that a flow brings to an intersection is the (long-run) fraction of time the light of flow f to serve all cars. Since a departure takes exactly 1 slot. The workload of flow f equals λ_f. The workload ρ_s that combination s brings to the intersection is the maximum of λ_f over all flows part of that combination:

$$\rho_s = \max_{f \in \mathcal{C}(s)} \lambda_f. \tag{6.1}$$

The workload of the intersection is the sum of the workload over all combinations:

$$\rho = \sum_{s=1}^{C} \rho_s = \sum_{s=1}^{C} \max_{f \in \mathcal{C}(s)} \lambda_f \,. \tag{6.2}$$

The workload ρ does not include the all-red time needed to clear an intersection.

We restrict our attention to under-saturated cases where queues are stable: $\rho < 1$.

Remark Note that the effective workload is in general higher than ρ, since the effective workload includes the fraction of time all lights are red for switching from green to one combination to green to another combination. For example, consider the F4C2 case, with $\lambda_f = 0.3$ for all flows. From (6.2) it follows that the workload is $\rho = 0.3 + 0.3 = 0.6$. When FC grants green for 3 slots to C_1 and for another 3 slots to C_2, then the cycle length equals 12 slots. During a single cycle on average $0.3 \cdot 12 = 3.6$ cars arrive at each queue. On average 7.2 slots in a complete cycle are actually used for and two slots are

used for switching between the two combinations to clear the intersection. The effective workload is thus $9.2/12 \approx 0.77$. The residual $12-9.2 = 2.8$ slots are used to deal with the uncertainty in car arrivals; since cars arrive at different flows independently some queues of a combination may be empty while cars depart from other queues of that combination.

Remark – Thickness of a combination – The workload ρ_f of a combination can be used to find an indication of how much green time a combination requires. In addition, we define the *'thickness' of combination* s: $\sum_{f \in \mathcal{C}(s)} \lambda_f$. The thickness of a combination is the mean number of cars that arrives at the intersection by flows in combination s. (The thickness of flow f is simply λ_f.

6.1.3 Outline

In the next section the problem is formulated as an MDP. For large intersections the number of possible queue states becomes tremendously large, prohibiting straightforward optimization of the MDP. Starting with a (nearly) optimal fixed cycle policy, a one-step policy improvement approach is proposed in Section 6.3. The approach is demonstrated and tested in Section 6.4 assuming cyclic control. Acyclic control and more advanced rules are introduced in Section 6.5 and tested in Section 6.6. Finally we report some conclusions in Section 6.7.

6.2 Formulation as a Markov Decision Problem

As argued the problem is modeled in discrete time. We refer to Figure 6.2 for the order in which we model the inspection of the state, the change of the lights, and the arrivals and departures of cars. The states are inspected at the beginning of a slot, next an action is selected and accordingly the lights are adjusted. Cars arrive at the beginning of a slot and may leave the very same slot, when the lights show green or yellow and no cars are waiting in front of it. At the end of a slot a number of cars have left their respective queues. Waiting costs are incurred over the number of cars present at the start of a slot; a car that arrives and passes the stopping line in the very same slot does not contribute to the cost over that slot since it has not experienced waiting time.

To formulate the MDP model, next the states, the decisions, the transitions and the costs are specified in detail.

6.2.1 States

The arrival and departure processes are assumed to be Markovian (i.e. memoryless), so the state of the traffic flows and related queues is described by the vector $\mathbf{q} = (q_1, \ldots, q_F)$, with q_f the number of cars in flow f present at the beginning of a slot. For the moment we do accept an infinite state space, in Section 6.2.6 we discuss the truncation of the queues in the MDP model.

The state space is completed by the state of the lights x. To allow both cyclic and acyclic control, we define x as a tuple (s, i), which indicate that the lights of combination s:

- are green when $i = 0$,
- are at the first yellow slot when $i = 1$,
- are at the second yellow slot when $i = 2$,
- are red as all other lights when $i = 3$.

Note that the 'all red' state $(s, i = 3)$ also marks that combination s was most-recently served. The total number of signal states x is thus $4C$. For acyclic control this number can be reduced to $3C + 1$, since then a single 'all red' state suffices.

The states at the start of a slot are thus denoted by the vector $(x, \mathbf{q}) = ((s, i), \mathbf{q})$.

6.2.2 Decisions

Decisions are taken at the beginning of a slot and executed instantaneously. Thus, if a decision grants green to a combination, cars of that combination can leave in the very same slot. The set of feasible decisions one may consider in state $((s,i),\mathbf{q})$ depends at least on the traffic light state (s,i) and maybe also on the number of queued cars $\mathbf{q}$.

- If the lights are green for combination s, i.e. if $i=0$, there are two feasible decisions:
  - keep the lights as they are, or
  - change from green to the first yellow slot to combination s.
- When the lights show yellow there is no choice, since the signaling process may not be interrupted:
  - At the end of a first yellow slot continue to the second yellow slot,
  - After the second yellow the only decision is to change into red for all flows.
- If all lights are red the feasible decisions are
  - to keep all lights red (if all queues are empty this might be optimal because switching takes 3 slots), or
  - to give green to one of the combinations. If the cyclic order is to be kept, we only allow to skip a combination if all queues for that combination are empty. When skipping a combination the cycle is resumed with the next non-empty combination: as a result of skipping combination 2 two successive green periods of combination 2 are far apart: 1, **2**, 3, 1, 3, 1, **2**, 3, etc.

The decision space in state $(x,\mathbf{q})=((s,i),\mathbf{q})$ is thus:

$$\mathcal{A}((s,i),\mathbf{q}) = \begin{cases} \{(s,0),(s,1)\} & \text{if } i=0 \quad \text{(lights of } s \text{ show green)}, \\ (s,i+1) & \text{if } i=1,2 \quad \text{(lights of } s \text{ show yellow)}, \\ \{(s,3),(s',0)\} & \text{if } i=3 \quad \begin{array}{l}\text{(all lights show red,} \\ s'\in\mathcal{S}\text{, but under cyclic control} \\ s'=\text{next non-empty combination)}.\end{array} \end{cases}$$

Remark – According to the given definition of $\mathcal{A}((s,i),\mathbf{q})$, the action space does not depend on $\mathbf{q}$. In the cases on which we report in this thesis, the action space does depend on the queue lengths as follows. To allow a fair comparison of the MDP policy against exhaustive control, we copy a special action that applies when no cars are present at the queues. Under exhaustive control, we suggest to freeze the signals when no cars are queued. The state of the lights (if not yellow) are frozen when the system is empty: green lights stay green and red lights stay red during the coming slot.

In some cases this may be optimal, but it is suboptimal in general. We do not bother about the optimality of this action, since, in the cases that we are interested in, it happens rarely that all queues are empty at the same time.

6.2.3 Transition probabilities

Action a implies an instantaneous change of the lights from state x into state a. If action a implies red to flow f, then the transition probability with respect to the queue length are

$$p_f(q_f, q_f|a) = 1 - \lambda_f, \quad \text{and} \quad p_f(q_f, q_f + 1|a) = \lambda_f. \tag{6.3}$$

If, instead, flow f receives, as a result of action a, green or yellow during the coming slot, the transition probabilities for the length of queue f are given by (with $y^+ = \max\{y, 0\}$):

$$p_f(q_f, (q_f - 1)^+|a) = 1 - \lambda_f, \quad \text{and} \quad p_f(q_f, q_f|a) = \lambda_f. \tag{6.4}$$

Note that in Equations (6.3) and (6.4) we have assumed that within a slot 0 or 1 car arrives at each queue, and, if present, 1 car departs from each queue whenever the lights shows green or yellow.

The transition probability from state $(x, \mathbf{q})$ to state $(a, \mathbf{q}')$, denoted by $p(x, \mathbf{q}; a, \mathbf{q}')$, is simply the product of the $p_f(q_f, q'_f|a)$'s:

$$p(x, \mathbf{q}; a, \mathbf{q}') = \prod_{f=1}^{F} p_f(q_f, q'_f|a) \ . \tag{6.5}$$

6.2.4 Criterion and costs

The criterion is the minimization of the long-run average waiting time per car. According to Little's law this is equivalent to minimizing the average number of cars waiting at all queues. Therefore we apply a linear cost structure with one unit of costs for every car present at the beginning of a slot. Let $c(\mathbf{q})$ denote the one slot costs or direct costs in state $(x, \mathbf{q})$, then

$$c(\mathbf{q}) = \sum_{f=1}^{F} q_f \ . \tag{6.6}$$

Note that waiting costs are incurred no matter the light shows green or yellow. The cost structure thus relates to the time spent in the queue assuming queued cars to have length 0. When cars have non-zero length and are queued horizontally, as in the next chapters, then waiting time (in slots) is measured also for queued cars that move forward one position per slot when the lights show green or yellow.

6.2.5 Successive approximations

In order to obtain an optimal policy, one may apply an SA algorithm. Therefore let $V_n(x, \mathbf{q})$ denote the expected minimal costs over an horizon of n slots when starting in state $(x, \mathbf{q})$. In principle the following DP relation holds between $V_{n+1}(x, \mathbf{q})$ and $V_n(\cdot, \cdot)$ for all states $(x, \mathbf{q})$:

$$V_{n+1}(x, \mathbf{q}) = c(\mathbf{q}) + \min_{a \in \mathcal{A}(x,\mathbf{q})} \sum_{\mathbf{q}'} p(x, \mathbf{q}; a, \mathbf{q}')\, V_n(a, \mathbf{q}') \ . \tag{6.7}$$

Since the state space is infinite, (6.7) cannot be used directly in an SA algorithm. Therefore the state space is truncated to a finite one. To compensate for any errors so-called externality costs are defined.

6.2.6 Reduction to a finite state space and Externality costs

In the MDP model we truncate each queue to a maximum of Q cars, such that all computations can be done in finite time. The bound on the queue length Q is preferably high enough such that overflow happens rarely. In the SA algorithm one thus only considers state vectors $\mathbf{q}$ with all elements q_f in $\{0, 1, \ldots, Q\}$.

In the evaluation of the transition from $\mathbf{q}$ to $\mathbf{q}'$, arrivals at queues that are full are rejected in the MDP model. In the real system cars cannot be rejected: q'_f may exceed Q for one or more queues f. When a car is not admitted at queue f in the MDP model, one needs to compensate for the waiting time and the additional delay the car would give to all other cars: already present or still to arrive in the real system. Therefore so-called *externality costs*, $E(a, \mathbf{q}')$, are included in the (expected) direct cost function.

Let $\mathbb{E}C(a, \mathbf{q})$ denote the expected cost to incur in state $(a, \mathbf{q})$. $\mathbb{E}C(a, \mathbf{q})$ is the sum of the direct waiting costs $c(\mathbf{q})$ and the expectation over the externality costs $E(a, \mathbf{q}')$:

$$\mathbb{E}C(a, \mathbf{q}) \equiv c(\mathbf{q}) + \sum_{\mathbf{q}'} p(x, \mathbf{q}; a, \mathbf{q}')\, E(a, \mathbf{q}'). \tag{6.8}$$

When the externality costs $E(a, \mathbf{q}')$ are known or approximated, one may apply successive approximations using Equation (6.9) for all states $(x, \mathbf{q})$ with $\mathbf{q} \leq Q \cdot \mathbf{1}$:

$$V_{n+1}(x, \mathbf{q}) = \min_{a \in \mathcal{A}(x, \mathbf{q})} \left(\mathbb{E}C(a, \mathbf{q}) + \sum_{\mathbf{q}'} p(x, \mathbf{q}; a, \mathbf{q}')\, V_n(a, \mathbf{q}^{\mathrm{T}}) \right), \tag{6.9}$$

where $q_f^{\mathrm{T}} \equiv \min\{q'_f, Q\}$.

A simple expression or approximation for the externality cost $E(a, \mathbf{q})$ is only available for some queueing models [61], but is not present for the complex queueing system that we are dealing with. A wrong approximation of the externality costs may affect the optimal strategy. On the one hand, when these costs are estimated too high, the optimal policy tries to stay away from the border Q and thus the systems tends to switch too early from green to yellow. On the other hand, when one underestimates the costs, it may be optimal to serve a queue that is 'full', only when all other queues are empty. We thus need a numerical scheme to (accurately) approximate the externality costs. A relatively easy way, which we explain below, is to approximate the costs by extrapolation of the value function in an SA algorithm.

Computation of externality costs by SA algorithm

Under an optimal policy for the stationary infinite horizon MDP, the externality costs are by definition of the value vector $\mathbf{V}_n$:

$$E(a, \mathbf{q}') = \lim_{n\to\infty} \left[V_n(a, \mathbf{q}') - V_n(a, \mathbf{q}^{\mathrm{T}}) \right], \tag{6.10}$$

where $\mathbf{q}^{\mathrm{T}}$ is the truncation of the vector $\mathbf{q}'$: $\mathbf{q}^{\mathrm{T}} \equiv (\min\{q'_1, Q\}, \ldots, \min\{q'_F, Q\})$. In addition we define the overflow, $\mathbf{q}^{\mathrm{O}} \equiv \mathbf{q}' - \mathbf{q}^{\mathrm{T}}$.

This definition suggest to approximate $E(a, \mathbf{q}')$ in iteration n of an SA algorithm by

$$E_n(a, \mathbf{q}') \equiv V_n(a, \mathbf{q}') - V_n(a, \mathbf{q}^{\mathrm{T}}) \tag{6.11}$$

Clearly, $E_n(a, \mathbf{q}')$ is 0, whenever all $q'_f \leq Q$. Since $V_n(a, \mathbf{q}')$ is unknown when q'_f exceeds Q for one or more flows f, $V_n(a, \mathbf{q}')$ is estimated by quadratic extrapolation based on the three values $V_n(a, \mathbf{q}^{\mathrm{T}} - 2\mathbf{q}^{\mathrm{O}})$, $V_n(a, \mathbf{q}^{\mathrm{T}} - \mathbf{q}^{\mathrm{O}})$, and $V_n(a, \mathbf{q}^{\mathrm{T}})$. In Appendix E one reads how to extrapolate multi-dimensional tabulated functions. Here we simply state the resulting formula for quadratic extrapolation of $\mathbf{V}$:

$$V_n(a, \mathbf{q}') \approx 3 \cdot V_n(a, \mathbf{q}^{\mathrm{T}}) - 3 \cdot V_n(a, \mathbf{q}^{\mathrm{T}} - \mathbf{q}^{\mathrm{O}}) + V_n(a, \mathbf{q}^{\mathrm{T}} - 2\mathbf{q}^{\mathrm{O}}), \tag{6.12}$$

Note that this extrapolation is feasible when $\mathbf{q}^{\mathrm{T}} - 2\mathbf{q}^{\mathrm{O}}$ contains no negative elements, i.e. when $\forall f : q_f^{\mathrm{O}} \leq \frac{1}{2}Q$). In the SA algorithm the overflow is at most 1 per flow, since at most 1 car arrives per flow. Furthermore, quadratic extrapolation works well, since the direct cost structure is linear in $\mathbf{q}$.

In (6.12) one reads the n-th approximation of the direct externality costs, $E_n(a, \mathbf{q}')$, when making a transition from $\mathbf{q}$ to $\mathbf{q}^{\mathrm{T}}$ instead to $\mathbf{q}'$:

$$E_n(a, \mathbf{q}') = 2 \cdot V_n(a, \mathbf{q}^{\mathrm{T}}) - 3 \cdot V_n(a, \mathbf{q}^{\mathrm{T}} - \mathbf{q}^{\mathrm{O}}) + V_n(a, \mathbf{q}^{\mathrm{T}} - 2\mathbf{q}^{\mathrm{O}}). \tag{6.13}$$

Since in the infinite horizon MDP, the direct cost must be stationary, the approximation of $E(a, \mathbf{q}')$ is fixed after a large number of, say M, iterations. For the dynamic control of traffic lights M is set to 100 slots (or more than 3 minutes in real time).

6.2.7 An SA algorithm with extrapolation of Externality costs

A complete successive approximation scheme to compute an optimal policy is the following:

Step 1. First, select a sufficiently large value of M, such that the externality costs are approximated over an horizon of sufficient length of M slots.
Next, for $n = 0$ to $M-1$ successively compute for all states $(x, \mathbf{q})$ with $\mathbf{q} \leq Q \cdot \mathbf{1}$:

$$V_{n+1}(x, \mathbf{q}) = c(\mathbf{q}) + \min_{a \in \mathcal{A}(x,\mathbf{q})} \left(\sum_{\mathbf{q}'} p(x, \mathbf{q}; a, \mathbf{q}') \ \left[E_n(a, \mathbf{q}') + V_n(a, \mathbf{q}^{\mathrm{T}}) \right] \right) \quad (6.14)$$

starting with $V_0(x, \mathbf{q}) = 0$, and $E_n(a, \mathbf{q}')$ as defined in (6.13).

Step 2. Store the resulting value vector $\mathbf{V}_M$ and compute the expected direct cost:

$$\mathbb{E}C(a, \mathbf{q}) = c(\mathbf{q}) + \sum_{\mathbf{q}'} p(x, \mathbf{q}; a, \mathbf{q}') \cdot E(a, \mathbf{q}') \quad (6.15)$$

using the following approximation of the externality costs

$$E(a, \mathbf{q}') \approx 2 \cdot V_M(a, \mathbf{q}^{\mathrm{T}}) - 3 \cdot V_M(a, \mathbf{q}^{\mathrm{T}} - \mathbf{q}^{\mathrm{O}}) + V_M(a, \mathbf{q}^{\mathrm{T}} - 2\mathbf{q}^{\mathrm{O}}). \quad (6.16)$$

Step 3. Starting with $n = M$ and $\mathbf{V}_M$ the value vector lastly computed in Step 1., recursively compute for all states $(x, \mathbf{q})$ with $\mathbf{q} \leq Q \cdot \mathbf{1}$:

$$V_{n+1}(x, \mathbf{q}) = \min_{a \in \mathcal{A}(x,\mathbf{q})} \left(\mathbb{E}C(a, \mathbf{q}) + \sum_{\mathbf{q}'} p(x, \mathbf{q}; a, \mathbf{q}') \ V_n(a, \mathbf{q}^{\mathrm{T}}) \right) \quad (6.17)$$

until the $span(\mathbf{V}_{n+1} - \mathbf{V}_n)$ is sufficiently small, say smaller than ϵ.

Suppose this happens at the N-th iteration. (One may show that convergence is guaranteed as the system empties every now and then, so that all states communicate.)

Step 4. If no optimizing actions are stored in Step 3., then a (nearly) optimal stationary strategy π is obtained from:

$$\pi(x, \mathbf{q}) = \arg \min_{a \in \mathcal{A}(x,\mathbf{q})} \left(\mathbb{E}C(a, \mathbf{q}) + \sum_{\mathbf{q}'} p(x, \mathbf{q}; a, \mathbf{q}') \ V_{N+1}(a, \mathbf{q}^{\mathrm{T}}) \right). \quad (6.18)$$

The average costs under π can be approximated by any element of $\mathbf{V}_{N+1} - \mathbf{V}_N$, which estimates the minimal long-run average number of cars in the system.

6.2.8 Computational complexity

Since we have made the state space finite, in principle the computations in (6.14) to (6.18) can be done in finite time. However, the multi-dimensionality of the system (one dimension per flow) usually leads to tremendous state spaces so that in general a straightforward dynamic programming approach is not possible in a reasonable amount of time. The computational complexity is primarily set by the number of queue states, which is exponential in the number of flows F. When no more than Q cars are admitted to each queue, then the number of queue states is $(1+Q)^F$. The number of traffic light states is $C \cdot (1+3)$ (under cyclic control).

When the workload of the intersection is not so high, a reasonable bound on the queue lengths might be 9 cars. Then, for an intersection with 12 flows, one has to consider thus 10^{12} queue states. When the flows are served in 4 combinations, the total number of states, including the $4 \cdot (1+3)$ traffic light states, thus becomes $1.6\,10^{13}$, which is far too many to solve the MDP in a reasonable time. Another limitation in implementing the successive approximation scheme may be the available RAM memory: for all states $(x, \mathbf{q})$ with $\mathbf{q} \leq Q \cdot \mathbf{1}$ at least the values of $V_n(x, \mathbf{q})$, $V_{n+1}(x, \mathbf{q})$ need to be stored.

Although we compensate in the MDP model for overflow of the queues, Q must be high enough such that overflow happens rarely. Furthermore, Q should be high enough since the SA algorithm derives in principle no optimal actions for states were one or more queues do overflow. When the overflow is limited to at most $\frac{1}{2}Q$, nearly-optimal actions can be derived by extrapolation of the value function using known values of the value vector. The quality of such estimates cannot be guaranteed.

6.3 One-step policy improvement (RV1)

Clearly the computational complexity of the MDP makes it practically impossible to solve many real-life MDPs. Therefore we suggest to apply a one-step policy improvement approach that is based on the decomposition of the state space. As outlined in Chapter 5, over the years such an approach has been applied successfully to a number of multi-dimensional decision problems (see [105], [165], [80], [81], [17], [18], and [124]).

Approach

The one-step policy improvement approach improves an initial policy for which the relative values are known. Under FC the relative values of the states can be determined after the decomposition of the state space. Therefore FC is taken as the initial strategy that will be improved through a one-step policy improvement algorithm. The following four steps are distinguished in our approach:

Step 1. Obtain a good FC:
Find a FC with cycle length D and effective green times d_s that results in a low overall long-run average waiting time per car. Therefore an incremental search algorithm is used in which the average waiting costs per car is computed flow-by-flow by evaluation of the underlying Markov chains. The Markov chain is thus decomposed into F periodic Markov chains, one for each flow.

Step 2. Derive the relative values of the D slots within the fixed cycle:
For each of the F Markov chains, one computes the relative values of the states under FC. Summing the relative values for all flows for given numbers of queued cars $\mathbf{q}$, one obtains the relative appreciation of each slot within the cycle.

Step 3. Set an improvement rule (RV1):
Formulate a one-step policy improvement rule using the relative values of the states under FC. We call the resulting policy RV1, with RV for relative value.

Step 4. Evaluate RV1 by Simulation:
Finally, RV1 is evaluated by simulation, and compared to a number of basic traffic light control strategies and, where possible, to the optimal MDP policy.

The four steps are discussed in greater detail in the next subsections. The simulation results of Step 4 are reported in Section 6.4.

6.3.1 Step 1 – Fixed Cycle control (FC)

The reason for choosing FC as initial control strategy is that the relative values of states under FC can be computed flow-by-flow. In the next subsection the numerical computation of the relative values is discussed for a single flow. In this section we focus on finding a good configuration of FC that results in a low average waiting time per car.

The Optimal-fixed-cycle algorithm

At this point we are just interested in setting a good or (nearly) optimal fixed cycle. For this we use a simple incremental search algorithm: the *Optimal-fixed-cycle algorithm*, which is reported in detail in Appendix D.2. The search starts with calling the *Minimum-cycle-length algorithm* to find a cycle of minimum length by which all queues are just stable under FC: all queues get assigned just enough green and yellow slots to be stable. The Minimum-cycle-length algorithm is reported in Appendix D.1. A configuration of FC is evaluated by numerically solving a set of F Markov chains. How each Markov chain (MC) is formulated and how it is solved is discussed in the next section.

The Optimal-fixed-cycle algorithm increases successively the cycle length D by 1 slot, according to the best increase of the effective green period for one of the combinations. This process of incremental search is stopped when the last-so-many increases did not yield a better FC. The best FC so far is considered as the (locally) optimal FC.

Numerical example

In Figure 6.3, the search process of the Optimal-fixed-cycle algorithm is illustrated for the F12C4 intersection of Figure 6.1(b)) with identical arrival rates ($\forall f : \lambda_f = 0.2$). The Minimum-cycle-length algorithm as reported in Appendix D.1 indicates that the cycle length must be at least 24 slots: during 4 different slots all lights show red. The effective green time is 5 slots for each combinations: the lights show green for 3 slots and yellow for 2 slots.

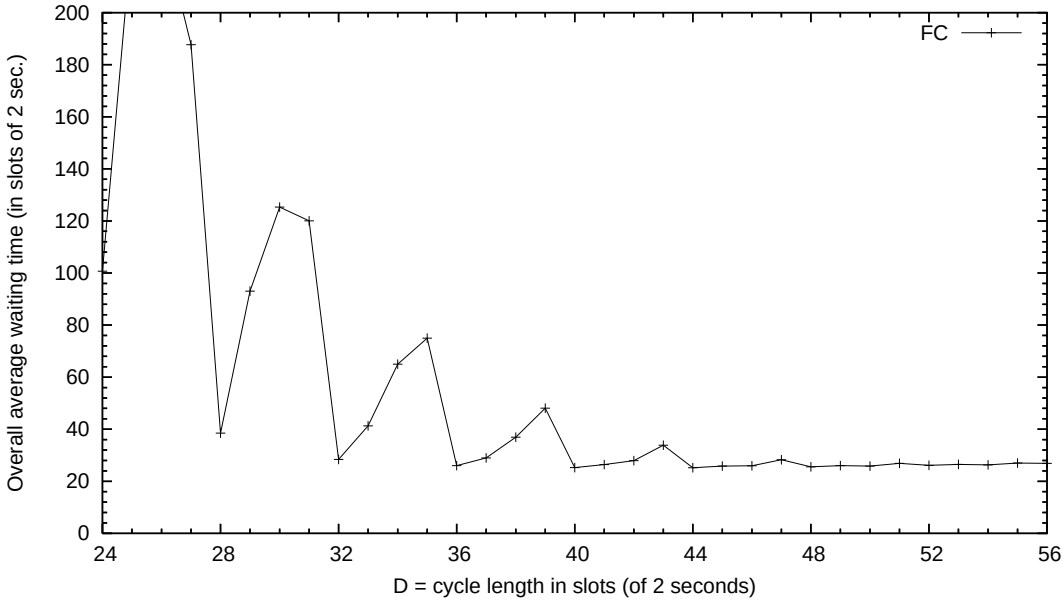
The Optimal-fixed-cycle algorithm successively increases the cycle length, starting with a cycle of length $D = 24$ slots. The length of the green period for a single combination is extended by one slot, such that the cycle length becomes 25 slots. Consequently, the red period for all other combinations increased by 1 slot, which may cause some queues to become instable. Whereas all queues are (just) stable when $D = 24$, some queues are

instable at D is 25, 26, and 27 slots[2]. At $D = 28$, the best FC is the one where the green periods of all four combinations are extended by 1 slot compared to the FC with $D = 24$.

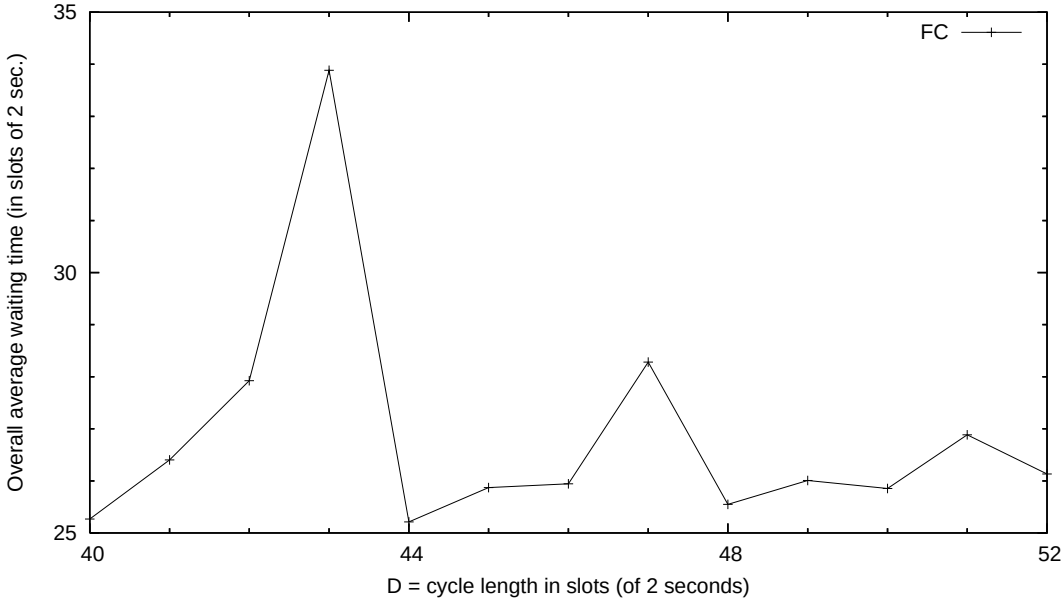
The long-run average waiting times reported in Figure 6.3 are for the best FC given cycle length D. The figures are accurate up to 4-5 decimals and obtained by numerically solving MCs. Clearly the response curve, depicting the relation between the long-run overall average waiting time and the cycle length, is far from convex and contains many local optima. When one would only connect the local optima in Figure 6.3(a) one obtains a curve that is quite flat for high values of D. Apparently it is better to end-up with a FC with a relatively high cycle length, than with a FC with a cycle length that is relatively short. In Figure 6.3(b) we have zoomed to show that cycle length 40 and 44 give the best results, although 52 slots still yields a low average waiting time. We deliberate on the sensitivity with respect to the chosen cycle length in Section 6.4.5.

Since in this example the arrival rates are identical, a locally optimal FC appears to have green periods of identical length. Locally optimal FCs are thus found for D being a multiple of $C = 4$. This is not the case when the arrival rates differ, as will be illustrated in Figure 8.12 in Chapter 8. This complicates the stopping criterion of the search algorithm. The incremental search algorithm can be stopped when k successive increments do not yield an improved FC. When all arrival rates are identical, k can be set to C. But, when some combination(s) have a higher workload than others, it may be necessary to set k higher depending on the skewness in the arrival rates. We do not guarantee the algorithm to end in the global optimum, but numerical experiments have shown that the Optimal-fixed-cycle algorithm generates a good FC.

[2]In the evaluation of a Markov chain for an instable queue the iterative process is cut-off at some maximum number of iterations.



(a) Sensitivity of the waiting time under FC when the cycle length ranges from 36 to 56 slots.



(b) Sensitivity around the optimal value of the cycle length.

Figure 6.3: Application of the Optimal-fixed-cycle algorithm to find optimal value of the cycle length for F12C4 with identical arrival rates ($\lambda_f = 0.2$). (The average waiting times are computed by solving a Markov chain for each flow. For an instable queue the iterative process is cut-off at some maximum number of iterations.)

6.3.2 Step 2 – Relative values of FC

Under FC each flow perceives a periodic green-yellow-red cycle of fixed length, independent of the queue lengths of the various flows. Let D be the duration of a cycle or the *cycle length*. Then the *effective green time* or the *departure time* of flow f is, under FC, a fixed number of d_f slots during which the lights of flow f show green or yellow. The last yellow slot is followed by $r_f = D - d_f$ red slots. Since cars depart from a queue only during green or yellow slots, we call these d_f slots *departure slots*.

Thus for flow f the fixed cycle results in a periodic MC with d_f departure slots and r_f red slots. The state of the chain can be described as a pair (t, q) with q the number of cars present at the beginning of a slot and t the number of the slot within the cycle, $t \in \{1, \ldots, D\}$. Whereas in the MDP model the traffic light state was $x = (s, i)$, the state of the lights under FC is simply the slot number t.

The equilibrium distribution, and thus the average costs per time unit, can be computed using a generating function approach, see Darroch [33] and Van Leeuwaarden [153]. But what we need are the relative values of the states to compare some time slot τ against slot t. The analysis of such a periodic MC and in particular the computation of these relative values can be done numerically by stochastic dynamic programming (SDP). As the state space for one flow is small (only 2-dimensional), one may set the bound on the queue length Q high enough such that the probability of rejecting an arrival plays hardly any role.

The total number of feasible states in each of the F MCs is $D{\cdot}(1{+}Q)$, which is considerably less than the number of states in the MDP model. Hence the relative values of states under FC can be computed quickly by value iteration. The immediate costs in state (t, q) is simply the number of cars present: q. In addition, one may incur expected externality costs to compensate for the error involved with the truncation of the queue lengths at Q. Although externality costs are computed in the programs developed, we omit the computation of any externality cost in the discussion below: when Q is set very high, such that the number of queued cars in the real systems stays virtually always well below Q, externality costs play hardly any role.

Let $v_n^f(t, q)$ be the total expected costs (=number of cars waiting) at queue f over n slots when starting in (t, q). For the evaluation of the MC for a single flow and for the computation of the relative values of states, one proceed as follows.

Computation of relative values under FC

For all flows $f = 1, \ldots, F$ execute the following four steps: Step 2a – Step 2d.

Step 2a. Define $v_0^f(t,q) = 0$ for all t and q and let ϵ take a very small value, e.g. 10^{-10}.

Step 2b. For all t and q recursively compute $v_{n+1}^f(t,q)$ as follows:

- if t is a departure slot for f:

$$v_{n+1}^f(t,q) = q + \lambda_f \cdot v_n^f(t+1,q) + (1-\lambda_f) \cdot v_n^f(t+1,(q-1)^+), \quad (6.19)$$

- if t is a 'red slot' to flow f:

$$v_{n+1}^f(t,q) = q + \lambda_f \cdot v_n^f(t+1,q+1) + (1-\lambda_f) \cdot v_n^f(t+1,q) \ , \quad (6.20)$$

where $v_n^f(D+1,\cdot)$ should be read as $v_n^f(1,\cdot)$.

until $span(\mathbf{v}_{n+D}^f - \mathbf{v}_n^f) < \epsilon$. Suppose this happens at first at the N-th iteration. For all t and q the difference $(v_{n+D}^f(t,q) - v_n^f(t,q))$ converges exponentially fast to the long-run average costs per cycle $(g^{(f)})$, as the D-step Markov chains are all irreducible.

Step 2c. Compute the long-run average waiting time (EW_f) at queue f in slots by applying Little's law: $EW_f = EL_f/\lambda_f$, with $EL_f =$ the long-run average number of cars present at the start of a slot:

$$EL_f = \frac{(v_{N+D}^f(t,q) - v_N^f(t,q))}{D}.$$

The overall average waiting time (EW) is simply the weighted sum:

$$EW = \frac{\sum_f \lambda_f \cdot EW_f}{\sum_f \lambda_f} = \frac{\sum_f EL_f}{\sum_f \lambda_f} = \frac{\sum_f (v_{N+D}^f(t,q) - v_N^f(t,q))/D}{\sum_f \lambda_f} \quad (6.21)$$

Step 2d. Choose a fixed reference state, $(D, 0)$, and compute the relative value vector $\mathbf{v}_{rel}^f$:

$$\mathbf{v}_{rel}^f \equiv \frac{\mathbf{v}_N^f + \cdots + \mathbf{v}_{N+D-1}^f}{D} - \frac{v_N^f(D,0) + \cdots + v_{N+D-1}^f(D,0)}{D} \cdot \mathbf{1} \ , \quad (6.22)$$

where $\mathbf{1}$ is a vector with all elements equal to 1.

Given the reference state $(D, 0)$, the relative values are unique. As discussed in Section 1.3.2 and Appendix B, the relative value definition in (6.22) is correct. The differences $v_N^f(q,t) - v_N^f(D,0)$ cannot be used for this purpose as FC is a periodic policy and consequently these differences change periodically with N.

Example – Consider the F4C2 intersection of Figure 6.1(a) where 4 flows are served in two combinations: flows 1 and 3 constitute C_1, and flows 2 and 4 constitute C_2. Suppose both combinations get green during 3 slots. Switching takes another 3 slots, so the duration of a cycle is 12 slots: $D = 12$. A typical relative value curve of flow 1 is depicted in Figure 6.4 for the case that 4 cars are waiting at queue 1. Slots 1 to 5 are departure slots for flow 1, the next 7 slots the signals of flow 1 show red. The worst position in the cycle is the start of slot 6, since all 4 cars have to wait for at least 7 slots. Apparently, the best position is the start of slot 1, because then the remaining green period last longest.

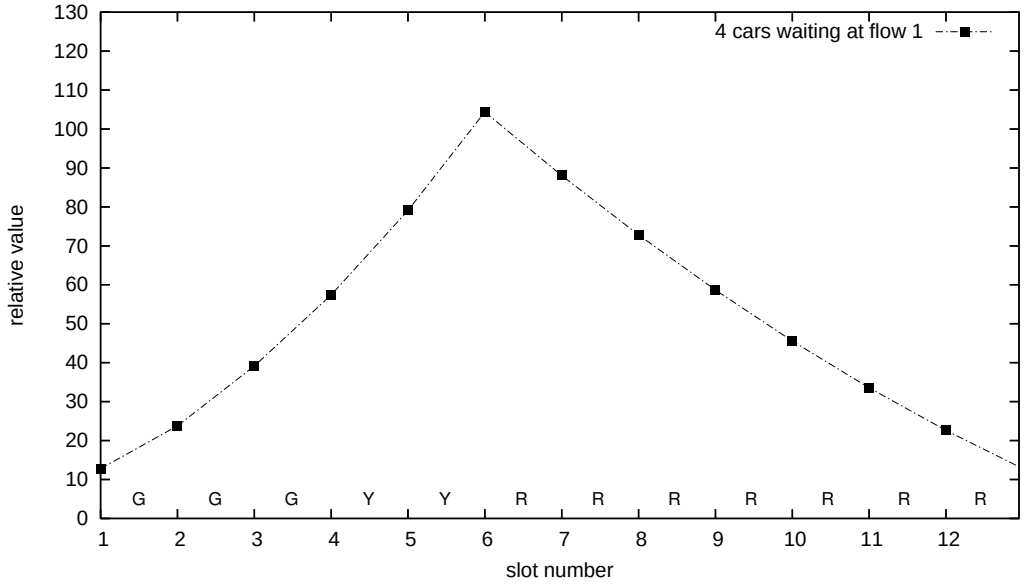


Figure 6.4: Relative value curve when 4 cars are waiting at flow 1.

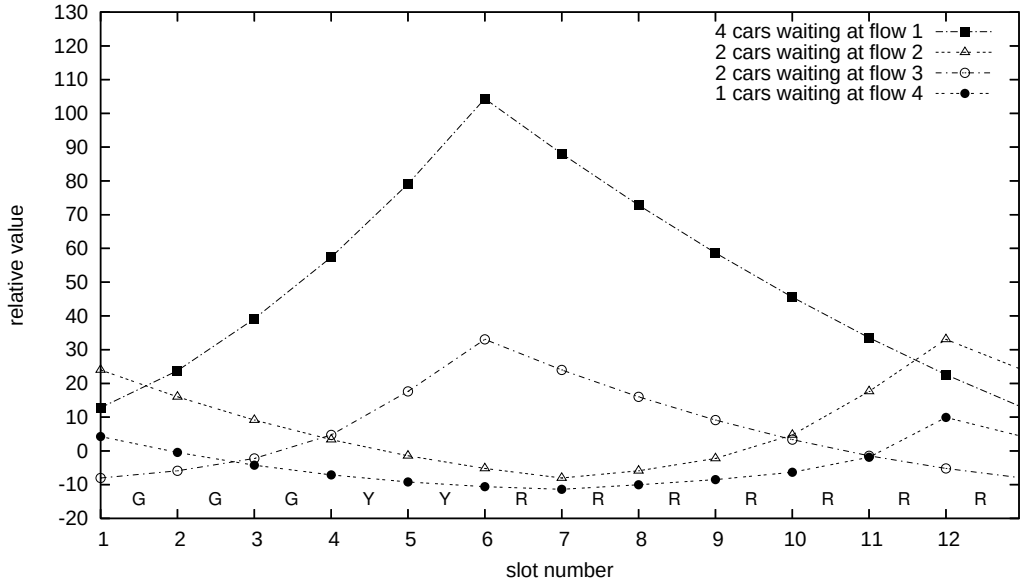


Figure 6.5: Relative value curves for flows 1 to 4 when $\mathbf{q} = (4, 2, 2, 1)$ cars are waiting.

Similar curves can be drawn for all flows and any number of cars in $\{0, 1, \ldots, Q\}$. In Figure 6.5 the relative value curves for all four flows are drawn when $\mathbf{q} = (4, 2, 2, 1)$ cars are waiting. Each flow has it own optimal slot number, which is usually at (or close to) the start of its green period. The curves of flow 2 and flow 4 show the difference in relative values between having 2 cars queued and only 1 car queued.

6.3.3 Step 3 – One-step policy improvement rule (RV1)

Now let us return to the system as a whole. Under FC the system's state is described by the pair $(t, \mathbf{q})$, with $\mathbf{q} = (q_1, \ldots, q_F)$ the numbers of queued cars and t the slot within the cycle. Related to t are the colors of the lights. In the example of the previous section, $t = 1$ is the first green slot of C_1, $t = 4$ is the first yellow slot of C_1, and $t = 7$ is the first green slot to C_2. Note that $t = 6$ and $t = 12$ relate to an all-lights-red slot. Slot 13 does not exist, since after slot $t = D = 12$ the cycle resumes at $t = 1$. During any slot t at most one combination has green or yellow, all other combinations have red.

Under FC the slots are thus visited in the order $1, 2, \ldots, D,\ 1, 2, \ldots$. When this order is interrupted, the state and color of the lights are changed accordingly. Now, let us return to the numerical example. Suppose $t = 2$ at the start of a slot (hence C_1 has right of way), then one may change the state of the light t from 2 into 4, such that the green period of C_1 is ended. In fact by allowing the dynamic adjustment of the state of the traffic lights

t, one may choose to resume FC at some slot τ rather than at slot t. The decision that one selects immediately affects the traffic lights and has indirectly impact on the number of cars waiting at each queue.

The one-step policy improvement rule looks for the best time jump within the cycle from t to τ. In state $(t, \mathbf{q})$, only time jumps to a slot $\tau \in \mathcal{T}(t, \mathbf{q})$ are allowed. The set $\mathcal{T}(t, \mathbf{q})$ contains all feasible time jumps when $\mathbf{q}$ cars are waiting at the start of slot t. $\mathcal{T}^{\mathrm{C}}(t, \mathbf{q})$ denotes the action space when the policy has to be cyclic. The action spaces are similar to $\mathcal{A}(x, \mathbf{q})$ in the MDP model. When t is a green slot to C^s, one may choose any τ that keeps the lights green to C^s, or to the τ that indicates the first yellow slot to C^s. The action space consists of a single element, i.e. t, when t refers to the start of the second yellow slot for some flow or to the start of an all-red slot. When t indicates that a new green period is about to start for combination s, one has to distinguish cyclic from acyclic control:

- Under cyclic control any τ that grants green to combination s, or set τ to $t-1$ such that the all-red period is extended by 1 slot;
- under acyclic control any τ that grants green to any combination may be selected, next to slots that imply red to all flows.

Selection of best time jump

In state $(t, \mathbf{q})$ every $\tau \in \mathcal{T}(t, \mathbf{q})$ is evaluated by summing the relative values related to τ: $\sum_{f=1}^{F} v_{rel}^{f}(\tau, q_f)$. Immediately thereafter, the lights are adjusted according to the value of τ that minimizes the sum of the relative values under FC. When evaluating a time jump, one assumes that after the jump FC is resumed at slot τ, although this will not be the case when decisions are revised at the start of every slot.

Hence the one-step optimal decision (time jump) in state $(t, \mathbf{q})$, follows from:

$$\sigma(t, \mathbf{q}) = \arg \min_{\tau \in \mathcal{T}(t, \mathbf{q})} \sum_{f=1}^{F} v_{rel}^{f}(\tau, q_f) \ . \tag{6.23}$$

These $\sigma(t, \mathbf{q})$ for all states $(t, \mathbf{q})$ together constitute the improved strategy RV1 that is hopefully close to optimal. Since a decision is updated every slot, RV1 breaks FC: the green periods under RV1 are not of a fixed length, but their lengths change dynamically.

Example – Revisit the previous example, in Section 6.3.2. To illustrate the concept, we show in Figure 6.6 the sum of the relative value curves over all flows with $\mathbf{q} = (4, 2, 2, 1)$ cars are waiting at flows 1 – 4. The best position in the cycle (yielding the lowest sum of relative values) is slot 1: the start of green to C_1. Although switching to this point seems to be best, it is not reachable from all points in the cycle. When C_2 has green the best feasible decision is to switch from green to yellow, since the curve has a local minimum at slot 10.

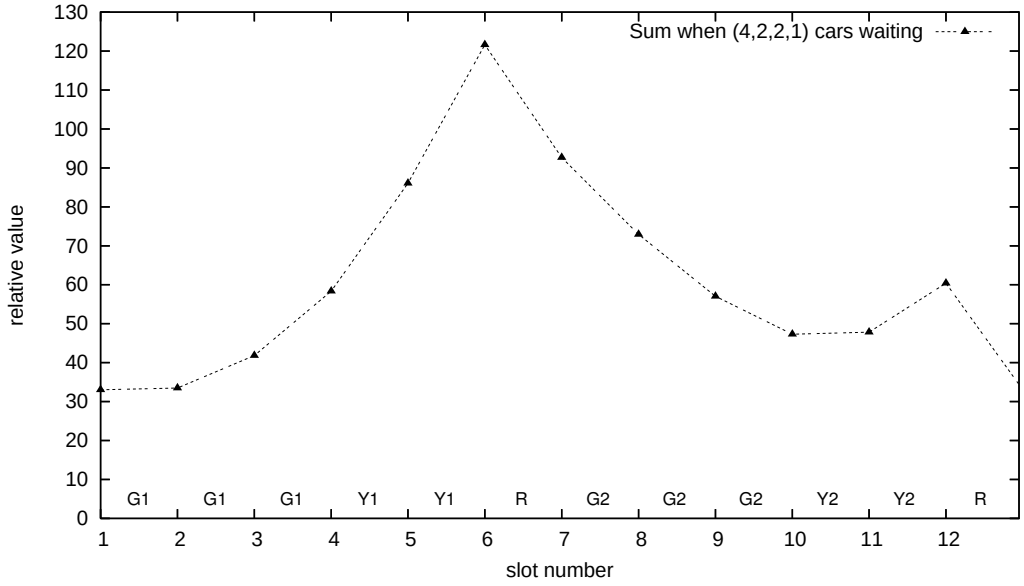


Figure 6.6: Sum of the relative value curves when $\mathbf{q} = (4, 2, 2, 1)$ cars are waiting at flow 1 to 4.

6.3.4 Step 4 – Simulation of RV1

To check the performance of RV1 one may in principle formulate a MC. For most intersections, the MC of RV1 cannot be analyzed, since the number of states $(t, \mathbf{q}) = (t;\, q_1, q_2, \ldots, q_F)$ is far too large. Since the actions and transitions do depend on the entire state description, the state space under RV1 cannot be decomposed; hence the flows cannot be analyzed in isolation of each other.

For large problems we can only evaluate RV1 by Simulation. Therefore a simulation model is developed and implemented in Delphi-Pascal. The process of cars arriving and leaving the queues and the control of the traffic lights is simulated in discrete time with time jumps equal to one slot (=2 seconds). At the start of every slot the number of cars

waiting at the queues ($\mathbf{q}$) is observed as well as the state of the traffic lights (t). Next, a decision τ is taken according to the policy under consideration, e.g. FC, RV1, or any exhaustive control policy. (Under FC $\tau = t$, under RV1 $\tau = \sigma(t, \mathbf{q})$.)

The decision τ is implemented instantaneously. Any cars arriving during a slot are assumed to do so at the beginning of that slot, any departures from a queue are modeled at the end of that slot. A car arriving at an empty queue and facing a green or yellow light passes the stopping line during the very same slot and hence contributes no waiting costs. At the start of the next slot the state of the traffic light is $\tau + 1$, and the above process repeats based on the new state of the queues.

Before the start of the simulation of RV1, the relative value curves for each flow and for all possible queue lengths are computed and stored. This takes hardly any time compared to solving the MDP, since the F Markov chains are of a much lower dimension: each MC has only 2 dimensions under FC compared to $F + 1$ dimensions for the MDP. When the simulation is started, every decision epoch the F relative value curves are summed based on the actual state $(t, \mathbf{q})$, as in Figure 6.6. Although based on FC, the improved policy RV1 is no longer a fixed cycle, since the green periods are interrupted (lengthened or shortened) every slot.

Remark – Since the relative values are computed flow-by-flow, the computation time remains low even when the upper bound Q on the queue lengths is very high: say 100 cars or even more. In the simulation model we assume that queues may contain an infinite number of cars. Although not likely when Q is set high enough, it may happen, at least theoretically, that the number of cars waiting at a queue exceeds Q. In that case, we need to estimate the relative value of that state since the relative value curves are known only for queues lengths $q \leq 100$. Then the relative value is estimated by quadratic extrapolation, see Appendix E.

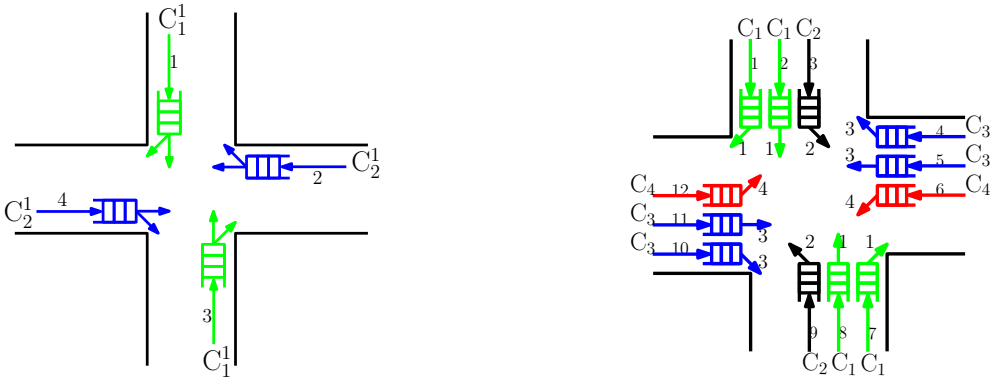
6.4 Test cases and results

To assess the quality of RV1, its performance is compared against FC, and some exhaustive control policies. For small problems the optimal MDP policy can be evaluated. Throughout this section we stick to cyclic control, hence $\mathcal{T}(t, \mathbf{q})$ in Equation (6.23) should be replaced by the action space for cyclic control $\mathcal{T}^{\mathrm{C}}(t, \mathbf{q})$. Acyclic control is studied in Sections 6.5 and 6.6.

6.4.1 Test cases – Infrastructures

Throughout this and the next chapters, the different policies are applied to two infrastructures: the F4C2 and F12C4 intersections that are displayed in Figure 6.7. These intersection were already introduced in Section 5.1.1. Whereas the F4C2 intersection may be realistic but simple, the F12C4 intersection has some interesting features. F12C4 has separate lanes for left turning traffic and non-identical numbers of flows per combination (C_1 and C_3 consists of twice as many flows as C_2 and C_4). Other infrastructures are studied in [54] and [55]. Instead of reporting results for many infrastructures, we believe its is at least as insightful to answer specific question for the F4C2 and F12C4 intersections under different workloads.

For each infrastructure, we consider different scenarios concerning the workload of the intersection and the symmetry in the arrival rates. In Table 6.1, we introduce the following terminology: *fully-(a)symmetric* and *partly-(a)symmetric*. In a, what-we-call, *'symmetric' case*, all combinations consist of an identical number of flows. F4C2 is thus a symmetric



(a) "F4C2" serve 4 flows in 2 symmetric combinations

(b) "F12C4" serve 12 flows in 4 asymmetric combinations

Figure 6.7: Two typical infrastructures.

Table 6.1: Terminology regarding the symmetry of test case with respect to the number of flows per combination and the arrival rates.

	# flows per combination	
	Identical for all C_s	Non-identical for all C_s
Arrival rates	'symmetric' intersection	'asymmetric' intersection
Identical $\forall f : \lambda_f = \lambda$	'fully-symmetric' case	'partly-asymmetric' case
Non-identical	'partly-symmetric' case	'fully-asymmetric' case

case, whereas F12C4 is asymmetric. When, in addition, the arrival rates in a symmetric case are identical for all flows, we deal with a *fully-symmetric case*; whereas when the arrival rates differ between flows, we speak of a *partly-symmetric case*. Asymmetric cases with identical arrival rates are called *partly-asymmetric cases*. When the arrival rates differ between flows, we call an asymmetric case a *fully-asymmetric case*.

For each infrastructure we formulate test cases in which the arrival rates are identical for all flows. In addition, we investigate a few cases with non-identical arrival rates, to find out whether 'thin' combinations are overruled by 'thick' combinations, given the definition of thickness in Section 6.1.2.

In the next section we report results for the two infrastructures under different work loads ρ (=0.4, 0.6, and 0.8), as defined in (6.2). Note that, as explained in Section 6.1.2, the effective workload is higher than ρ, since definition (6.2) does not include the all-lights-red time when switching to another combination. We do not consider cases where the effective workload is 1 or higher.

Simulation horizon

The accuracy of the results presented in the next subsections is primarily set by the number of simulation runs and the length of each run. All reported simulation results are based on 100 runs of 72,000 slots per run, corresponding to 4,000 hours in the real system. At the start of each run a warming-up period of 450 slots (=15 minutes) applies. The reported mean waiting times are accurate up to (at least) 2 to 3 figures; to safe space we do not report confidence intervals.

6.4.2 Simple intersection F4C2

Identical arrival rates (F4C2)

In Table 6.2 the results are presented for the fully-symmetric F4C2 intersection at varying workloads. In the cases $\rho = 0.4, 0.6, 0.8$ all flows have identical arrival probabilities per slot of 0.2, 0.3, 0.4 respectively. The Optimize-fixed-cycle algorithm suggests cycle lengths of 8, 12 and 22 slots respectively, as reported in seconds in the next-to-last row. The departure times are the lengths of the effective green period under FC. RV1 is based on FC with these cycle lengths and departure times.

Table 6.2: Overall mean waiting time (in sec.) for the fully-symmetric F4C2.

Rule	$\rho = 0.4$		$\rho = 0.6$		$\rho = 0.8$	
RV1	5.06		7.01		14.2	
FC	5.43	+7%	8.27	+18%	17.0	+20%
XC	5.76	+14%	8.82	+26%	19.9	+40%
XC-1	5.03	-1%	7.21	+3%	15.5	+9%
XC-2	5.09	+1%	7.31	+4%	14.2	+0%
MDP cyclic	4.89	-3%	6.95	-1%	13.5	-5%
FC cycle length (in sec.) and departure times per combination	16 (6, 6)		24 (10, 10)		44 (20, 20)	

The performance of the cyclic RV1 strategy obtained by the one-step policy improvement algorithm, is close to that of the optimal cyclic MDP strategy. Next to the average waiting times, we report the relative difference compared to RV1. The cyclic MDP policy performs only a few percent better. FC and XC yield on average 15% and 27% higher waiting times; when the load is high ($\rho = 0.8$) the differences are greatest. On average RV1 is just a little bit better than the anticipating exhaustive variants (XC-1 and XC-2), but the difference is small for this simple fully-symmetric case.

Non-identical arrival rates (F4C2)

Two F4C2 cases with non-identical arrival rates are considered:

Case I. The flows within the same combination have identical arrival rates, but C_2 is three times as thick as C_1: $\lambda_1 = \lambda_3 = \lambda$ and $\lambda_2 = \lambda_4 = 3\lambda$. (Hence $\rho_2 = 3\rho_1$.)

Case II. Flow 1 is a third as thick as the other flows: $\lambda_1 = \lambda/3$ and $\lambda_2 = \lambda_3 = \lambda_4 = \lambda$. (Hence $\rho_2 = \rho_1$.)

Table 6.3: Mean waiting times (in sec.) for two partly-symmetric F4C2 cases at $\rho = 0.6$.

Rule	EW	% above RV1	EW_1	EW_2	EW_3	EW_4
Case I						
$(\lambda_1, \ldots, \lambda_F) = (0.15, 0.45, 0.15, 0.45)$						
RV1	5.9		10.5	4.4	10.4	4.4
FC departure times 6, 14 sec.	6.9	+17%	11.2	5.4	11.2	5.4
XC	7.5	+27%	11.0	6.3	11.0	6.3
XC-1	6.6	+12%	8.6	5.9	8.6	5.9
XC-2	7.3	+24%	7.7	7.2	7.7	7.2
MDP (cyclic)	5.9	-0%	10.4	4.4	10.4	4.4
Case II						
$(\lambda_1, \ldots, \lambda_F) = (0.1, 0.3, 0.3, 0.3)$						
RV1	6.5		6.1	5.6	8.3	5.7
FC departure times 10, 10 sec.	8.0	+23%	5.2	8.3	8.3	8.3
XC	7.7	+18%	6.8	7.4	8.7	7.4
XC-1	6.5	+0%	5.4	6.2	7.3	6.2
XC-2	6.7	+3%	4.7	6.7	7.6	6.7
MDP (cyclic)	6.3	-3%	5.2	5.9	7.5	5.9

The results are reported in Table 6.3 for a workload of $\rho = 0.6$ (hence for Case I $\lambda = 0.45$ and for Case II holds $\lambda = 0.3$). The average waiting time, EW_f, is presented in the last four columns for each flow f. The overall average waiting time per car is reported in the second column. Again, next to the overall average waiting time per car, the relative differences compared to RV1 are reported. FC yields a waiting time that is for Case I and II respectively 17% and 23% above that under RV1. The overall average waiting time under all policies has reduced in both cases compared to the fully-symmetric case.

The impact of the arrival rates on the best configuration of FC, is read through the reported departure times: in Case I the cycle length remains $6 + 14 + 2 \cdot 2 = 24$ seconds. In Case II both the cycle length and the departure times are unaffected, since both combinations are equally thick.

By comparing the results for the two cases, we observe that flows that are part of a thicker combination experience a lower waiting time. This might seem counterintuitive, but a thicker combination gets a better treatment since it puts more weight to the scale. If a thin flow and a thick flow are together in one combination, then the thin flow benefits but the thick flow suffers a bit compared to when it would be part of a thicker combination.

6.4.3 Computation times on F4C2: MDP versus RV1

The computation complexity of the SA algorithm for computing an optimal MDP strategy is $\mathcal{O}(C^2(1+Q)^{2F})$. The complexity order is much lower for evaluating the Markov chains under FC, which is input to RV1. The computation of the relative values for all F flows is $\mathcal{O}(F \cdot D \cdot (1+Q)^2)$.

To illustrate the difference in complexity order, we report some running times on an in-2007-modern PC, an Intel Pentium 2.8GHz processor and 2Gb RAM- memory. The RAM-memory is sufficient to registers about 10^8 states. Depending on the implementation and the available time one can execute a successive approximation scheme for cases with millions of states. We discuss the running time for a simple example using the program, written in Delphi-Pascal, developed to generate results that are included in this thesis.

Consider the F4C2 intersection. When the load of the intersection is high, say 80%, the number of waiting cars virtually never exceeds 18, as may be checked by simulation. Hence we truncate queues at $Q = 18$ cars. The total number of states is about a million (or to be precise $(1+18)^4 \cdot (2 \cdot (1+3)) = 1,042,568$). The computation of the optimal cyclic MDP policy takes about 12 hours for 520 iterations, or 86 seconds per iteration of the SA algorithm. The computation time can be reduced significantly, by setting the upper bound Q somewhat lower. This is only possible because we accurately compute the externality costs by quadratic extrapolation. Further, one may note that a nearly optimal MDP policy is found far before iteration 520, but the SA algorithm continues iterating until the gain of the related Markov chain is closely approximated.

Computing for all flows the relative values of all states under FC takes only a few seconds, depending on the required accuracy. The computation time needed to obtain a nearly optimal MDP policy is thus much higher for a problem of the same size.

6.4.4 Complex intersection F12C4

For the F12C4 intersection the optimal MDP strategy cannot be computed because the number of states is prohibitively large. Even when in the MDP model queues are truncated at 2 cars, the total number of states is still very large: $8.5\,10^6$ states ($= (1+2)^{12}$ queue states times 16 traffic light states). F12C4 is not only computationally more complex because of the number of states, but also because the combinations are asymmetric: C_1 and C_3 consist of 4 flows, whereas C_2 and C_4 consists of two flows only. It seems to be more profitable to under-serve combinations C_2 and C_4, since serving the other two combination results in more departures per time slot as long as cars are present at the respective queues.

The asymmetry makes it more difficult to define simple rules that perform well. We keep the same definition of exhaustive control: all queues must be empty before the green signals are turned into yellow. Under XC-2 (and XA-2), green lights are turned into red as soon as at each of the flows that have right of way 2 or less cars are queued.

We consider both a partly-asymmetric and a fully-asymmetric F12C4 case. For the partly-asymmetric case (where all arrival rates are identical), we discuss consecutively:

- the impact of the asymmetry and the workload on the overall average waiting time, and
- how the asymmetry of the combination affects the waiting time distribution and the queue length distribution.

For the fully-asymmetric F12C4 case (with non-identical arrival rates), we test whether a very thin combination is getting under-served under a high workload. Therefore we assume that C_2 is 6 times as thin as C_1 and C_3, and three times as thin as C_4.

In Section 6.4.5, we discuss the impact of the sub-optimal cycle lengths D on the performance of FC and RV1.

Identical arrival rates (partly-asymmetric F12C4)

In Table 6.4 the results for varying workloads at a F12C4 intersection are presented for the case where all flows have identical arrival intensities (0.1, 0.15, and 0.2 for $\rho = 0.4$, 0.6 and 0.8 respectively). RV1 outperforms all other strategies. When the workload of the intersection is low (0.4) XC-2 performs equally well. At a high load XC-2 is too simplistic. At $\rho = 0.8$ XC-2 yields an average waiting time that is 28% higher than under RV1. Then FC performs even better than XC-2.

Table 6.4: Overall mean waiting time (in sec.) at different loads for F12C4 ($\lambda_f = \rho/4$).

Rule	$\rho = 0.4$		$\rho = 0.6$		$\rho = 0.8$	
RV1	13.5		19.3		41.8	
FC	15.0	+11%	23.7	+23%	50.5	+21%
XC	19.2	+42%	33.4	+73%	89.8	+115%
XC-1	14.9	+10%	25.1	+30%	70.1	+68%
XC-2	13.5	+0%	19.6	+2%	53.3	+28%
FC cycle length (in sec.)	32		40		88	
FC departure times (in sec.)	(6, 6, 6, 6)		(8, 8, 8, 8)		(20, 20, 20, 20)	

In Table 6.5, we study the waiting time at the different flows when the workload is 0.8. Although the arrival rates λ_f are identical for all flows, the mean waiting times differ per combination. Combinations C_1 and C_3 are 'thicker' than combinations C_2 and C_4, since the latter two combinations have only two flows each, whereas C_1 and C_3 consists of four flows each. Therefore C_1 and C_3 experience a lower waiting time than combinations C_2 and C_4 under all policies except FC. The average waiting times per car under FC are identical since all flows experience a departure time (or an effective green time) of 20 seconds (10 slots) per cycle.

Although FC seems to be most fair in the sense that the flows experience the same average waiting time, RV1 performs much better: the average waiting time to flows of C_1 and C_3 is 13 seconds (or 26%) lower than under FC, while the average waiting times to C_2 and C_4 are virtually the same as under FC. Further observe that both XC and anticipative-exhaustive control (XC-1 and XC-2) perform even worse than FC, when the workload is high.

Table 6.5: Mean waiting times (in sec.) for symmetric F12C4 at $\rho = 0.8$.

Rule	*EW* overall		*EW* C_1, C_3	*EW* C_2, C_4
RV1	41.8		37.4	50.6
FC dep. times 20, 20, 20, 20 sec.	50.5	+21%	50.5	50.4
XC	89.8	+115%	88.5	92.4
XC-1	70.1	+68%	68.9	72.4
XC-2	53.3	+28%	52.1	55.8

Waiting time and queue length distribution (partly-asymmetric F12C4)

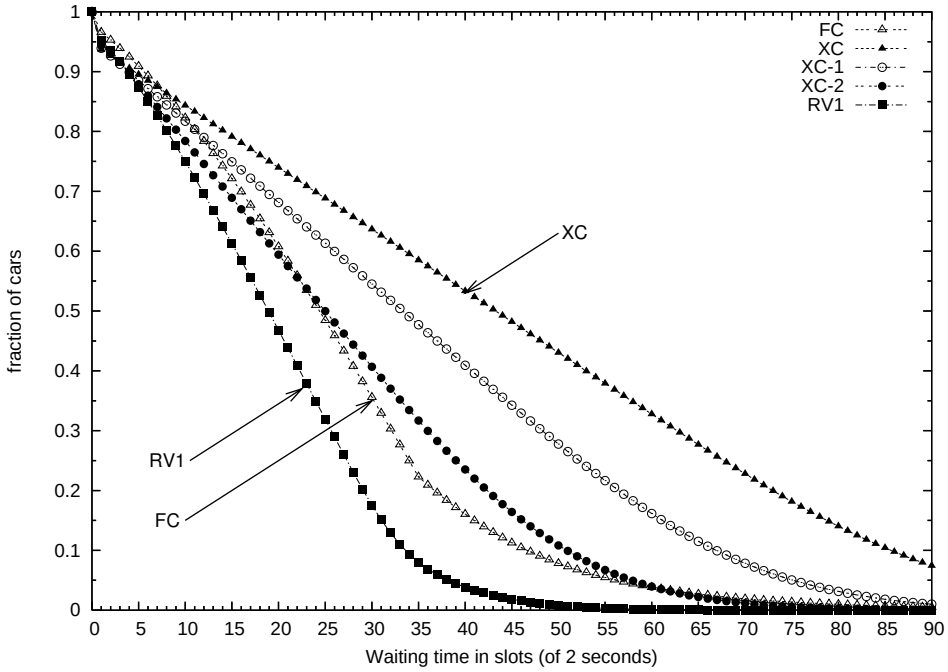
Waiting time distribution

Although our primary criterion is the average waiting time, we are also interested in the distribution of the waiting time. From the waiting time distribution one reads the fraction of cars that experience a waiting time above some level. Therefore we have computed the inverse cumulative distribution or the tail distribution of the waiting time based on the individual waiting times that are registered by the simulation program. Related to the distribution of the waiting time is the distribution of the queue length at the start of a slot: the more often a high queue length is observed the more cars experience a high waiting time.

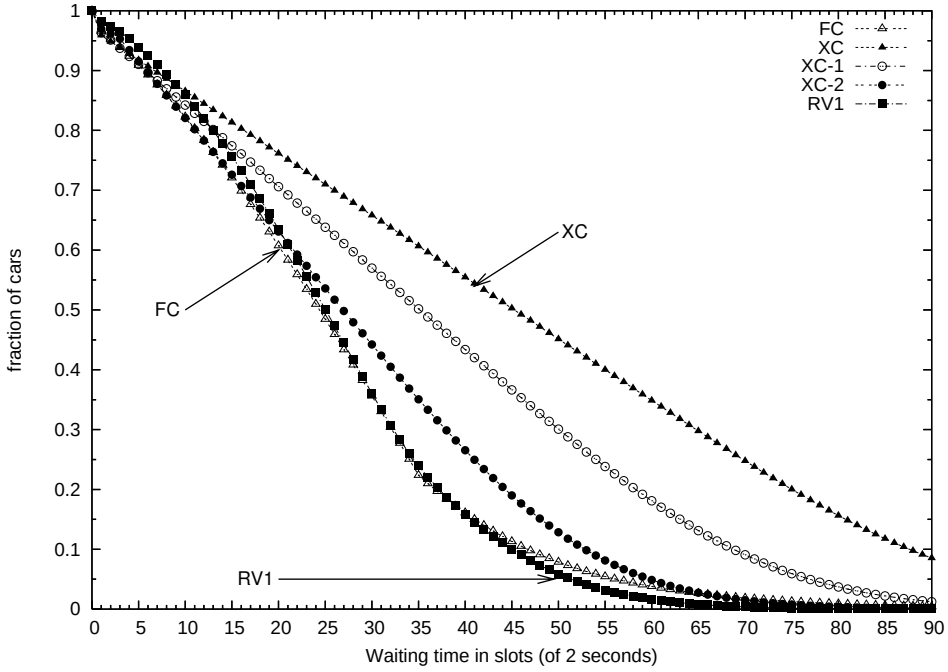
In Figures 6.8 and 6.9 we show the tail distributions of both the individual waiting time and the queue length at the start of a slot under the various control strategies: RV1, FC, and XC, XC-1, XC-2. Since the arrival rates are identical at all queues the only difference between flows is whether they are part of a thick or a thin combination. The distributions are computed after having simulated the policies for 4,000 hours (100 runs of 72,000 slots each), during which on average 1.44 million cars arrive at each flow. On the horizontal axis of Figure 6.8(a) and 6.8(b) one reads the individual waiting time which is an integer number of slots ranging from 0 to 90 slots (3 minutes). Vertically one reads the fraction of cars experiencing a waiting time greater than or equal to the value read at the horizontal axis.

Figure 6.8(a) depicts the waiting time distribution for the 8 flows part of the thick combinations. The probability of excessive waiting time, is lowest under RV. For example only 17% of the cars arriving at combination 1 or 3 experience a waiting time exceeding 1 minute (30 slots), while under FC still about 36% of the cars has to wait for at least 1 minute. Under pure-exhaustive control about 65% has to wait for 1 minute and often much more. Although the differences are smaller we conclude from Figure 6.8(b) that flows part of thin combinations do favor RV1 since it avoids excessive waiting times better than all other strategies.

Hence RV1 is superior not only in its mean waiting time but also in its distribution. A great reduction in the average waiting time at most of the queues is observed, while excessive waiting times at the other queues occur less frequently than under FC.



(a) Tail distribution of waiting time of cars in thick combinations: $f \in \mathcal{C}(1) \cup \mathcal{C}(3)$.



(b) Tail distribution of waiting time of cars in thin combinations: $f \in \mathcal{C}(2) \cup \mathcal{C}(4)$.

Figure 6.8: Inverse cumulative distributions (tail distributions) of the waiting time for flows f of thick and thin combinations of partly-asymmetric F12C4 ($\rho = 0.8$ and $\lambda_f = 0.2$).

Queue length distribution

In Figure 6.9 we show for the several policies the tail distributions of the queue lengths at the start of a slot. Based on the queue length distribution Figures 6.9(a) and 6.9(b), we draw similar conclusions: the queues of thick combinations are shorter under RV1 than under any of the other policies. At the other queues there is only a small difference in queue lengths between FC and RV; again the tail under RV1 is less heavy than under FC.

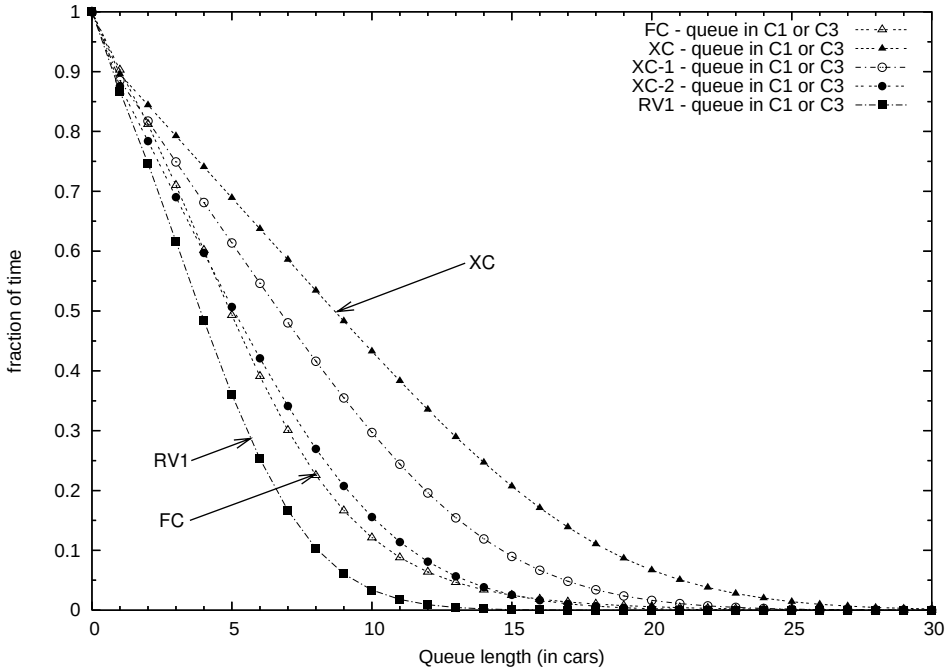
When comparing the waiting time curves of the FC policy to the queue length distribution under FC, we observe that the waiting time curve shows a bend at 35 slots, while the queue length distribution does not show a similar irregularity. This bend is explained by noting in this case an all-red period lasts 34 slots. Cars that that could not be served during the first green period that they encounter, need to wait at least another 34 slots higher for the start a next green period. The curves under the dynamic control policies do not show such an bend, since the all-red period has no fixed length under dynamic control. The queue length distribution under FC does also not show such an irregularity since cars arrive uniformly over time.

Non-identical arrival rates (fully-asymmetric F12C4)

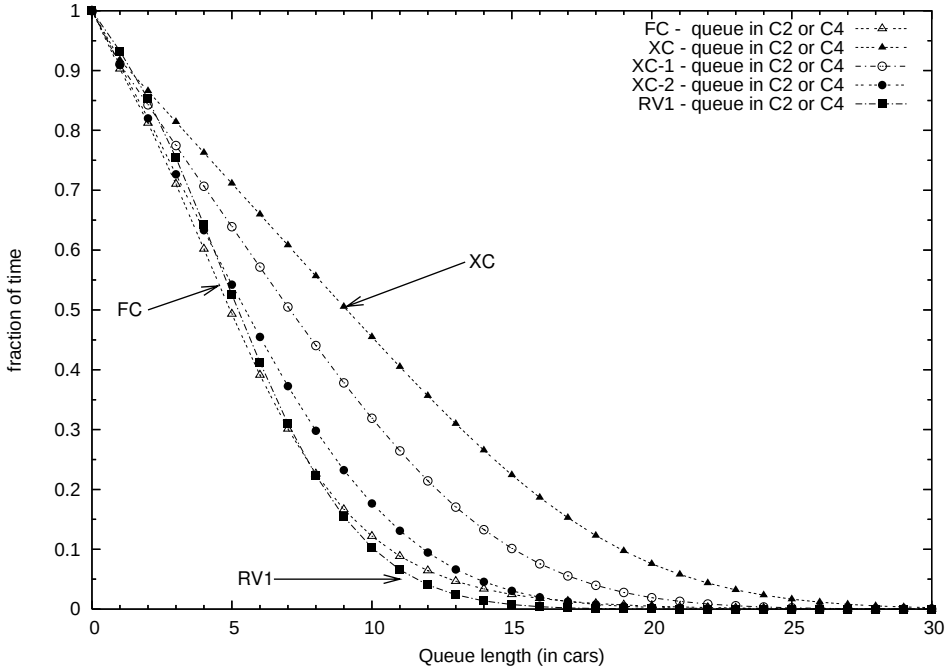
To study the impact of non-identical arrival rates in combination with an asymmetric number of flows per combination, we study the fully-asymmetric F12C4 intersection with workload 0.8. We assume C_2, which consists only 2 flows (flow 3 and 9) to have an arrival rate that is only a third of all other 10 flows. The arrival rates are thus 0.08 for flows 3 and 9 and 0.24 for all other flows. Then the overall load is again 0.8. Thus, C_2 is $(4 \cdot 0.24)/(2 \cdot 0.08) = 6$ times as thin as C_1 and C_3, and C_2 is three times as thin as C_4. Since C_2 adds little weight to the scale, one may expect C_2 to suffer high waiting times. In Table 6.6 the average waiting time per car is reported for all flows.

Table 6.6: for F12C4 at $\rho = 0.8$: C_2 is six times as thin as C_1 and C_3.

Rule	*EW* overall		*EW* C_1,C_3	*EW* C_2	*EW* C_4
RV1	39.4		34.9	66.6	48.7
FC dep. times 22, 8, 22, 22 sec.	47.1	+20%	45.6	69.4	45.6
XC	85.1	+116%	82.8	107.8	86.9
XC-1	66.6	+69%	64.6	84.8	68.3
XC-2	50.5	+28%	48.8	65.0	52.6



(a) Tail distribution of queue length of queues in thick combinations: $f \in \mathcal{C}(1) \cup \mathcal{C}(3)$.



(b) Tail distribution of queue length of queues in thin combinations: $f \in \mathcal{C}(2) \cup \mathcal{C}(4)$.

Figure 6.9: Inverse cumulative distributions (tail distributions) of the queue lengths for flows f of thick and thin combinations of partly-asymmetric F12C4 ($\rho = 0.8$ and $\lambda_f = 0.2$).

The waiting time of cars in C_2 is (considerably) lower than under FC, XC and XC-1, although the waiting time at C_2 is indeed higher than at all other flows. Apparently, C_2 is not under-served. XC-2 performs marginally better for C_2, but the average over all queues is 28% higher than under RV1. Pure-exhaustive control performs very bad, since it does not anticipate departures during the yellow slots. FC performs better than XC-2 but yields a waiting time that is still 20% above that of RV1.

We conclude that, also when arrival rates are non-identical, RV1 is superior, and that RV1 does not result in excessive waiting times at queues of thin combinations.

6.4.5 Sensitivity regarding the cycle length D

In Section 6.3.1, we have observed already that the curve of the average waiting time as a function of the cycle length D, is quite flat around the optimal value of D. For the partly-asymmetric F12C4 case we have seen that the local minima are where D is a multiple of $C = 4$ slots. We now return to that example and show that RV1 is even less sensitive to the chosen cycle length of the underlying FC.

In Figure 6.10 we have plotted the average waiting times for D ranging from 24 to 56 slots with increments of 4 slots. Although the overall long-run average waiting time under FC is high when D is set low, say $D = 24$, the average waiting is much lower under RV1. It appears that RV1 is almost insensitive to the cycle length of the underlying FC for which the relative values are computed.

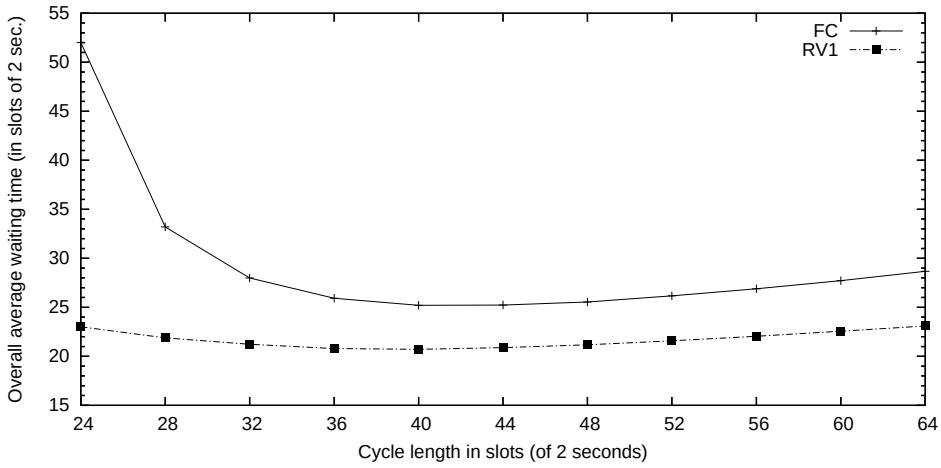


Figure 6.10: Sensitivity of FC and RV1 policy to cycle length for F12C4 at workload 80%

6.4.6 Conclusions

The optimal dynamic control of traffic lights, given the number of cars waiting at each queue, involves a high-dimensional state space. Therefore an optimal MDP policy can only be found for small intersections such as the F4C2 intersection. For more complex intersections such as the F12C4, one relies on heuristic and approximate solutions, such as RV1.

The newly developed policy RV1 is based on a one-step policy improvement algorithm over FC. Since RV1 improves FC, first one needs to determine a good configuration of FC, i.e. nearly optimal departure times and the related nearly-optimal cycle length D. Even when the selected FC is sub-optimal, RV1 shows very good improvements; it is quite robust for a broad range of cycle lengths D.

RV1 improves FC and exhaustive control policies at varying loads. At a high workload of the intersection, the control scheme has to be sophisticated. RV1 shows the greatest improvement over the other policies when the workload is high, say 0.8. Also when the intersection is asymmetric in the number of flows in each combination, like in F12C4, the control rule has to be more sophisticated than FC or exhaustive control. The F4C2 and F12C4 cases show that RV1 reduces the waiting time under FC by about 20%. For the F4C2 infrastructure, we have shown that RV1 performs nearly optimal. Especially for the F12C4 infrastructure, (anticipative) exhaustive control policies may perform even worse than FC. Clearly the definition of when to switch green lights into yellow should be refined when multiple queues can be served simultaneously. RV1 is refined at this point, and easy to implement.

6.5 More-advanced improvement rules

Although the results for RV1 are promising, we are interested in more advanced policies that make use of the relative values of the states under FC. First, we develop a variant called RV2 in which two decisions are evaluated simultaneously to find a best time jump. Next, three acyclic control policies RV1S, RV1M and RV2M are constructed.

6.5.1 Adjust two green periods (RV2)

The new policy RV1, developed and tested in the previous sections, performs very well, while it is based on only a single improvement step over FC. RV1 adjusts the length of the actual green period by making a time jump in the cycle: to evaluate a time jump one assumes that FC is resumed from that point onwards. In order to select a time jump RV1 ignores future time jumps that will be taken in the real system. Although for the F4C2 infrastructure, we have checked that RV1 performs nearly optimal, this cannot be checked for the F12C4 infrastructure. For the F12C4 infrastructure, decisions based on the queue lengths are more refined, therefore one may expect to get better decisions when a time jump is evaluated by also taking a next time jump into account.

A one-step policy improvement over RV1 cannot be executed, because the relative value vectors cannot be derived for RV1 since the state space under RV1 cannot be decomposed. Therefore we propose to include a second time jump, which plans where to resume FC after the next switching phase. In the evaluation of a first time jump, the second time jump allows thus to shorten the length of the next green period. We refer to this new policy as RV2. In Figure 6.11 the two time jumps are depicted. Suppose at time t the lights are green to some combination. Then an immediate time jump to τ_1 is evaluated by assuming that the next green period will be started at slot τ_2 instead of at slot r. (e and r denote respectively the end of the green period and the end of the next all-red slot, which is the start of the next green period under FC.)

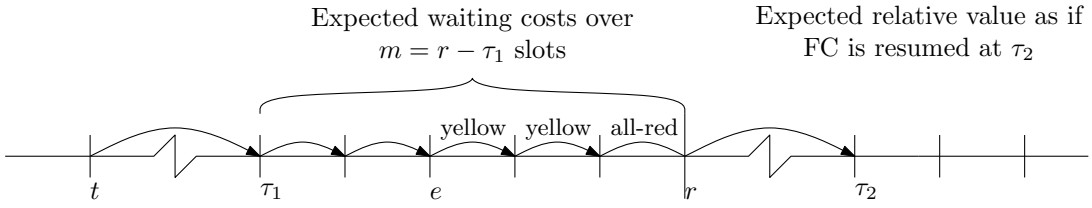


Figure 6.11: Setting RV2 actions requires evaluating two jumps τ_1 and τ_2

RV2 adjusts the lights according to a time jump τ_1, but for the evaluation of the time jump, RV2 optimizes over the length of the next green period by selecting a second time jump τ_2. Given the actual state $(t, \mathbf{q})$ at the start of a slot, RV2 derives the best pair (τ_1, τ_2) that minimizes the long-run average waiting cost assuming that FC is resumed at τ_2. When evaluating a time jump RV2 assumes FC to continue from slot τ_1 to the end of the next all-red slot, after which a second time jump τ_2 is selected, as illustrated in Figure 6.11. The end of the all-red slot coincides with the start of slot r at which a next green period would start (if not interrupted by τ_2). RV2 thus implements only the first time jump from t in to τ_1. The second time jump in RV2 is introduced solely to obtain a more accurate evaluation of jumping from t to τ_1.

For example, $\tau_1 = t-1$ implies to lengthen the current green period by one slot for the flows that have right of way at slot t. Under RV1 all cars queued at the other combinations need to wait one more slot. Under RV2 however, the next green period may be shortened, just as what may happen in the real system, and consequently the red period of combinations visited thereafter may be shortened. A time jump that may look not beneficial under RV1 could thus be beneficial in the real system. RV2 better resembles the real system since it may anticipate the action taken at the end of the next all-red slot.

Computations of RV2 actions

The process depicted in Figure 6.11 can be modeled as a MC. Except for the jump from r to τ_2, the process resembles an MC of FC. The MC is characterized by the state $(t, \mathbf{q})$, and the time jump τ_2. The process at queue f is independent of the arrival and departure processes at the other queues. The relative values for this MC may be computed flow-by-flow. Let $u^f_m(\tau_1, q_f, \tau_2)$ denote the relative value for flow f of starting, m slots before the start of the next green period, the MC in slot τ_1 with q_f cars queued and breaking FC at the start of a next green period by jumping to slot τ_2.

For the computation of the relative values for the interrupted FC, one may use the approximation of the relative values, $\mathbf{v}^f_{rel}$, of FC. In words, $v^f_{rel}(\tau_2, q_{f,m})$ is used as terminal costs when ending in state $(\tau_2, q_{f,m})$; $q_{f,m}$ is a stochastic variable indicating the number of cars waiting at queue f after m slots. We add to $v^f_{rel}(\tau_2, q_{f,m})$, the expected waiting costs over the first m slots. Since we have truncated the (in principle) infinite horizon in the approximation of the relative values by $\mathbf{v}^f_{rel}$, we need to compensate for unequal horizons by subtracting m times the average costs per slot under FC, $g^{(f)}$.

RV2 algorithm – The following algorithm is used to derive RV2 actions: Steps 1 – 3 can be computed off-line. Step 4 is executed run time: every slot a best RV2 action is read from Equation (6.27). The algorithm implicitly computes the relative values of states for Markov chains similar to the one presented in Figure 6.11.

Step 1. Approximate for all flows f, the relative values of the states under FC, $\mathbf{v}^f_{rel}$, and the gain, $g^{(f)}$, of the MC for flow f.

Step 2. Set for all flows f, for all $q_f \in \{0, 1, \ldots, Q\}$, and for all $\tau_1, \tau_2 \in \{1, \ldots, D\}$:

$$u^f_0(\tau_1, q_f, \tau_2) = v^f_{rel}(\tau_2, q_f). \tag{6.24}$$

Step 3. Compute recursively for for all flows f, for $m = 1$ to $d_f + 1$:
for $\tau_1 := 1$ to D, for all $q_f \in \{0, 1, \ldots, Q\}$:

- if τ_1 implies red to flow f:

$$\begin{aligned} u^f_m(\tau_1, q_f, \tau_2) \;=\; q_f \;+\; & \lambda_f \quad u^f_{m-1}(\tau_1 + 1, q_f + 1, \tau_2) \;+ \\ +\; & (1 - \lambda_f) \quad u^f_{m-1}(\tau_1 + 1, q_f, \tau_2) \\ -\; & g^{(f)} \end{aligned} \tag{6.25}$$

  (when $q_f = Q$, then $u^f_{m-1}(\tau_1 + 1, q_f + 1, \tau_2)$ is approximated by quadratic extrapolation),

- if τ_1 grants green or yellow to flow f:

$$\begin{aligned} u^f_m(\tau_1, q_f, \tau_2) \;=\; q_f \;+\; & \lambda_f \quad u^f_{m-1}(\tau_1 + 1, q_f, \tau_2) \\ +\; & (1 - \lambda_f) \quad u^f_{m-1}(\tau_1 + 1, (q_f - 1)^+, \tau_2) \\ -\; & g^{(f)}. \end{aligned} \tag{6.26}$$

Step 4. An optimal RV2 action τ_1 in state $(t, \mathbf{q})$ is online computed by:

$$\arg \min_{\substack{\tau_1 \in \mathcal{T}(t, \mathbf{q}) \\ \tau_2 \in \mathcal{T}(r, \mathbf{q})}} \sum_{f=1}^{F} u^f_{r-\tau_1}(\tau_1, q_f, \tau_2), \tag{6.27}$$

(with $r =$ slot number at which a next green period would start when FC would be continued).

Before presenting results for RV2 (in Section 6.6), we first introduce a few more-advanced acyclic RV policies.

6.5.2 More-advanced acyclic RV policies (RV1M and RV2M)

In Section 6.4 we have studied results for cyclic control policies. By using action space $\mathcal{T}(t, \mathbf{q})$ instead of $\mathcal{T}^{\mathrm{C}}(t, \mathbf{q})$, acyclic control policies are considered. In the evaluation of a feasible time jump τ, RV1 assumes FC to resume at τ. Hence RV1 does not allow to evaluate a decision by assuming a different cyclic order in which the combinations receive green.

Therefore we have developed another RV policy: RV1M. Under RV1M a time jump to τ is evaluated more carefully by allowing to modify the cyclic order in which the combinations receive green. RV1M selects the best τ based on the best cyclic order at which FC may resume. Since decisions are updated every slot, only decision τ is implemented every slot.

Below we deliberate on the acyclic counterpart of RV1, and on RV1M.

Acyclic RV1

For example, consider the F12C4 case and the green period of C_3 has just ended. Now, the controller needs to select a combination to serve next. When τ grants green to C_1, FC is resumed in the order C_1-C_2-C_3-C_4 - C_1-C_2-etc. Hence under RV1 C_4 is skipped and thus served after all other combinations are visited. Since this might be too bad to C_4, a jump to τ might be rejected under RV1, although it could be optimal to serve C_1 before C_4.

RV1M

Under RV1M we may modify the cyclic order to better evaluate the jump to τ. (The 'M' in RV1M stands for Modify the cyclic order.)When τ grants green to C_1, RV1M not only considers the cyclic order C_1-C_2-C_3-C_4 - C_1-C_2-etc. Instead RV1M evaluates τ as if FC continues at the best of the $((C-1)! = 3! =)$ six possible cyclic orders:

Cyclic order 1:	C_1-C_2-C_3-C_4 - C_1-C_2-...
Cyclic order 2:	C_1-C_2-C_4-C_3 - C_1-C_2-...
Cyclic order 3:	C_1-C_3-C_2-C_4 - C_1-C_3-...
Cyclic order 4:	C_1-C_3-C_4-C_2 - C_1-C_3-...
Cyclic order 5:	C_1-C_4-C_2-C_3 - C_1-C_4-...
Cyclic order 6:	C_1-C_4-C_3-C_2 - C_1-C_4-...

For the selection of the best τ from the relative values of the states under FC, we need to consider in this case six potential new intended cycles ϕ. The set of all possible cycles, is denoted by $\Phi(\tau)$ and depends on the slot number τ in the base cycle C_1-C_2-C_3-C_4. In principle RV1M is still a one-step policy improvement based on the relative values of states under FC, but next to τ one selects a best cyclic order, ϕ, of combinations by which FC resumes. Under RV1M the best decision is thus the best pair (τ, ϕ), with $\phi \in \Phi(\tau)$ being the new intended cyclic order.

In order to value a decision (τ, ϕ), one needs the relative values of the states under FC. The relative values of states under FC with cyclic order ϕ can be derived from the relative values (6.22) of states under FC with cyclic base order C_1-C_2-C_3-C_4. Instead of saving $(C-1)!$ relative value tables, we suggest to save on memory usage, using the fact that the relative values of states for different ϕ are only shifted in the slot number t.

The relative values $\mathbf{v}^f_{rel}$, in Equation (6.22), relate to the original cycle (C_1-C_2-C_3-C_4). Suppose that under this cyclic order flow f receives green from slot $t = r_f$ to slot e_f. When the cyclic order changes, f receive green at other slot numbers: the start of a green period may not no longer be at slot r_f, but say at slot $r_f + \delta_f(\phi)$. The relative value curve of flow f is shifted by a constant δ_f which depend on the new intended cycle ϕ.

Example – Consider a case where $C = F = 4$ (one flow per combination) and τ grants green to C_1. The cyclic base order is (C_1-C_2-C_3-C_4), and the green period of flow f starts at slot r_f. When one evaluates the cyclic order $\phi = (C_1$-C_3-C_2-$C_4)$, then the green period of C_2 is postponed by $\delta_2 = r_4 - r_3$ slots: which is the effective green time of C_3 plus one all-red slot. C_3 gets green $r_3 - r_2$ slots earlier than under the cyclic base order. The delay δ_3 is thus negative for C_3, since the time until it receives green is shortened: $\delta_3 = r_2 - r_3$. The order of C_1 and C_4 is preserved, hence $\delta_1 = \delta_4 = 0$.

For this particular order ϕ the delays are thus $\delta(\phi) = (0, r_4 - r_3, r_2 - r_3, 0)$, which are the shifts for looking up the right relative values.

Actions under RV1M – The optimal actions under RV1M follow from Equation (6.28), where $\tau + \delta_f(\phi)$ should be read as $[(D + \tau + \delta_f(\phi)) \mod D]$ whenever $\tau + \delta_f(\phi)$ falls not in $\{1, 2, \ldots, D\}$.

$$\sigma^{\text{RV1M}}(t, \mathbf{q}) = \arg \min_{\tau \in \mathcal{T}(t,\mathbf{q})} \left(\min_{\phi \in \Phi(\tau)} \sum_{f=1}^{F} v^f_{rel}(\tau + \delta_f(\phi), q_f) \right). \tag{6.28}$$

RV2M

Similarly one may investigate multiple cyclic orders ϕ under RV2: we call the resulting policy RV2M. Under RV2M one thus selects the best triple (τ_1, τ_2, ϕ). We skip a formal definition of RV2M since it involves complicated notations.

Complexity of RV1M and RV2M

When a feasible time jump is to be evaluated over all possible cyclic orders, the computational complexity grows exponentially in the number of combinations. The combination to serve first is set by the time jump (τ or τ_1). The remaining $C-1$ combinations can be visited in $(C-1)!$ different orders. (If the combination that has been served most recently has to wait until all other combinations are served, then the number of cyclic orders to consider is $(C-2)!$.)

For the F12C4 intersections one deals with $C = 4$ combinations. The number of possible cycles is small enough for considering all permutations. As a result simulating RV2 takes considerable more time than simulating RV1. For cases where the number of combinations is (much) larger, one could save time by considering not all permutations of the $C-1$ combinations, but only those in which two combinations are pulled forward in the intended cycle. Then instead of $(C-1)!$ permutations only $(C-1)\cdot(C-2)$ possible cycles are considered. The simulation program in which we test RV1M and RV2M allows for considering a subset of all permutations.

6.6 Evaluation of advanced RV policies

With the introduction of the new policies we are interested in comparing cyclic against acyclic control. Furthermore, we question whether the more advanced rules RV2, RV1M and RV2M yields much lower average waiting time than RV1. In this section we investigate the quality of these policies and compare them against FC and a number of (anticipative) exhaustive control policies. For the F4C2 case acyclic policies are not of interest, since F4C2 consists of only two combinations. Therefore the focus is on the infrastructure F12C4, under different loads. Again all results are obtained by simulating each policy for 4,000 hours (100 runs of 72,000 slots), such that the results reported are accurate up to 2 or 3 digits.

6.6.1 F12C4 with identical arrival rates

In Table 6.7 one reads for varying workloads at a partly-asymmetric F12C4 intersection, the long-run average waiting time per car overall flows. The top part of the table, containing the results for the cyclic policies, greatly coincides with Table 6.4, except that the results for RV2 are added. RV2 appears to perform equally well as RV1: although RV2 is based on a more accurate evaluation, it hardly improves RV1.

When the control of the lights may be acyclic, RV1 shows an improvement of 7.7% over cyclic RV1, when the workload is low (0.4). However when the workload is low, anticipative-exhaustive control XA-2 performs even better. (Remember: under XA, XA-1, and XA-2 one switches to the combination with the longest queue at the end an all-red slot.) The lowest waiting time at $\rho = 0.4$ is obtained under RV1M and RV2M: 12.4% below RV1. Apparently, modifying the cyclic order is important when the workload is low.

When the workload is high ($\rho = 0.8$), the more advanced acyclic rules perform hardy any better than RV1. When the load is high it becomes critical not to waste much time on switching or on serving queues at which not much traffic is waiting. The best anticipative exhaustive control policy yields a waiting time that is about 28% above that of RV1.

Conclusions – One may conclude that when the workload is low one needs to select carefully which combination to serve next, while when the workload is high one should take care in formulating when to switch. For the best results RV1M (or RV2M) is favored, but when the control has to be cyclic RV1 is favored because of it requires less and more simple computations than RV2.

Table 6.7: Overall mean waiting time (in sec.) at different loads for F12C4 ($\lambda_f = \rho/4$).

Rule	$\rho = 0.4$		$\rho = 0.6$		$\rho = 0.8$	
Cyclic policies:						
RV1	13.5		19.3		41.8	
RV2	13.5	0.0%	19.4	+0.1%	41.6	−0.5%
FC	15.0	+10.9%	23.7	+23%	50.5	+20.8%
XC	19.2	+41.9%	33.4	+73%	89.8	+114.8%
XC-1	14.9	+10.1%	25.1	+30%	70.1	+67.2%
XC-2	13.5	+0.0%	19.6	+2%	53.3	+27.6%
FC cycle length (in sec.)	32		40		88	
FC departure times (in sec.)	(6, 6, 6, 6)		(8, 8, 8, 8)		(20, 20, 20, 20)	
Acyclic policies:						
RV1	12.5	−7.7%	18.9	−2.1%	41.7	−0.2%
RV1M	11.9	−12.4%	18.3	−5.3%	41.7	−0.2%
RV2M	11.9	−12.4%	18.3	−5.4%	40.9	−2.1%
XA	18.7	+37.9%	33.4	+72.6%	90.0	+115.4%
XA-1	14.0	+3.1%	25.1	+29.7%	70.2	+68.1%
XA-2	12.3	−9.1%	19.1	−1.0%	53.8	+28.8%

Waiting time per flow

Although the arrival rates per flow are identical (0.1, 0.15, 0.2 when the workload is 0.4, 0.6 and 0.8 respectively), the mean waiting times may differ per combination. C_1 and C_3 are 'thicker' than C_2 and C_4, since the latter two combinations have only two flows each whereas C_1 and C_3 consists of four flows each. Under a policy that takes the number of queued cars into account, such as the RV policies, we may expect that car drivers in C_1 and C_3 experience a lower waiting time than car drivers in C_2 and C_4.

In Table 6.8 detailed results are reported for $\rho = 0.8$. RV1 and RV2 yield a much lower overall average waiting time than FC, but not at the expense of a high average waiting time at the queues of the thin combinations C_2 and C_4. Under cyclic control, the average waiting times at C_2 and C_4 are just as high as under FC. The waiting time at the other queues is reduced by more than 25% from 50.5 seconds to about 37 seconds.

Under acyclic control, thin combinations may be skipped or their service may be ended too early, in favor of serving a thick combination. Nevertheless, RV1, RV1M and RV2M yield average waiting times that do not differ much from those under FC. The waiting time at C_2 and C_4 stays very close to that under FC, while the waiting time at the other

Table 6.8: Mean waiting times (in sec.) for F12C4 at $\rho = 0.8$ ($\forall f : \lambda_f = 0.2$).

Rule	*EW* overall		*EW* C_1, C_3	*EW* C_2, C_4
Cyclic policies:				
RV1	41.8		37.4	50.6
RV2	41.6	−0.5%	37.1	50.3
FC dep. times 20, 20, 20, 20 sec.	50.5	+20.8%	50.5	50.4
XC	89.8	+114.8%	88.5	92.4
XC-1	70.1	+67.2%	68.9	72.4
XC-2	53.3	+27.6%	52.1	55.8
Acyclic policies:				
RV1	41.7	−0.2%	37.3	50.6
RV1M	41.7	−0.2%	36.2	52.9
RV2M	40.9	−2.1%	35.2	52.5
XA	90.0	+115.4%	88.0	94.0
XA-1	70.2	+68.1%	68.3	74.1
XA-2	53.8	+28.8%	51.8	57.7

queue is as low as 35.2 seconds: a reduction of 31%. Compared to acyclic exhaustive control the waiting times at all queues is much lower under the RV policies. Even the best anticipative acyclic exhaustive control policies yield waiting times that are higher than under FC.

The difference between cyclic and acyclic control, in terms of the average waiting times, is very small.

6.6.2 Non-identical arrival rates (F12C4)

Hypothesis

Flows that contribute little to the total workload may suffer high waiting times under acyclic control, since they may be skipped or their green period is ended too early in favor of serving thick combinations. Under the RV policies, we hope and expect that this will not happen too much, since letting many cars waiting at a thin combination is penalized by the definition of the relative values.

Ending a green period too early implies that all cars left behind at that queue need to wait for the next green period. A thin combination has a shorter green period in FC than a thick combination. Letting say 5 cars waiting at a thin combination is consequently less

Table 6.9: Mean waiting times (in sec.) for fully-asymmetric F12C4 at $\rho = 0.8$ – C_2 is six times as thin as C_1 and C_3.

Rule	*EW* overall		*EW* C_1,C_3	*EW* C_2	*EW* C_4
Cyclic policies:					
RV1	39.4		34.9	66.6	48.7
RV2	39.3	−0.2%	34.4	66.3	48.9
FC dep. times 22, 8, 22, 22 sec.	47.1	+19.5%	45.6	69.4	45.6
XC	85.1	+115.9%	82.8	107.8	86.9
XC-1	66.6	+68.9%	64.6	84.8	68.3
XC-2	50.5	+28.1%	48.8	65.0	52.6
Acyclic policies:					
RV1	38.7	−1.7%	33.6	75.8	46.5
RV1M	38.3	−2.9%	32.5	82.5	46.9
RV2M	37.6	−4.5%	31.8	79.8	46.4
XA	82.7	+109.7%	73.8	201.9	78.7
XA-1	64.8	+64.4%	57.7	158.1	62.3
XA-2	49.7	+26.2%	44.0	121.1	48.8

desirable according to the relative values than letting 5 cars waiting at a thick combination. Since under FC some cars may have to wait for even two or more cycles before getting served, when some queue become long, the RV policies aim at keeping all queues short.

We expect thus very thin combinations not to be skipped under the RV policies, although they may experience higher waiting times than cars at the other queues. We test this hypothesis for the F12C4 case with the arrival rates of the queues in C_2 being only a third of the arrival rate at the other ten queues. The workload of C_1 and C_3 is thus six times as high, since these combinations consist of four flows each. When the total work load (ρ) is 0.8 the arrival probability for the flows in C_2 is 0.08, for the other ten flows it is 0.24. The results are reported in Table 6.9.

Results

Under cyclic control a thin combination has to be served before a next thick combination can is helped, but it service may be stopped (well) before its queues become exhausted. Nevertheless, the long-run average waiting time at a thin combination is not very high under RV1 or RV2. The long-run average waiting time at C_2 is higher than at the other queues, but it is (considerably) lower than under FC, XC and XC-1. Indeed the relative

values prevent long queues at the thin combinations. The lowest waiting time to C_2 is experienced under XC-2, but the difference with cyclic RV1 and RV2 is small. The overall average waiting time under XC-2 is 28% above that under cyclic RV1 and RV2.

Under acyclic control the service of C_2 may be postponed in order to serve a thicker combination. Since the visiting order is no longer cyclic, the service of a thin combination is no longer enforced in the action space. Because of this, one observes a higher average waiting time at C_2, than under cyclic control.

The difference in the average waiting time between cyclic and acyclic control is more visible, now that not all combinations have identical loads. Moreover RV2M yields a somewhat lower waiting time than RV1. Compared to FC the waiting time under RV2M at the eight queues of C_1 and C_3 has dropped by more than 30% (14 seconds), while the waiting time at C_2 has increased by only 15% (8 seconds). Acyclic anticipative-exhaustive control XA-2 performs very bad, since C_2 is only served when the length of one it's queues is the longest over all 12 queues; due to the low arrival rate this may take a relatively long time during which any cars present have to wait.

All RV policies yield thus fairly low waiting times even at the thin flows, apparently the relative values avoid queues to become very long.

6.6.3 Conclusions for F12C4 cases:

- The relative values used in the RV policies prevent queues to become long, even for a thin combination, which contributes little to the total workload.
- When the total workload is low, acyclic control may be preferred; when the total workload is high the difference between cyclic RV policies and acyclic RV policies is small.
- RV1 is the preferred policy for cyclic control, since RV2 is more complicated while it does not lead to significantly lower average waiting times.
- For acyclic control the best option seems to be RV2M, although RV1 or RV1M may be may be favored for requiring less computations.

6.7 Conclusions and Discussion

6.7.1 Conclusions

The dynamic control of traffic lights, such that the long-run average waiting time over all flows is minimized, can be formulated as an MDP. However an optimal solution can be computed only for intersections where the number of queues is small. For most realistic intersections, one relies on heuristics such as FC, and vehicle actuated control, which is a mixture between exhaustive control and FC. Under FC the system has a well-structured state space, which can be decomposed into subspaces for each flow. The relative values of states under FC can thus be computed numerically for each flow in isolation of all others. These relative values are used to develop what we call RV policies, in which a best position (slot number) in the cycle is determined given the queue lengths.

RV1 is based on a one-step policy improvement algorithm with FC as the initial policy. RV2 optimizes every slot over two successive time jumps, although only the first time is implemented. RV1M and RV2M do allow to modify the cyclic order while evaluating the time jumps in a fixed cycle. Based on a number of test cases for two different infrastructures we conclude:

- RV1 yields a great improvement of FC and exhaustive control policies.
- RV1 is robust in the chosen cycle length for the underlying FC: RV1 performs also very good when based on a sub-optimal FC. This is important since finding an optimal FC is in general difficult.
- When the workload is high, the more advanced RV policies (such as RV2, RV1M, and RV2M) show only a minor improvement compared to RV1.
- When the workload is low, acyclic control, e.g. by RV1M, is favored.
- The one-step policy improvement algorithm, which results in policy RV1, is promising for reducing the waiting times at signalized intersections.

6.7.2 Discussion

The quality and robustness of the RV policies have been investigated and compared based on several interesting cases. In this section we briefly discuss how the RV policies can deal with additional objectives and with a number of complexities that arise in practice.

Different objectives in cost function

We have focussed on minimizing the overall long-run average waiting time. Another criterion, commonly studied in traffic light engineering, is to minimize the number of stops made by the cars. The cost structure can easily capture this criterion by including cost for each stop a car makes in the queue. The question to minimize the number of stops may be more prevalent when dealing with networks of intersection.

Serving pedestrians and cyclists

In the cases that we have studied we excluded flows of pedestrians and cyclists. The departure process of pedestrians and also that of cyclists differ from the departure process of cars, since when the light turns green all pedestrians waiting at a cross walk start crossing the street more or less at the same time. Furthermore, the number of pedestrians may not be known to the controller: the controller only receives information about whether there are any pedestrians are waiting. The same may hold to cyclists: the number of cyclists waiting at an intersection may not be known to the controller and cyclist do not leave one-by-one. At some intersections cameras or induction loops are present to count or estimate the number of cyclists waiting, but usually their waiting time cannot be estimated accurately, since only the first moment of pressing a button to call for service is registered.

Therefore cyclists and pedestrians require a special treatment. In practice the green period is fixed to a pre-specified time that a cyclist or pedestrian needs to cross an intersection, which is independent of the actual number of cyclists or pedestrians waiting at a light. The service of cyclists or pedestrians can be included in FC as special combinations. In FC fixed periods are then included during which cyclists or pedestrians are served. These fixed period may be skipped only when no cyclist or pedestrian is present. Furthermore, the RV policies are not allowed to shorten the green period of these special combinations.

Under acyclic dynamic control we may need to specify a 'soft' bound b on their waiting time; when pedestrians or cyclist are waiting more than b slots, they are the next

combination to get green. The process of serving car traffic is then continued until some pedestrian(s) or cyclist(s) wait for more than (at least) b slots. Then, as soon as a green period to car traffic ends, the respective lights for pedestrians and cyclists turn green.

Public transport

Trams and busses often drive on separate lanes in Dutch cities, and at some places they get priority over car traffic for political and environmental reasons. Green periods to cars are ended instantaneously when a bus or tram approaches an intersection. The RV policies can of course be interrupted in this way, but for the minimization of the waiting time we suggest to let RV1M decide about when to end a green period based on the number of cars waiting at the queues. Public transport thus gets priority as soon as an all-red slot is visited. Such a priority setting fits better to the objective of minimizing the long-run average waiting time, while it still provides priority to public transport.

When at an intersection busses and trams arrive very frequently one may want to include one or more special combinations in FC, just as for cyclists and pedestrians. When queueing of trams and busses is allowed, one may even compute relative values of the queue lengths for public transport. To give buses and trams a higher weight one may incur (much) higher cost for busses and trams for each slot that they are queued. One may even introduce a cost structure that is progressive in the number of queued busses and trams.

More complex cycles and sub-cycles in FC

In the cases that we have studied, the set of flows $\mathcal{F}$ was split into C mutually exclusive subsets, called combinations. Under cyclic control each flow is visited once every cycle, and an all-red slot applies when switching between two combinations. In general, a flow may be part of multiple combinations, and the lights of one or more flows may stay green while switching form one combination to another. The cyclic RV1 and RV2 policies can easily be modified to cases where a flow is visited multiple times in a cycle, since one only needs to know during which slots of FC its lights are green, yellow, and red.

In the practice of traffic light engineering, a cycle may contain two (or more) sub-cycles: e.g. in both sub-cycles all or just a part of the flows is served, but the first sub-cycle may be longer than the second. The RV policies can easily be extended to make time jumps in a cycle of multiple sub-cycles: one then allows to jump from a short cycle to a longer

cycles. Since RV1 is hardly sensitive to changes in the cycle length we do not expect great improvements of considering sub-cycles.

Serving conflicting flows simultaneously

Thus far we have presumed that conflicting traffic flows are not in the same combination and thus do not get right of way simultaneously. In practice, some conflicting flows may receive green simultaneously, in which case basic traffic rules apply to solve any conflicts. For example, left-turning traffic flow f may be served together with opposite thru traffic originating from flow h. Flow f than needs to give right of way to flow h, whenever cars leave queue h.

In deriving the relative values of states, one thus needs to acknowledge that the departure process of f depends on the number of cars present at queue h. The transitions become a bit more involved. Suppose that only flow h has priority over f, and no flow in the same combination has priority over h. When slot t grants green to flows f (and h), Equation (6.19) then becomes:

$$\begin{aligned} V^f_{n+1}(t,q_f,q_h) \;=\; q_f \;&+\; \lambda_f\cdot\lambda_h \;\cdot\; V^f_n(t+1,q_f+1,q_h) \\ &+\; \lambda_f\cdot(1-\lambda_h) \;\cdot\; \big[\, V^f_n(t+1,q_f+1,q_h-1)\cdot I(q_h\geq 1) \\ &\qquad\qquad + V^f_n(t+1,q_f,0)\cdot I(q_h=0)\,\big] \\ &+\; (1-\lambda_f)\cdot(1-\lambda_h) \;\cdot\; \big[\, V^f_n(t+1,q_f,q_h-1)\cdot I(q_h\geq 1) \\ &\qquad\qquad + V^f_n(t+1,(q_f-1)^+,0)\cdot I(q_h=0)\,\big] \\ &+\; (1-\lambda_f)\cdot\lambda_h \;\cdot\; V^f_n(t+1,q_f+1,(q_h-1)^+). \end{aligned}$$

The value vector of flow h can be computed as before as, using Equation (6.19), since its departure process does not depend on any other flow. The state space is thus decomposed in a somewhat different way. The relative values are computed in a very similar way as before, but the relative value vector of flow f is three-dimensional, while that of h is two dimensional: $v^f_{rel}(t,q_f,q_h)$ versus $v^h_{rel}(t,q_h)$.

Outlook

In the next chapters, the RV policies are extended to include information on near future arrivals, and the policies are applied to networks of intersections.

Chapter 7

Single intersection with arrival information

7.1 The control problem with arrival information

Current technology allows not only to measure the number of cars waiting at each queue, but is also capable of detecting in the vicinity of intersections ([93], [65]). Information on near future arrivals comes from induction loops cut into the surface of the roads or from cameras positioned above or along the approach ([93], [65]). In the latter case, cars driving in the range of a camera are observed and by using image processing techniques their position on the approach and the driving speed can be estimated. The traveling speed of cars can also be derived from the time elapsed between a car passes two subsequent induction loops. The controller translates the information in an estimated time until a car reaches the tail of a queue (ToQ).

In the decision model that we introduce in the next section we assume that the estimated arrival times are rounded to slots of 2 seconds: cars that arrive at the ToQ within $\langle m-1, m]$ slots from now are said to arrive m slots from now. For each car on the approach one can thus estimate the number of slots it takes to reach the ToQ. This arrival information can be translated into a vector $\mathbf{a}^f = (a^f_1, \ldots, a^f_{M(f)})$ for each flow f, with elements a^f_m the number of cars in flow f that arrive m slots from now at the end of Q_f. $M(f)$ denotes the (maximum) number of slots for which arrival information is available for flow f. For some flows only information on the queue length is available, and no information on near-future arrivals is available by a lack of cameras or upstream induction loops: $M(f) = 0$.

The information on the arrival times of near future arrivals can be used to reduce the

long-run average waiting time by anticipating these arrivals. Inclusion of the arrival information in the MDP model, that we have introduced in Chapter 6, results in an explosion of the state space. The state of the system will then be $(x, \mathbf{q}, \mathbf{a}^1, \mathbf{a}^2, \ldots, \mathbf{a}^F)$ instead of $(x, \mathbf{q})$.

For example, consider the simple F4C2 case with only four flows that are served in two combinations, and suppose that for each flow 5 slots arrival information is available. When cars keep a safe driving distance of 1 slot, a_m^f can take only 2 values: 0 or 1 car. For each flow f, the vectors $\mathbf{a}^f$ can thus take $2^5 = 32$ possible values. The total number of arrival patterns over the next 5 slots is thus $32^4 \approx 1$ million. In Chapter 6, we have discussed that the number of traffic light states in the MDP model is $C \cdot (1+3) = 2 \cdot 4 = 8$ and the number of queue states is 10^4 when in the MDP model queues are truncated at $Q = 9$ cars. The total number of states becomes thus already $8 \cdot 10^{10}$ for a simple intersection with only 4 flows.

Clearly, the optimal MDP policy cannot be computed within a reasonable time. Fortunately, as we have learned in Chapter 6, a one-step policy improvement over FC is possible, since under FC the state space can be decomposed into F subspaces. In this chapter, a similar approach is followed and the RV policies are extended, such that the available arrival information is taken into account. This extension is of particular interest for the control of networks of intersections, as we will discuss in Chapter 8.

Outline

After having extended the RV policies in the next section, an advanced simulation model is presented, through which we test the policies in a quite realistic setting. The simulation model is discussed in Section 7.3. The new policies are put to the test for the F12C4 infrastructure, which is interesting for its combinations being asymmetric in the number of flows. The results are reported in Sections 7.4 and 7.5, for varying scenarios concerning the available number of slots of arrival information.

7.2 One-step policy improvement of FC

Through a one-step policy improvement algorithm one may improve FC and hopefully obtain a nearly optimal policy. Therefore the relative values of states $(t, \mathbf{q}, \mathbf{a}^1, \ldots, \mathbf{a}^F)$ are to be computed.

7.2.1 Relative values of states under FC

Decomposition

Under FC the signal state is a single number $t \in \{1, 2, \ldots, D\}$ indicating the slot number in the cycle. The signal state changes each slot from t to $t+1$ (or to 1 when $t = D$) independent of the state at the queues. Therefore the state space $(t, \mathbf{q}, \mathbf{a}^1, \mathbf{a}^2, \ldots, \mathbf{a}^F)$ can be decomposed into subspaces $(t, q_f, \mathbf{a}^f)$ for each flow f. The relative values of states $(t, \mathbf{q}, \mathbf{a}^1, \ldots, \mathbf{a}^F)$ can thus be computed flow-by flow for states $(t, q_f, \mathbf{a}^f)$.

State transitions

Before we show how the relative values of states $(t, q_f, \mathbf{a}^f)$ are computed for each flow f, the state transitions under FC are examined. We impose the following assumptions to structure the state transitions in the decision model, such that the arrival information is accurate:

- all cars drive at constant and identical speed,
- cars are not allowed to change lanes,
- no cars reach their destination somewhere along the approach.

Example – Consider the two successive state transitions depicted in Figure 7.1. The first picture shows some initial state $(t, q_f, \mathbf{a}^f) = (\text{"Y1"}, 2, 1, 0, 0, 1, 0)$. Suppose, signal state t is the first yellow slot ("Y1") of f, two cars, labeled A and B, are queued, and another two cars (C and D) are on their way to the intersection. One slot later, the new signal state relates to the second yellow slot ("Y2") to f, and car A has left the queue. The queue state stays $q_f = 2$, since car C has joined the queue. Car D has moved one slot, and is thus only 3 slots away from the end of the queue. After another slot, B has left the queue and the queue consists of car C only, car D is now two slots away from the tail of the queue.

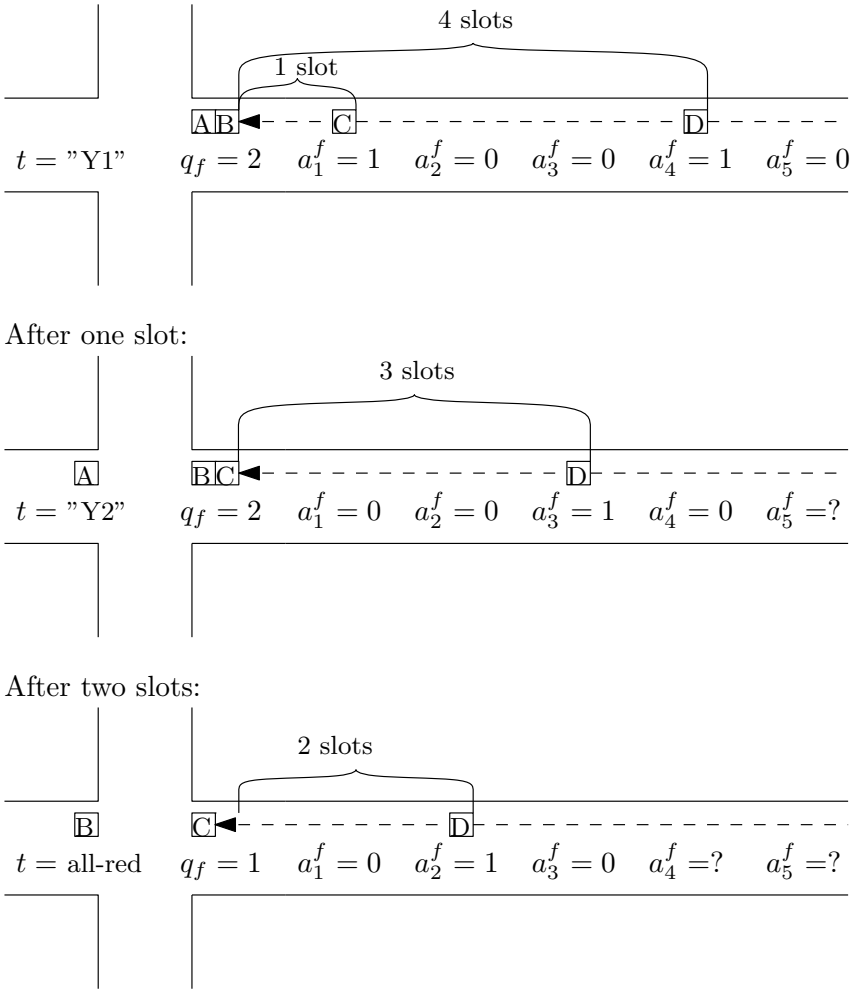


Figure 7.1: Successive transitions under FC with arrival information for a single flow.

Remark – In reality, the distance between car C and D is a bit more than 2 slots, since car C has moved forward in the queue. In the decision model this phenomenon is ignored by assuming vertical queueing: queued cars are assumed to take zero length. However, in the simulation model to be discussed in Section 7.3, we assume horizontal queueing: queued cars take a positive length.

Value iteration for relative values of states $(t, q_f, \mathbf{a}^f)$ under FC

Similar to Equation (6.22) in the previous Chapter, the relative values under FC can be computed by:

$$\lim_{N\to\infty} \sum_{f=1}^{F} \frac{1}{D} \sum_{d=0}^{D-1} \left(V_{N+d}\left(t, q_f, \mathbf{a}^f\right) - (n+d)\cdot g^{(f)} \right), \tag{7.1}$$

where $g^{(f)}$ and $V_{N+d}\left(t, q_f, \mathbf{a}^f\right)$ are approximated by a value iteration algorithm.

When five slots of arrival information is available to flow f, then $V_N^f\left(t, q_f, \mathbf{a}^f\right)$ is recursively computed from Equation (7.2), by successively increasing n, starting with $n = 0$ and $\mathbf{V}_0^f = \mathbf{0}$:

$$\begin{aligned} V_{n+1}^f\left(t, q_f, a_1^f, a_2^f, a_3^f, a_4^f, a_5^f\right) = q_f + \quad & (1-\lambda_f) \quad \cdot V_n^f\left(t, (q_f + a_1^f - \Delta_t^f)^+, a_2^f, a_3^f, a_4^f, a_5^f, 0\right) \\ + \quad & \lambda_f \quad \cdot V_n^f\left(t, (q_f + a_1^f - \Delta_t^f)^+, a_2^f, a_3^f, a_4^f, a_5^f, 1\right), \end{aligned} \tag{7.2}$$

where $\Delta_t^f = 1$ when t grants green or yellow to flow f, and 0 when t implies red to flow f.

The relative value in (7.1) can be used in a one-step policy improvement algorithm to construct an RV policy in which arrival information is used. The gain, or the long-run average costs per slot under FC, $g^{(f)}$, follows from the vector of constants $\lim_{N\to\infty}(\mathbf{V}_N^f - \mathbf{V}_{N-D}^f)$.

Complexity

Since $\mathbf{a}^f$ is a vector with dimension $M(f)$, the state description $(t, q_f, \mathbf{a}^f)$ is high-dimensional. The number of possible vectors $\mathbf{a}^f$ is $2^{M(f)}$, since at most 1 car arrives per slot. Even when Q and D are set as high as 100 cars respectively 100 slots and when $M(f) = 5$ the number of states, $D \cdot (1+Q) \cdot 2^{M(f)}$, is still well below 1 million. Computation of the relative values for all states $(t, q_f, \mathbf{a}^f)$ under FC is then still possible. When $M(f)$ is larger, solving all relative values may consume too much time and memory. Therefore we discuss an alternative approach of computing the relative values.

Online computation of relative values with arrival information

Instead of computing the relative values for all states $(t, q_f, \mathbf{a}^f)$ under FC off-line, one may compute these relative values online. Therefore let $q_{f,m}$ denote the number of cars waiting at queue f after m slots. Starting with $q_{f,0} = q_f$, one may compute $q_{f,1}$, to $q_{f,M(f)}$, as the number of cars waiting at queue f can be predicted using the arrival information $\mathbf{a}^f$ for the coming $M(f)$ slots.

For computing the relative values for all states $(t, q_f, \mathbf{a}^f)$, one first computes for flow f the total waiting costs over the coming $M(f)$ slots, for which arrival information is available. Next, one adds to this the relative value for all states $v^f_{rel}(t', q_{f,M(f)})$ as terminal costs of ending in state $(t', q_{f,M(f)})$ after $M(f)$ slots.

The computation of the waiting costs over the first $M(f)$ slots is relatively easy, since the transitions happen deterministically during the first $M(f)$ slots, assuming the arrival information is accurate. One evaluates the state $(t', q_{f,M(f)})$ that is reached after $M(f)$ slots, by the relative value of state when no arrival information is available: i.e. $v^f_{rel}\left(t', q_{f,M(f)}\right)$ as defined in (6.22) in the previous chapter. In the next two subsections, the online evaluation of decisions under the RV policies is explained in more detail.

Remark – Online computation yields the same relative values but is favored above the off-line computation of $V^f_n(t, q_f, a^f_1, \ldots, a^f_{10})$, when $M(f)$ is large. When $M(f) = 10$, off-line-computation would imply the computation of about 10^7 elements $V^f_n(t, q_f, a^f_1, \ldots, a^f_{10})$ for each flow f. Whereas online computation requires only 10,000 relative values $v^f_{rel}\left(t', q_{f,M(f)}\right)$ to be evaluated for each flow f, as if no arrival information is available. The number of operations to be executed online to evaluate a single time jump is only $\mathcal{O}(M(f))$.

7.2.2 Extension of RV1 and RV1M with arrival information

Using a one-step policy improvement algorithm with initial policy FC, we have derived in the previous chapter the policies RV1 and RV1M. These policies are now extended such that arrival information is included in the selection of the best time jump τ.

The number of slots of arrival information is added to the name of the policy: e.g. RV1(5) assumes that accurate arrival information is available for all flows for the next 5 slots; under RV1(5,0,5,0) 5 slots arrival information is used for combinations 1 and 3, while no information is available to C_2 and C_4.

Extension of RV1

In order to find out what the best decision τ is, first the waiting costs over the next $M(f) = 5$ slots are determined, as illustrated in Figure 7.2. Next, the relative value of the state visited after 5 slots is added as terminal costs.

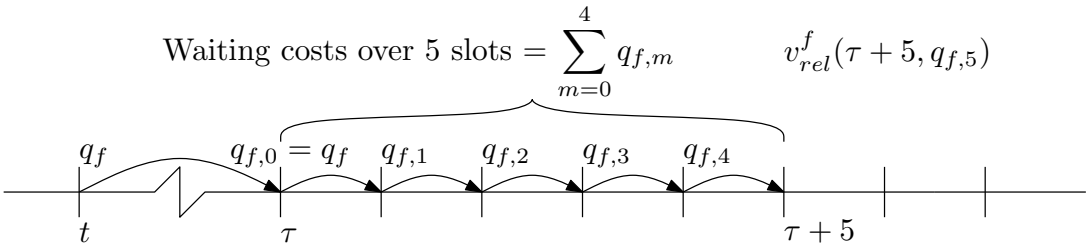


Figure 7.2: Evaluating RV1(5): RV1 with arrival information for the next 5 slots.

The state visited after $M(f) = 5$ slots is derived as follows, using the notation $q_{f,m}$ for the queue length after m slots. The initial queue length $q_{f,0}$ is q_f and changes after 1 slot into $q_{f,1} = \left(q_{f,0} + a_1^f - \Delta_\tau^f\right)^+$, with Δ_τ^f is the maximal number of the departures from queue f within slot τ, which is in our case 1 when τ is a departure slot to flow f and 0 otherwise. Note that by this definition a car that arrives at an empty queue will depart during the very same slot when the respective lights are a green or yellow. Since the car is not queued it contributes zero waiting costs.

The queue length m slots after a time jump to slot τ, $q_{f,m}$, is recursively set by:

$$q_{f,m} = \begin{cases} \left(q_{f,m-1} + a^f_m - \Delta^f_{\tau+m-1}\right)^+ & \text{if } m \in \{1,2,\ldots,M(f)\} \\ \\ q_f & \text{if } m = 0, \end{cases} \tag{7.3}$$

with $\Delta^f_y = \Delta^f_{y-D}$ when $y > D$ such that $y - D \in \{1,2,\ldots,D\}$.

The waiting costs at queue f over the first $M(f)$ time slots is simply the sum $\sum_{m=0}^{M(f)-1} q_{f,m}$. After these $M(f)$ slots $q_{f,M(f)}$ cars are waiting at queue f, and the signal state is $\tau + M(f)$.

The relative cost value to flow f of starting in state $(\tau, q_f, \mathbf{a}^f)$ thus becomes:

$$\sum_{m=0}^{M(f)-1} q_{f,m} + v^f_{rel}\left(\tau + M(f), q_{f,M(f)}\right). \tag{7.4}$$

The relative value of state $(\tau, q_f, \mathbf{a}^f)$ can be computed online for each flow, once $v^f_{rel}(t, q_f)$ is computed and stored for each state (t, q_f) according to (6.22).

RV1(·) actions

The selected time jump τ in state $(\tau, q_f, \mathbf{a}^f)$ under RV1(·) follows from:

$$\arg \min_{\tau \in \mathcal{T}(t,\mathbf{q})} \sum_{f=1}^{F} \left(\sum_{m=0}^{M(f)-1} q_{f,m} + v^f_{rel}\left(\tau + M(f), q_{f,M(f)}\right) \right). \tag{7.5}$$

The cyclic version of RV1(·) with arrival information is simply obtained by changing the acyclic action space $\mathcal{T}(t,\mathbf{q})$ in Equation (7.5) by $\mathcal{T}^{\mathrm{C}}(t,\mathbf{q})$.

Remark – Note that the process of looking ahead a number of slots is similar to the evaluation of jumps under RV2. Big difference is that now the first $M(f)$ transition happen deterministic and are evaluated online. Furthermore, we do not need to subtract $m = M(f)$ times the gain, $g^{(f)}$, since $M(f)$ does not depend on the decision τ. (However, as we will see in the next subsection, one has to compensate for unequal horizon lengths when extending RV2.)

Extension of RV1M

The acyclic policy RV1M can also be extended to include arrival information. Under RV1M the evaluation of the waiting costs over the first $M(f)$ slots is done in a very similar way. The notations become more evolved, since under RV1M one optimizes over all possible cycles ϕ. Since slot numbers $1, 2, \ldots, D$ relate to the base cycle 1-2-...-C - 1-2-etc., the slots are shifted by $\delta_f(\phi)$ slots under cyclic order ϕ , as discussed in Section 6.5.2.

Consequently we should replace τ by $\tau + \delta_f(\phi)$ in the definition of $q_{f,m}$ in (7.3):

$$q_{f,m} = \begin{cases} \left(q_{f,m-1} + a^f_m - \Delta^f_{\tau+\delta_f(\phi)+m-1}\right)^+ & \text{if } m \in \{1, 2, \ldots, M(f)\} \\ \\ q_f & \text{if } m = 0. \end{cases} \tag{7.6}$$

RV1M(·) actions

The selected time jump τ in state $(\tau, q_f, \mathbf{a}^f)$ under RV1M(·) follows from Equation (7.7):

$$\arg \min_{\substack{\tau \in \mathcal{T}(t,\mathbf{q}) \\ \phi \in \Phi(\tau)}} \sum_{f=1}^{F} \left(\sum_{m=0}^{M(f)-1} q_{f,m} + v^f_{rel}(\tau + \delta_f(\phi) + M(f), q_{f,M(f)}) \right). \tag{7.7}$$

7.2.3 Extension of RV2 and RV2M with arrival information

The inclusion of arrival information into RV2 is much more complicated, at least in its notations. Remember under RV2, two time jumps τ_1 and τ_2 are considered. Instead of presenting an algorithmic description we illustrate in Figure 7.3 how to evaluate a decision (τ_1, τ_2). We distinguish two cases, regarding the amount of arrival information $M(f)$ in relation to the time $(r - \tau_1)$ until the next green period, which would start at slot r when not interrupted by τ_2:

Case I. $M(f) > r - \tau_1$: the horizon of arrival information is longer than the time between the first and the second jump,

Case II. $M(f) \leq r - \tau_1$: no arrival information is available after the jump to τ_2.

In Case I, one simply determines the waiting costs over the next $M(f) = 5$ slots and add to it the relative value of the state in which one ends after the $M(f) = 5$ slots. To evaluate the first part one has to be aware of the second time jump that breaks FC.

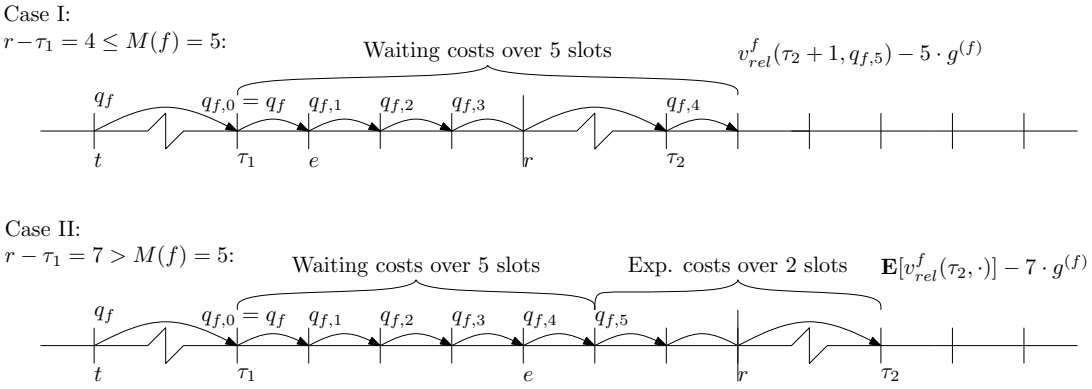


Figure 7.3: Evaluating RV2(5): RV2 with arrival information for the next 5 slots.

In Case II, the evaluation of (τ_1, τ_2) consists of three parts. For the first part arrival information is available and each transition happens deterministically, for the second part one computes the expected waiting costs over the slots until the second time jump τ_2, and finally the third part is the expected relative value of the states in which the system may end after $r - \tau_1$ slots. In fact, the latter two parts correspond to the evaluation of the decision $(\tau_1 = r - 2, \tau_2)$ under RV2, starting in state $(r - 2, q_{f,5})$. To compensate for unequal horizon lengths, one has to subtract $M(f)$ times the average cost per slot, $g^{(f)}$.

Both RV2 and RV2M can be extended this way, but we skip the formal notation of how the actions are selected.

7.3 The simulation model

In the next section the extended RV policies that include arrival information are tested by simulation. In this section, the simulation model of the previous chapter is extended, such that arrival information is available in the model. The extended simulation model is more realistic: cars are no longer stacked vertically, which was the case in the previous chapter. Instead cars are queued horizontal and we assume each queued car takes a fixed length, of say 7 meters. In addition, we allow cars to travel at different speeds.

In the next subsections we discuss principal components and characteristics of the simulation model.

7.3.1 Discrete time

As before the simulation happens in discrete time, since the lights are adjusted every 2 seconds (=1 slot). Information available, at the start of each slot, are the state of the traffic light, the length of the queues and the estimated times at which upstream driving cars arrive at the queue.

7.3.2 Car arrivals

In the simulation model randomly generated car arrivals enter a lane at some fixed distance from the respective stopping line, say at 500 meter. When cars travel at a speed of 50 km/h, it takes a car 18 slots to travel to the downstream stopping line. For each lane car arrivals are generated according to a Bernoulli experiment (0 or 1 arrival per slot per flow). Since we consider an intersection in isolation, we do not consider batch or platoon arrivals.

When an approach consist of three parallel (adjacent) lanes (e.g. for through-traffic and left and right turning traffic), maximal three cars may arrive at the approach per slot. Cars generated in the same slot at parallel lanes and that travel at identical speed, keep driving next to each other till they join their respective queues. When car speeds differ, a car may overtake another car that drives on another lane. However, a fast driving car may not change lane to overtake a slow moving car that drives on the same lane. In fact we do not allow cars to change lane, except when crossing an intersection. All generated cars stay in the system until they have passed the stopping line: no car has a destination along the approach to the intersection.

7.3.3 Car speed and arrival information

Upon entering the system a car gets assigned a *'desired' traveling speed*. The actual traveling speed is lower, when a car has to slow down because of slower moving traffic directly in front of it. A car keeps on driving at its desired speed until it joins a queue, or until it has to adjust its actual speed to slow moving cars directly in front of it. At any time cars keep a safe driving distance of 2 seconds (=1 slot). Speed adjustments are made instantaneously in zero time.

In cases where traveling speeds do vary, desired traveling speeds are generated randomly from a triangular distribution $\text{Tri}(a = 40, b = 50, c = 60)$, with mean and most likely speed set to 50 km/h, the minimal desired speed is 40km/h and maximum speed is 60km/h. According to this distribution: about 50% of the car drivers wish to drive at a constant speed between 47 and 53 km/h, 75% of the car drivers has a desired speed between 45 and 55 km/h. If not hampered by other cars a car that drives 60km/h takes only 30 seconds (15 slots) to reach the stopping line, which is located 500 meters downstream. A slowest moving car takes 45 seconds to travel the same distance.

The extended RV policies, introduced in this chapter, require estimated arrival times as input. To be more precise for a number of future slots ($M(f)$), one needs to know whether a car is expected to arrive or not at each of flows. Therefore the position of each car in the lanes is tracked by which the controller estimates the time it takes until a car joins a queue. Since cars are queued horizontally, queue spill back happens: the end of the queue moves upstream when cars join the queue and the lights are red. In estimating the arrival time, one thus considers the current length of the queue in meters, the actual position of the car and of all cars in front of it. When the driving speed of a car is not known to the controller, the time it takes before the car joins the queue is estimated based on the mean desired speed. Again, the length of a queue is computed as if queued cars take 7 meters each.

7.3.4 Departure process

Cars leave a queue one-by-one keeping an traveling distance of 1 slot. When passing the stopping line cars accelerate in zero time to their desired traveling speed and immediately leave the system. Blocking of the intersection by other cars does not happen. Furthermore, it is assumed that conflicting flows are not served simultaneously, hence, whenever a light shows green or yellow one car (if present) may leave the queue. (We have discussed already at the end of the previous chapter that the model can be modified such that conflicting

flows are part of the same combination.) All other cars queued shift one position in the queue. If a queued car takes 7 meters, queued cars travel 7 meters a slot (3.5 m/s or 12.6 km/h), during green and yellow slots.

7.3.5 Queueing process

The simulation model is microscopic, since individual cars are distinguished and their position and speed in the lanes is tracked. The distance of a car to the queue changes over a slot, since the car itself moves towards the queue, but also the end of the queue moves up and downstream when cars are joining or leaving the queue. The arrival process of cars at the end of the queue is then not quite a Bernoulli process, also because cars travel in platoons caused by slow-moving traffic that delays other cars. In the Markov chain models by which feasible decision are evaluated, we stick to Bernoulli arrival processes.

An ambiguity – The rate at which cars arrive at the queue is not constant, when queued cars take non-zero length. In the MC model cars arrive at queue f at a fixed rate λ_f, while in the simulation cars enter the lane at an fixed upstream point at rated λ_f. Although the cars enter lane f at a fixed rate, they do not arrive at a fixed rate at the queue. The arrival rate is a bit higher when the position of the Tail of the Queue (ToQ) moves upstream, i.e. when the lights are red. Similarly the effective arrival rate is somewhat lower when the ToQ moves downstream, i.e. when the lights are green. The differences are expected to be small, but present as illustrated by the next example

Example – Suppose cars arrive at rate $\lambda = 0.2$ cars per slot at a lane and get assigned a constant speed of 50km/h, which equals $13\frac{8}{9}\text{ms}^{-1}$ or $27\frac{7}{9}$m. per slot. Then on average every 5 slots (10 seconds) a car enters the lane. Suppose two cars driving along the lane keeping a distance of 5 slots, which corresponds to $5 \cdot 27\frac{7}{9} = 139$ meters (from front bumper to front bumper). Just before the first car arrives at the queue the second car is still about 139 meters away from the ToQ. Just after the arrival of the first car at the ToQ, the queue length increases by seven meters when the light shows red. Hence the distance between the second car and the ToQ drops instantaneously, from 139 to 132 meters. Hence the second car reaches the end of the queue after about 9.5 seconds instead of after 10 seconds. When the light shows red the effective expected inter-arrival time is thus 9.5 seconds, which relates to an arrival rate of 0.21 instead of 0.2. When the lights show green the arrival rate is lower than 0.2 since the ToQ moves downstream.

The queue dynamics cause slight differences between the average waiting time reported by the simulation model and those computed by solving Markov chains. The Markov chain approach underestimates the average waiting time, since it assumes a constant arrival rate. The results are small and may be of no practical importance, given that we have simplified the process of joining a queue: cars may adjust their traveling speed based on the color of the light. When in the simulation model cars speeds are identical and queued cars take zero length, the results of the MC and the simulation model coincides.

7.3.6 Waiting time definition

Throughout the Chapters 6 to 8 we use the same definition of waiting time: the time spent in the queue. Nevertheless, we pay here some attention to this definition, since individual waiting times depend on whether cars are queued vertically, as in the previous chapter, or horizontally (as in this chapter).

For example, consider a car drives 55 meters upstream from the stopping line at speed 50km/h, and 4 cars are queued. When queued cars are queued vertically, i.e. queued cars take zero length, it will join the queue after 2 slots. However, when queued cars take 7 meters each and the lights show red for the coming slot, then the ToQ is located $4 \cdot 7 = 28$ meters upstream. Consequently the car is just 27 meters away from the ToQ, and it will join the queue already after 1 slot. The average waiting time for this car is consequently 1 slot higher than under vertical queueing. In queueing theory one usually assumes vertical queueing; then the waiting time is computed as if cars drive at normal speed to the stopping line.

We believe that the given definition of waiting time under horizontal queueing resembles the waiting time experienced by car drivers, although one may want to discriminate between being queued in front of a red light and being queued in front of a green light. Modifications at this point are possible but not considered.

7.4 Base cases and results

The extended RV policies that include arrival information are tested for the F12C4 infrastructure (see Figure 7.4) with 4 approaches each consisting of 3 lanes of 500 meters length. The following data further characterize the base case:

- Car arrivals happen 500m. upstream by Bernoulli experiments with probabilities λ_f,
- The arrival rates are identical to all flows f: $\lambda_f = \lambda$,
- Individual desired cars speeds (in km/h) are drawn from Tri(40,50,60),
- Five slots of arrival information is used for each flow, i.e. the arrival times are estimated for cars that are $5 \cdot 27\frac{7}{9} \approx 139$ meters away from the ToQ (a car that is between 111 and 139 meters from the ToQ is modeled to arrive after 5 slots when no cars travel in front of it and the lights stay red for 5 slots; when 4 cars travel in front of it, the car reaches the ToQ already after 4 slots).

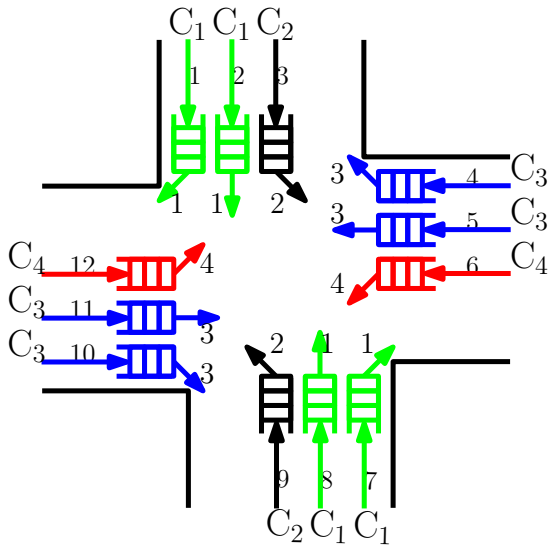


Figure 7.4: "F12C4" serve 12 flows in 4 asymmetric combinations

We investigate for varying loads $\rho = 4\lambda$ the impact of arrival information on the overall average waiting time. For $\rho = 0.6$ we check whether thin combinations of only 2 queues (C_2 and C_4) suffer a higher waiting time than thick combinations of 4 queues (C_1 and C_3).

A sensitivity study with respect to the number of slots of arrival information, and asymmetric arrival rates follows in Section 7.5.

7.4.1 Vary workloads

In Table 7.1 we report the overall average waiting time in seconds at loads $\rho = 0.4$, 0.6 and 0.8. The arrival rates are identical: 0.1, 0.15, and 0.2 respectively. As expected, the average waiting time for FC, RV1 and RV1M are somewhat higher than those reported in the previous chapter in Table 6.4, because now cars are queued horizontally and cars arrive in platoons due to speed differences. The difference is primarily the traveling time from the ToQ to the stopping line when traveling at normal speed. The difference is greater when the workload of the intersections is higher, since the average queue length is higher.

Taking into account 5 slots of arrival information yields, under cyclic RV1(5), a reduction of the overall long-run average waiting time by 6% when the workload is 0.4. The reduction is only 3% when the workload is 0.8.

Under acyclic control similar improvements are achieved by considering 5 slots of arrival information. Compared to cyclic RV1, the waiting time is reduced by 21% under RV1M(5) when the workload is 0.4. Again the difference is much smaller (3.5%) when the workload is 0.8. Just as we have seen in the previous chapter, the difference between cyclic and acyclic control is small when the workload is high.

To save space, the results for RV2(5) and RV2M(5) are not reported. Furthermore, the average waiting time under these policies are almost identical to those under RV1(5) and RV1M(5).

Table 7.1: Mean waiting time (in sec.) for partly-asymmetric F12C4 at varying loads (ρ).

Rule	$\rho = 0.4$		$\rho = 0.6$		$\rho = 0.8$	
Cyclic policies:						
RV1	13.9		20.2		44.2	
RV1(5)	13.1	−6.0%	19.1	−5.1%	42.8	−3.1
FC	15.5	+11.8%	24.9	+23.6%	53.4	+21.0%
FC cycle length (in sec.)	32		40		88	
FC departure times (in sec.)	(6, 6, 6, 6)		(8, 8, 8, 8)		(20, 20, 20, 20)	
Acyclic policies:						
RV1	12.8	−7.7%	19.7	−2.2%	44.2	0.0%
RV1M	12.2	−12.5%	19.1	−5.1%	44.0	−0.4%
RV1(5)	11.9	−14.6%	18.7	−7.4%	42.8	−3.0%
RV1M(5)	11.0	−21.1%	17.9	−11.4%	42.6	−3.5%

7.4.2 Average waiting time per queue at load 0.6

For the base case with workload 0.6 (arrival rates are 0.15), we are interested in whether the thick combinations (C_2 and C_4) benefit more from having arrival information than thin combinations (C_2 and C_4). Therefore consider Table 7.2, which reports the average waiting time for each combination under the different policies.

Accounting for 5 slots of arrival information, is in favor of both thick and thin combinations: the average waiting time at both combinations is reduced by about 1 second, which is about 5%. Under acyclic control the thicker combination benefit slightly more from having arrival information. For example, under RV1M(5) the waiting time at C_1 and C_3 has been reduced by 1.4 seconds compared to RV1M, which is about 8.5%, the waiting at the thin combinations is reduced by only 4.5% (1.1 seconds).

Remark - Although not reported in the table, of all cyclic policies, RV2(5) gives the lowest average waiting time at the thick combinations: 15.3 seconds compared to 17.5 seconds under RV1(5). Further we observe that acyclic control is in favor of the combinations C_1 and C_3 with 4 queues, while the waiting time at C_2 and C_4 are a bit higher but still lower than under FC. The lowest overall mean waiting time when including 5 slots arrival information comes from RV1(5) and RV1M(5).

Table 7.2: Mean waiting times (in sec.) for F12C4 ($\rho = 0.6$ and $\lambda = 0.15$).

Rule	*EW* overall		*EW* C_1, C_3	*EW* C_2, C_4
Cyclic policies:				
RV1	20.2		18.5	23.5
RV1(5)	19.1	−5.1%	17.5	22.5
FC dep. times 8, 8, 8, 8 sec.	24.9	+23.6%	24.8	24.9
Acyclic policies:				
RV1	19.7	−2.2%	17.5	24.2
RV1M	19.1	−5.1%	16.5	24.4
RV1(5)	18.7	−7.4%	16.3	23.4
RV1M(5)	17.9	−11.4%	15.1	23.3

7.5 Sensitivities

In this section we check for the F12C4 case with load 0.6,

- how the results change when more or less slots of arrival information are used,
- the impact of having no arrival information for some flows and
- the importance of arrival information to a combination that is three times as thin than the other (two or three) thick combinations.

7.5.1 Amount of arrival information

On the one hand, one may expect that decisions will improve when more arrival information is available, and used, in controlling the lights. On the other hand, a few slots of arrival may be sufficient, since the state of the lights can be revised at the start of any slot (except during the switching phase). Since during the two yellow slots of a switching phase cars may still pass the stopping line, using two slot of arrival information, as in RV1(2), may result in a significant improvement of RV1.

The RV policies evaluate decisions as if future decision follow from FC, instead of from an RV policy. Hence considering too many slots arrival information might result in decision that are good for the future arrivals, but that may be sub-optimal for the cars currently waiting. Since decisions can be revised every decision epoch, looking too many slots ahead may result in bad decisions.

For the F12C4 case with workload 0.6, we test the extended RV policies for varying number of slots ($M(f) = m$) of arrival information: $m = 0$–15. The resulting overall average waiting time is plotted in Figure 7.5. The fixed cycle on which the RV policies are based has cycle length 20 slots and each effective green period last 4 slots.

As expected, taking only 2 slots of arrival information gives already a significant reduction in the overall average waiting time. Considering a third slot of arrival information is beneficial for the RV1 policies (RV1(m) and RV1M(m)), but not under the RV2 policies (RV2(m) and RV2M(m)). Apparently, under an RV2 policy with exactly three slots of arrival information the decision (τ_1, τ_2) is too much dominated by τ_2. Using four (or more) slots of arrival information gives much better results.

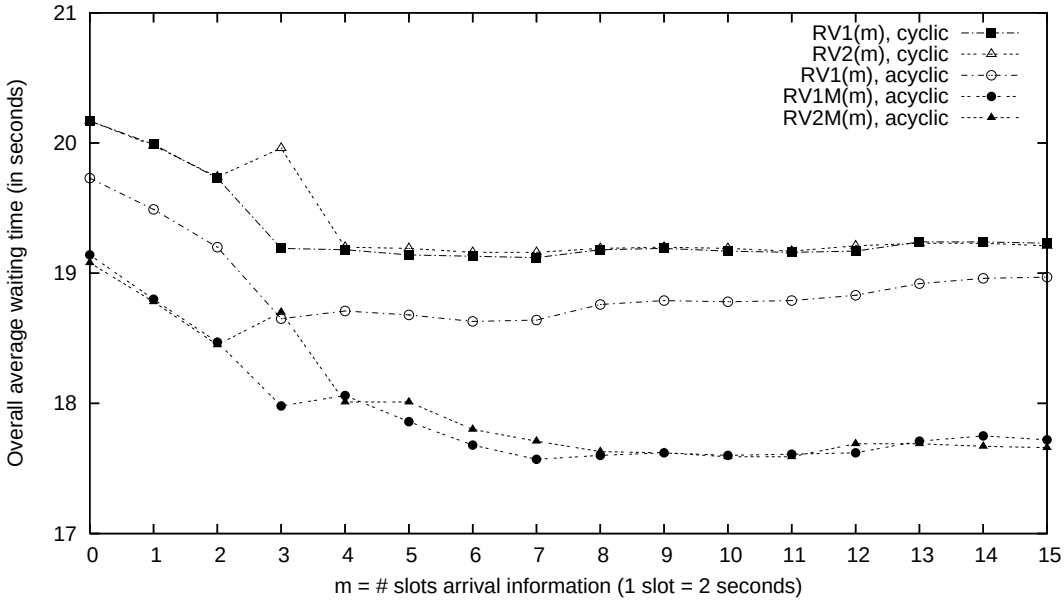


Figure 7.5: Impact of the amount of arrival information on long-run average waiting time.

With three slots of arrival information, the queue state after three slots is known exactly. Changing from green to yellow means that the exact queue length is known upon the jump to τ_2. Keeping the lights green for at one more slot, implies that the state is not exactly known when a next state τ_2 is to be selected. The choice of τ_1 determines, whether the queue state is known or upon the selection of τ_2. The uncertainty in the queue state may affect the choice of τ_1. This uncertainty is not present when four slots of arrival information is used. This explains the peak of the curves of RV2(m) and RV2M(m), at $m = 3$.

For most RV policies taking 4 or 5 slots of arrival arrival information gives already the best results. Considering more slots yields hardly any improvement, except for the RV1M and RV2M policies that benefit slightly from having a few more slots arrival information. Since the RV2(m) policies hardly improve the RV1(m) policies, we do not report on them in the remainder of this chapter.

7.5.2 Skewness in amount of arrival information

Thus far it was assumed that all flows have an equal number of slots of arrival information. In this and the next section cases are studied where each flow has either 5 slots of arrival information or has no arrival information at all.

Consider the party-asymmetric F12C4 case with arrival rates equal to $\lambda = 0.15$. For all flows, 5 slots of arrival information is available, except for C_2, which has no arrival information at all. The number of slots of arrival information is included in the name of a policy: e.g. under RV1(5,0,5,5) uses 5 slots of arrival information but not for C_2.

For being a thin combination with a low arrival rate, one may expect that the waiting time at C_2 is highest. Since arrival information is available for the two thick combination, we expect that the waiting time at the thick combinations is lowest. In Table 7.3 the waiting times at all queues are reported for both the cyclic and the acyclic RV policies. As a benchmark we report also FC and the RV policies that do not use any arrival information. For cyclic control RV1(5) is preferred above RV1. The best policy is RV1M(5), which improves RV1 by 10.1%. Taking 5 slots of arrival information into account and allow acyclic control seems to help reducing the long-run average waiting time.

Under all policies the average waiting time is indeed lowest at the thick combinations. The waiting time at the thin combination C_2, which has no arrival information, is under RV1(5) hardly higher than under RV1, which ignores any arrival information for all flows. All other flows benefit under RV1(5) from using arrival information. Seemingly, a combination suffers more from having a low number of flows, and consequently a low weight, than from lacking arrival information.

Table 7.3: Mean waiting times (in sec.) for partly-asymmetric F12C4 ($\rho = 0.6$ and $\lambda = 0.15$): impact of no arrival information for C_2.

Rule	$E(W)$ overall		EW C_1,C_3	EW C_2	EW C_4
Cyclic policies:					
RV1	20.2		18.5	23.5	23.5
RV1(5,0,5,5)	19.3	-4.1%	17.5	23.6	22.4
FC dep. times 8, 8, 8, 8 sec.	24.9	+23.6%	24.8	24.9	25.0
Acyclic policies:					
RV1	19.7	-2.2%	17.5	24.2	24.3
RV1M	19.1	-5.1%	16.5	24.4	24.4
RV1(5,0,5,5)	18.9	-6.4%	16.3	24.2	23.5
RV1M(5,0,5,5)	18.1	-10.3%	15.2	24.6	23.3

7.5.3 Skewness in arrival rates and arrival information

When arrival information is lacking for a very thin combination, i.e. a combination with a relatively low load, then cars arriving at that combination may experience high waiting times. This hypothesis did not hold for the case, studied in the previous subsection, where all arrival rates and all effective green times were identical. Now we check this hypothesis, for the following two fully-asymmetric F12C4 cases with workload 0.6:

Case I: the arrival rate at C_2 is only a third of that at the other queues: hence the thick combinations are six times as thick. Furthermore all flows except those in C_2 benefit from 5 slots arrival information.

Case II: the arrival rates at the North and South approach are a third of those at the West and East approach. Furthermore the West and East approach give 5 slots of arrival information, but the North and South approaches give no arrival information at all.

Case I – Combination 2 is third as busy and provides no arrival information

In this case, the arrival rates of flows in $\mathcal{C}_2$ are only 3/50, whereas for the other flows the arrival rates are 9/50. The total load is thus 0.6 (= (9 + 3 + 9 + 9)/50). Under RV1(5,0,5,5), RV1(5,0,5,5) and RV1M(5,0,5,5) all flows except those in C_2 have 5 slots arrival information.

According to the Optimal-fixed-cycle algorithm, the best configuration of FC has cycle length 54 seconds (26 slots), with effective green times or departure times 14, 6, 14, and 12 seconds for combinations 1 to 4 respectively. The effective green times of C_1 and C_3 is identical since they have the same load: i.e. both combinations consist of 4 flows with identical arrival rates. C_4 has the same arrival rate but consist of only 2 flows and is thus less thick: therefore its green time is a bit lower than that of C_1 and C_3. The very thin combination, C_2, has a very short effective green time of only 6 seconds. Its red period last $54 - 6 = 48$ seconds; that is 8 times as long as the green period.

On the one hand, there are three reasons for expecting C_2 to have a higher average waiting time than the other combinations:

- C_2 consist of only 2 flows, whereas C_1 and C_3 consists of 4 flows each,
- C_2 is six times as thin as both C_1 and C_3,
- C_2 has no arrival information, whereas all other flows may benefit from 5 slots of arrival information.

On the other hand, the green period of C_2 in FC is short, which means the queue lengths are to be kept short to avoid incurring high relative cost values for flows in $\mathcal{C}_2$. Hopefully, the average waiting times for C_2 under the RV policies do not exceed those of FC.

In Table 7.4, one reads the average waiting time under the different polcies with and with no arrival information. Under cyclic control, the waiting time of C_2 has increased by only 1 second as a result of using arrival information, while the waiting time at C_4 is reduced by 2 seconds. Under acyclic control, the average waiting time at C_2 is about twice as high as under cyclic control. Clearly, C_2 suffers, in favor of a reduction in the waiting time at all other flows. The difference in the overall average waiting time between using 5 slots of arrival information or none, is also under acyclic control only 1 or 2 seconds: the waiting at 10 queues has reduced by 1 to 2 seconds, whereas the waiting time at C_2 increases by less than 1 second.

Compared to FC, the overall average waiting can be reduced by 27%: from 24.8 seconds under FC to 18.1 seconds under RV1M(5,0,5,5). For cyclic control the reduction may be 19%: from 24.8 to 20.1 seconds under cyclic RV1(5,0,5,5).

Table 7.4: Mean waiting times (in sec.) for a fully-asymmetric F12C4 case ($\rho = 0.6$ and $\lambda = (\frac{9}{50}, \frac{3}{50}, \frac{9}{50}, \frac{9}{50})$): the impact of low arrival rates and no arrival information to C_2.

Rule	*EW* overall		*EW* C_1,C_3	*EW* C_2	*EW* C_4
Cyclic policies:					
RV1	21.1		20.5	24.7	22.0
RV1(5,0,5,5)	20.1	-4.4%	19.5	25.8	20.6
FC dep. times 14, 6, 14, 12 sec.	24.8	+17.6%	22.7	29.1	31.9
Acyclic policies:					
RV1	19.2	-8.9%	17.7	43.6	18.7
RV1M	19.0	-10.0%	16.0	52.3	19.9
RV1(5,0,5,5)	18.5	-12.3%	17.0	44.4	17.5
RV1M(5,0,5,5)	18.1	-14.2%	15.3	53.1	17.7

Case II – North and South approaches (C_1 and C_2) provide no arrival information and are only a third as busy

The workload set by the North and South approaches is a third of the workload at the East and West approach. To get a total workload of 0.6 the arrival rates are set to 3/40 at flows in $\mathcal{C}_1$ and $\mathcal{C}_2$, and 9/40 at all other flows. In Table 7.5 we report the waiting times for this fully-asymmetric F12C4 case. The best FC found by the Optimize-fixed-cycle algorithm grants green or yellow to C_1, C_2, C_3, and C_4, for respectively 6, 6, 16, and 14 seconds (or 3, 3, 8, and 7 slots). The cycle length is thus 50 seconds (25 slots). Under FC the waiting time is 23.2 seconds. The overall average waiting time drops by 25% from 23.2 under FC to 17.2 seconds under RV1M(0,0,5,5).

The conclusions are quite similar to those for Case I. Due to the asymmetry in the arrival rates, flows in C_2 experience a high average waiting time under acyclic control. Since C_1 consists of twice as many flows as C_2, the difference in the average waiting time between cyclic and acyclic control is much less to C_1 than to at C_2. The lowest waiting times are experienced at C_3 and C_4 since these combinations contribute most to the workload of the intersection and because they benefit from having arrival information.

The impact of taking into account 5 slots arrival information reduces the waiting time at C_3 and C_4, while it hardly increases the waiting times at C_1 and C_2. The reduction in the overall average waiting time is moderate: about 5%.

Table 7.5: Mean waiting times (in sec.) for a fully-asymmetric F12C4 cases ($\rho = 0.6$ and $\lambda = (\frac{3}{40}, \frac{3}{40}, \frac{9}{40}, \frac{9}{40})$): impact of low arrival rates and no arrival information to C_1 and C_2.

Rule	*EW* overall		*EW* C_1	*EW* C_2	*EW* C_3	*EW* C_4
Cyclic policies:						
RV1	20.4		22.9	24.9	18.7	20.6
RV1(0,0,5,5)	19.4	-4.9%	23.2	25.3	17.4	19.1
FC dep. times 6,6, 16, 14 sec.	23.2	+13.6%	30.0	30.0	18.8	25.2
Acyclic policies:						
RV1	18.8	-7.9%	22.5	33.9	15.6	17.8
RV1M	18.1	-11.3%	24.0	36.1	13.8	16.8
RV1(0,0,5,5)	18.0	-12.0%	23.0	34.0	14.4	16.4
RV1M(0,0,5,5)	17.2	-15.7%	24.8	36.9	12.5	15.0

7.6 Conclusions

The dynamic control of traffic lights using information on the queue lengths and on estimated arrival times of near-future car arrivals, can be formulated as an MDP. Even when the arrival information is omitted, the computational complexity is too high to solve realistic cases. With the inclusion of, say, 5 slots of arrival information for each flow, even small problems, e.g. an F4C2 case, cannot be solved. For practical problems one thus relies on heuristics and approximate solutions.

When cars drive at constant speed, are not allowed to change lanes, and are queued vertically, the state space under FC is decomposable into subspaces for each flow. This decomposition is exploited in the analysis of the underlying Markov chain. In principle the relative values for states including the arrival information can be computed flow-by-flow. Instead we exploit the fact that transitions happen (approximately) in a deterministic fashion. Therefore the (infinite) horizon is split into two parts: for the first part arrival information is available, for the second part relative values can be computed to evaluate the future under FC with no arrival information. Over the first part of the horizon, the transitions under FC from one state to the next happen deterministically; thus one may compute easily the waiting costs over this period. To evaluate the second part, the relative values under FC are used, as explained in Chapter 6.

By a one-step policy improvement algorithm, one selects the best time jump in FC given the waiting cost over the period with arrival information and the relative value of the state at which one ends under FC. All of the RV policies introduced in Chapter 6 can be extended to deal with arrival information. The extension of the RV2 policies is somewhat complicated. From simulation of the F12C4 intersection in isolation, we conclude:

- Only a few slots of arrival information suffices to achieve a reduction in the overall average waiting time by 3 to 6%.
- Compared to FC the new policy reduces the average waiting time by 20 to 30%.
- When the workload is low, the reduction is greater than when the workload is high.
- The cyclic RV1 policy with arrival information shows low waiting times; even in cases where some flows have no arrival information.

The conclusions may hold specifically for the selected F12C4 infrastructure. Results for the F4C2 intersection are presented in the next chapter, in which networks of intersections are discussed.

Chapter 8

Networks of intersections

8.1 Introduction

When intersections are part of a network, departures from one intersection may be arrivals to another intersection. Typically the departure process is far from uniform, as the process is controlled by an on/off server: the traffic lights that changes from green to yellow to red. When all cars leaving a queue proceed in the same direction and travel at constant speed, cars travel in platoons to a down-stream intersection.

Under *coordinated FC* a *network cycle* with cycle length D to all intersections is defined. The length and start of the green periods at the intersections can be set such that the long-run average waiting time is minimal. Therefore it may help to set so-called *green waves*: then cars leaving one intersection do not need to wait at the downstream intersection as its lights change just-in-time into green. In the practice, green waves may be hard to set and are easily lost as platoons get dispersed due to speed differences of cars and as some cars may turn left or right.

Coordinated FC through a network cycle is only possible for a limited number of cycle lengths D. Consequently coordinated FC may yield a higher average waiting time than uncoordinated FC, with locally optimal cycle lengths D_i for each intersection i. In this chapter we discuss how our decomposition approach applies to the control the lights in a network. We test by simulation whether coordinated and uncoordinated FC can be improved by dynamically adjusting traffic lights using information on the queue lengths and (if available) information on estimated arrival times.

8.2 One-step policy improvement for network control

8.2.1 Centralized coordinated control

In the Chapters 6 and 7 several policies are developed, that greatly improve FC by applying a policy improvement algorithm. For the control of network of intersection one may choose as initial policy any FC: coordinated or uncoordinated.

Coordinated FC requires a good network cycle. Therefore one needs to optimize over the cycle length D, the length of the effective green periods $d_{i,s}$ of each combination of flows s, and the start of a cycle for each intersection i, such that the long-run average waiting time per car is minimized. In general, finding an optimal cycle is a complicated task, since it depends, amongst other, on the distance between the intersections, on the (distribution of the) traveling speeds, on the routes selected by cars and on the arrival rate of each flow, as will be demonstrated in Section 8.3.

Once a network cycle is defined, a one-step policy improvement of coordinated FC can be done in a similar way as for an intersection in isolation. For each flow the slots of the network cycle are green, yellow, or red slots. The relative appreciation or relative value of a slot can be determined flow-by-flow by solving the underlying Markov chains. Every decision epoch one may select the best position in the cycle. However, there are two reasons why such an approach may not work well:

1. The network cycle can be interrupted only when at none of the intersections a light shows yellow; there is no choice when a switching phase is executed at one or more intersections, i.e. the lights of one (or more) combination(s) show yellow.

2. For coordinated control the implied decision at each intersection should be in line with historical decisions at its adjacent intersections, since it takes time for departing cars until they arrive at a downstream intersection.

Instead of improving the network cycle we propose a decentralized control of the traffic lights and try to achieve some synchronization by using arrival information.

8.2.2 Decentralized control

Under decentralized control the network is decomposed into its intersections at which the lights are controlled by FC with locally optimal cycle lengths. A locally optimal FC can be approximated by the Optimal-fixed-cycle algorithm in Appendix D by falsely modeling arrivals as Bernoulli experiments, instead of acknowledging that cars may arrive in platoons that are formed at upstream intersections. The resulting cycle lengths are not necessarily the same for all intersections. Although decentralized FC likely performs worse than coordinated FC, it may be a good initial policy for a one-step policy improvement. A motivation to take decentralized FC as the initial policy is that locally optimal FCs can be derived much more easily than a coordinated network FC.

The RV policies of the previous chapters are thus interesting to apply to a network of intersections. Once again, instead of considering the state of the entire network only local information for an intersection is used to adjust the traffic lights. There is no need to specify the offsets, since these are lost under dynamic control by the RV policies. By taking a number of slots of arrival information into account, one may obtain some degree of synchronization. By the nature of dynamic control, one should not expect to observe green waves under the RV policies. But, what we do hope is that the long-run average waiting time per car is much lower than under any of the other strategies, even lower than under coordinated FC.

8.2.3 Some modeling assumptions

In all cases that we will study, we presume that all cars flow through the network: all cars arrive from outside the network and do not have their destination in the network. In addition, we assume that the distance between the intersections is large enough to assume that under the tested policies the intersections will not be blocked by cars waiting at downstream queues.

Further, we assume that at each intersection the traffic flows are grouped such that non-conflicting opposite flows are served together. In the simulation model we do not allow cars to overtake slower traffic that proceeds along the same lane. (These assumption is used when specifying how opposite green waves are set.) We do not consider the optimization of the grouping of flows nor do we optimize the sequence in which the combinations are served. Principle assumptions regarding the test cases and the simulation model are discussed in Section 8.4.

8.2.4 Outline

As FC is an important benchmark to check the results of the new control policies, we discuss ideal and less-ideal circumstances to FC and how a good coordinated FC is set for specific cases in Section 8.3.

By simulating several network scenarios, the performance of the RV policies is compared against that of FC and exhaustive control. In Section 8.4, the test cases as well as a simulation model are presented.

In Section 8.5, we check for the performance of the rules under conditions that vary from utmost-ideal conditions for coordinated FC, to, what we consider, more realistic conditions. Therefore we illustrate the optimization of coordinated FC and demonstrate the impact on the average waiting time of the distance between intersections, the traveling speed of cars and the turning behavior of car drivers.

Although minimizing the overall average waiting time per car is our main objective, we study in Section 8.6 the progression of cars along a simple arterial. Next, in Section 8.7, we consider more complex arterials with more intersections and other more-complicated intersections. Finally, we consider, in Section 8.8, a network of nine intersection on a grid of three-by-three intersections.

8.3 Optimization of coordinated FC

To gain some insight, we explain the optimization of coordinated FC by means of an simple example under ideal conditions. Next less-ideal circumstances are discussed. Numerical results to compare different cycles of FC are reported in Section 8.5.2.

8.3.1 The I2F4C2 infrastructure: some notations, terminology and complications

Before we discuss how to set a good network cycle, we introduce an arterial with two simple intersections of the F4C2 type, as displayed in Figure 8.1. This simple infrastructure appears to be very insightful for optimizing the network cycle. We refer to this case as the I2F4C2 infrastructure, where I2 indicates the number of intersections along the arterial. The $I = 2$ intersections are numbered from left to right, or say from West to East: $i \in \{1, \ldots, I\}$. The distance between the intersection may be expressed in seconds or slots: $T_{1,i}$ is the time it takes to travel from intersection 1 to intersection i, when driving at constant speed equal to the speed limit and all lights are green. In all cases that we study $T_{1,i}$ is set to the distance in meters divided by a speed limit of 50km/h, which is $13\frac{8}{9}$ms^{-1} or $27\frac{7}{9}$m. per slot.

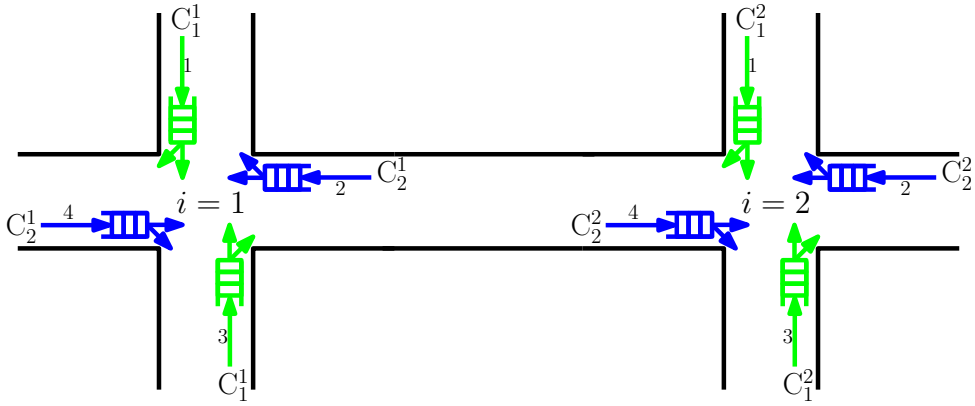


Figure 8.1: Arterial with 2 simple intersections: I2F4C2.

The West-to-East and East-to-West approaches are called the arterial. The traffic flows between the two intersection are the *internal flows*. The flows entering the network from the outside are called the *external flows*. Again, flows are numbered clockwise for each intersection. Combination s of intersection i is in short notation C_s^i; queue f at intersection i is referred to by Q_f^i.

Cars that leave Q_4^1 and Q_2^2 travel in two groups, called *platoons*, from one intersection to the other. When we assume that cars do not turn left or right, all cars in the platoons that travel along the arterial originate from one of these queues. When, in addition all cars travel at identical speed, the platoons remain intact. Under these assumptions a green wave under FC is easily accomplished.

For coordinated FC a network cycle applies. A cycle at intersection i starts ψ_i slots after a cycle starts at intersection 1. To find the best coordinated FC, one should integrally optimize over the effective green times $d_{i,s}$, the resulting cycle lengths D_i and the start ψ_i of a cycle for both intersections. Note that when the cycle length differs between the intersections, the lights cannot be unsynchronized and thus a (static) green wave cannot be set. Therefore the search is limited to FCs with identical cycle lengths $D_i = D$. Moreover, we assume that the infrastructures of the intersections are identical and that the arrival rate of a flow is the same at every intersection. Hence, the durations of the effective green times $d_{i,s}$ for every combination s is the same at all intersections: $d_{i,s} = d_s$.

In the next two subsections we focus on setting green waves for the I2F4C2 infrastructure, since green waves may reduce the long-run average waiting time per car.

8.3.2 A green wave in one direction

By setting the offset equal to the traveling time between the two intersections, one sets a green wave in one direction, from West to East, but not necessarily in the opposite direction. Furthermore, when cars do not travel at a constant speed equal to the speed limit, some cars may not experience a green wave. Slow moving cars may cause a big delay to cars directly behind them: when the lights at the downstream intersection have just turned into red upon the arrival of a car, it has to wait for the next green period.

Even when cars travel at a constant speed equal to the speed limit, not all cars may experience a green wave when some other cars are still queued upon the arrival at the downstream traffic light. These queued cars may originate from another queue: for example, cars that have turned right at Q_3^1 may still be present at Q_4^2 upon the arrival of cars originating from Q_4^1. To clear the queues, the lights should be switch to green before a new platoon of cars arrive. Consequently, the tail of the platoon may not experience a green wave, when the green periods of the downstream and upstream intersection have identical length.

8.3.3 A green wave in two directions

When the average number of cars traveling from East to West differs not much from the opposite direction one may prefer a green wave in both directions. This might also be desired in the light of minimizing the long-run average waiting time per car. Green waves can be set in both directions for any cycle length D, when one could control the speed at which cars travel. In our study, we do not control car speeds.

To ease the discussion at this point, we focus on what we call *to-FC-ideal cases* in which we assume that:

- all cars drive precisely at the speed limit, and
- no cars turn left or right.

In the next section, we discuss the impact of less-ideal conditions.

Network cycle length

For coordinated control the signals at the intersections should be phased and thus the cycle lengths should be identical. Since we study the problem under the *to-FC-ideal conditions*, the length of the green periods to C_2^1 and C_2^2 are equal. One condition to obtain a green wave is that a multiple of the cycle length D equals two times the traveling time $T_{1,2}$, according to Equation (8.1):

$$n \cdot D = 2 \cdot T_{1,2}, \qquad n \in \mathbb{N}. \tag{8.1}$$

The traveling time $T_{1,2}$ and the cycle length D are measured either in slots or in seconds. The equation is explained through the following simple example.

Example – Consider an arterial with two signalized intersection of the F4C2 type. All traffic is thru traffic and cars drive at constant speed such that the distance between the intersections is $T_{1,2} = 22$ slots. Suppose at time 0 a cycle of FC starts by changing the lights of C_2^1 into green. Then cars leaving Q_4^1 will arrive at Q_4^2 after $T_{1,2} = 22$ slots. When no cars are waiting at Q_4^2 and the lights of C_2^2 change into green at time 22, then all cars traveling from West to East experience a green wave as long as the green periods at the two intersections are of equal length.

In the opposite direction cars pull up from Q_2^2 at time $T_{1,2} = 22$ and travel to intersection 1 where they arrive at time $2 \cdot T_{1,2} = 44$. To impose a green wave in two directions, a new cycle should start (at least) every $2 \cdot T_{1,2} = 44$ slots. The cycle length should thus be $D = 2 \cdot T_{1,2} = 44$, $D = T_{1,2} = 22$ or $D = T_{1,2} \,/\, 2 = 11$ slots. (In this example $D = 2 \cdot T_{1,2} \,/\, 5$ is not considered as it does not result in an integer cycle length. Rounding D is possible but will imply that the average waiting time in the two opposite direction will differ. FC with $D = 2 \cdot T_{1,2} \,/\, 5$ and lower cycle lengths are not possible as a cycle last at least 8 slot: switching takes 3 slots and the minimum green time is 1 slot for each combination.)

The best choice of D depends on the load of the intersection, as set by the arrival rates.

Optimal offset

A second condition that needs to be satisfied to achieve a green wave in two directions concerns the *offset*: the time between the start of a cycle at intersection 1 and the start of a cycle at intersection 2. When cars travel at constant speed and do not turn left or right, an offset ψ_2 equal to the traveling time between the intersections is optimal given D is set according to Equation (8.1). Multiple values of the offsets may yield a green wave, we therefore define the offset according to Equation (8.2).

$$\psi_i = T_{1,i} \mod D. \tag{8.2}$$

When D and $T_{1,i}$ are in slots, then ψ_i is also in slots. (Note that by definition $\psi_1 = 0$.)

Space-time diagram

In Figure 8.2 we illustrate through a so-called *space-time diagram* that green waves are set in both directions. The horizontal bars show the signaling scheme with respect to the flows in Q_2^1 and Q_2^2: the horizontal flows. The green, yellow, and red periods are marked by different colors. At the two plots at the top, the cycle length equals $D = 22$ respectively $D = 11$ slots and the offset equals $\psi_2 = 0$. When the cycle length equals $D = 44$ slots, as in the last plot, the offset is $\psi_2 = 22$ slots.

In the first plot at time 0 a green period of 8 slots starts at both intersections, followed by 2 yellow slots: the effective green time is 10 slots. The arrows indicate the progression of the traffic between the intersections: to simplify the plots we have assumed cars to travel at a constant speed as if they do not require time to accelerate when leaving a queue or to slow down when approaching the end of a queue.

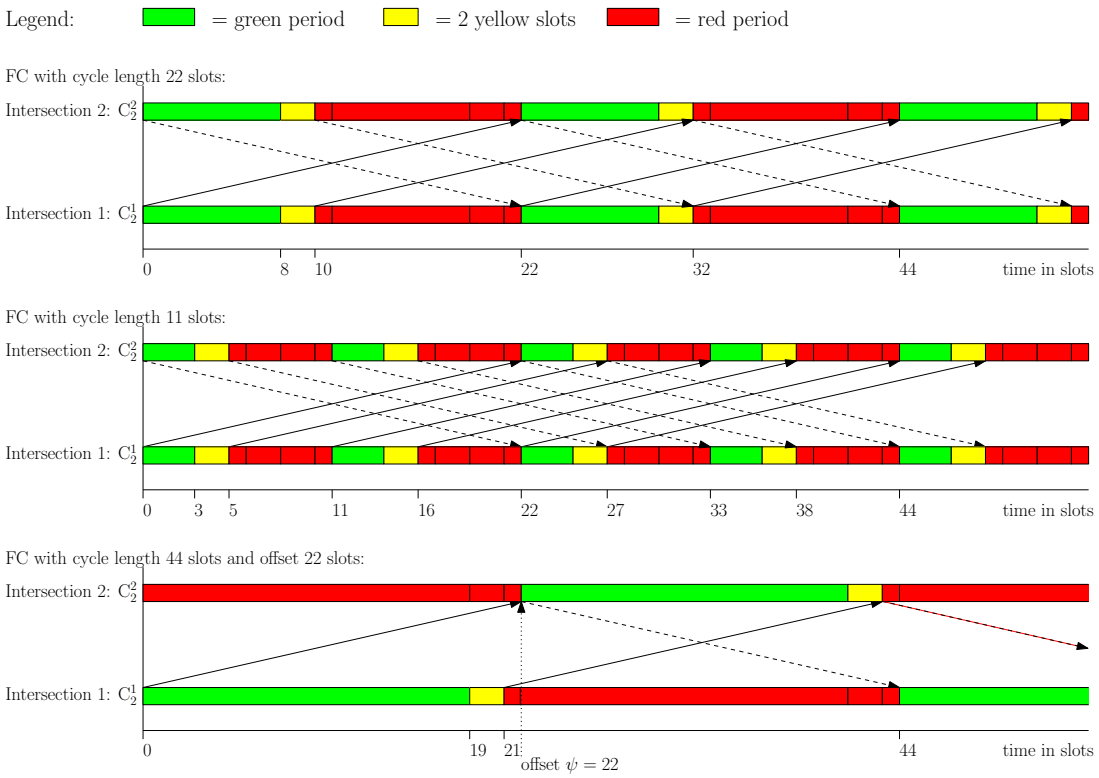


Figure 8.2: Synchronizing FC for a to-FC-ideal I2F4C2 case: green waves can be set in both direction when speeds are constant and identical; $\rho_1 = \rho_2 = 0.8$, $T_{1,2} = 22$ slots.

The plot shows that cars leaving Q_4^1 and Q_2^2 during the first $d_{i,2} = 8 + 2 = 10$ slots experience a green wave at the respective downstream intersection at slots 22 – 32. In this case the green wave holds in two directions. Similarly green waves are plotted for cycle length $D = 11$ respectively $D = 44$ slots. Note that two opposite green waves at cycle length $D = 44$ slots require an offset of $\psi_2 = 22$ slots: hence at time 0 any cars present depart from Q_4^1, but no car leaves Q_2^2.

8.3.4 To-FC-ideal cases and less-ideal cases

The formulas in Equations (8.1) and (8.2) hold under the to-FC-ideal conditions, which we have formulated in Section 8.3.3. That is, cars travel at identical speeds equal to the speed limit and cars do not turn left or right. When in addition to these conditions the locally optimal cycle length found by the Optimal-fixed-cycle algorithm satisfies Equation (8.1), we deal with a what-we-call a *to-FC-utmost-ideal case*. In other words, in addition to the to-FC-ideal conditions, the traveling distance is thus ideal to set green waves while setting the locally optimal cycle lengths.

In reality the conditions may be not ideal:

1. cars may travel at different speeds,
2. cars are allowed to turn, and
3. the distance between intersections is fixed, while the locally optimal cycle length depends on the workload.

Then green waves may hold for only a fraction of the cars, but are easily lost, as we discuss below. In Section 8.5, we numerically show the impact when one or more of the conditions do not hold.

Green waves are distorted when speeds vary

When traveling speeds of cars differ, not all cars will experience a green wave. On the one hand fast moving cars, if not hampered by slow-moving traffic, may arrive at the downstream intersection before the green period starts. On the other hand, slow-moving cars may arrive (just) too late at the next intersection and have to wait for the next green period.

In the space-time diagram in Figure 8.3, the first two arrows show the progression of a fast moving car, respectively a slow moving car, that travel from intersection 1 to intersection 2. The third arrow has a bend and shows that a fast moving car gets delayed by a slow moving car. As we assume that cars are not allowed to overtake slow moving cars, platoons are formed. Consequently, the tail of a platoon may stay behind at the downstream intersection when the lights are red upon its arrival. These queued cars delay new arrivals.

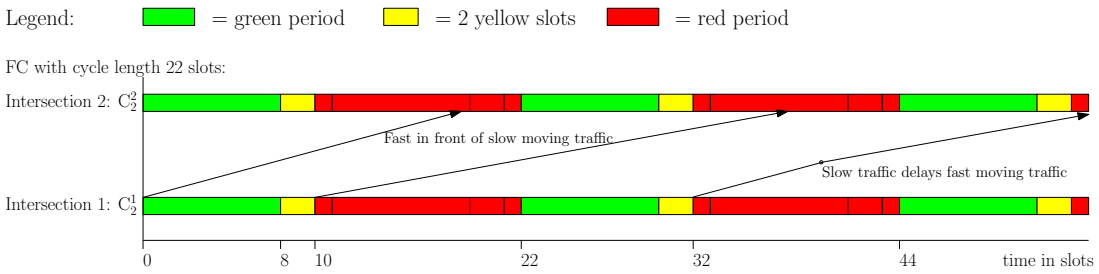


Figure 8.3: Illustration that green waves under coordinated FC are lost when car speeds vary: the I2F4C2 case with $\rho_1 = \rho_2 = 0.8$ and ideal distance $T_{1,2} = 22$ slots (=611 meters).

Dealing with right turning traffic

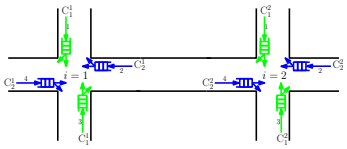
A green wave may be lost also when cars are (still) queued at the downstream intersection when a platoon of cars arrive at an internal queue. The queued cars may be the tail of a previous platoon or they may originate from another flow: e.g. cars that turn right when leaving Q_3^1 will accumulate at Q_4^2 and will thus delay thru traffic originating from Q_4^1.

When cars are allowed to turn right the green waves that are set under the to-FC-ideal conditions are easily lost. To reduce the waiting time one may adjust the cycle length and offset such that the internal queues are depleted before a new platoon arrives. Depending on the number of queued cars the head of a platoon may experience a green wave, but, depending on the length of the platoon, the tail of the platoon face a red light.

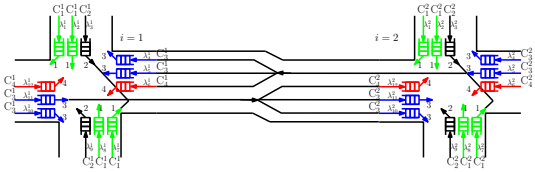
The length of a platoon is influenced by the duration of the green period at the upstream intersection. When the green time of the internal traffic flows is longer than that of the external flows, one may avoid that the tail has to stop while the head of a platoon experiences a green wave. Since in the infrastructures that we study, we have grouped opposite thru traffic in the same combination, we will not consider different green periods to internal and external flows. In general, setting a green wave in two directions is hard to settle when traffic leading to a queue originates from different combinations.

8.4 Simulation model for networks

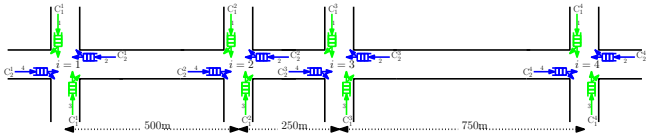
The RV policies developed in the previous chapters are applied to several network cases. Figure 8.4 shows a set of four infrastructures that are simulated in the next sections. The simulation model described in Section 7.3 is therefore extended, such that the outflow of one intersection is the inflow to a downstream intersection. The progression of cars along the lanes is tracked by monitoring the actual position of each car. In practice the positions are either known by cameras or induction loops or estimated based on the time elapsed since a car has left the upstream intersection ([93], [65]).



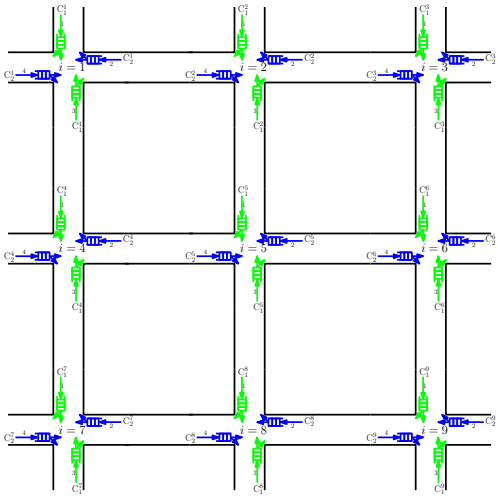
(a) I2F4C2: a simple arterial,
see Sections 8.5 and 8.6.



(b) I2F12C4: an arterial with multiple lanes,
see Section 8.7.



(c) I4F4C2: four unequally-spaced intersections,
see Section 8.7



(d) I3x3F4C2: a network of nine intersections on a grid,
see Section 8.8.

Figure 8.4: Outlook on test infrastructures and related sections.

We summarize the most relevant features and assumptions in the simulation model:

1. The simulation happens in discrete time with fixed time increments by 1 slot of 2 seconds.

2. Cars have their origin and destination outside the network.

3. External arrivals within a slot happen 500 meters upstream and are generated by a Bernoulli experiment for each flow f (with the probability of an arrival in a slot equal to λ_f^i).

4. The (desired) speed of a car is either 50 km/h for all cars, or is drawn from a Triangular distribution, Tri(40,50,60), with minimum, mean, and maximum car speed equal to respectively 40, 50 and 60km/h.

5. A car adjusts its speed to slow moving cars directly in front of it, since they are not allowed to overtake other cars. Cars keep a safe traveling distance of 2 seconds.

6. The time at which a car arrives at the Tail of the Queue (ToQ), is estimated based on the exact position of all cars, their traveling speed, and the position of the ToQ (=queue length).

7. As in most microscopic simulation programs, cars are queued horizontally: each queued car takes 7 meters.

8. When the lights show green or yellow the queued cars travel 7 meters per slot (=12.6km/h), since each slot 1 car leaves the queue. Cars that leave a queue accelerate in zero time to their desired speed (or less when hampered by slow moving cars). Similarly cars decelerate in zero time when joining a queue, or when running into slow moving traffic.

9. Cars are routed randomly based on fixed probabilities for turning left or right and lane selection, since they do not have specific destinations. Car drivers change lanes only when crossing an intersection. The routing model is explained in more detail in Sections 8.5.4 and 8.7.2.

The simulation model that we have developed, contains thus a number of realistic features that are not included in the decision model. Principal objective is to minimize the long-run average waiting time per car, which is defined as the integer number of slots a car spends in the queue. Despite the discrepancies between the simulation model and the Markov decision model, we expect the RV policies to perform well as will be shown in the next sections.

Outline of the simulation study

In Table 8.1 we show how the remainder of this chapter is organized. We consider five studies, I to V, based on four infrastructures. A number of scenarios will be related to each study; these are explained and motivated in the respective sections.

Table 8.1: Organization of the simulation study on controlling networks of intersections.

Study	Infrastructure	Motivation	Section
I	I2F4C2	Impact of distance, traveling speed and right turning traffic	8.5
II	I2F4C2	Progression on arterial	8.6
III	I4F4C2	Multiple intersection at unequal distances	8.7.1
IV	I2F12C4	More complex intersections	8.7.2
V	I3x3F4C2	Network of 9 intersections on a grid of 3-by-3	8.8

Regarding the test cases, we limit the number of test cases FC by focussing on the performance under a high workload: $\rho_i = 0.8$. In the previous chapters we have learned that the difference between the RV policies and the (uncoordinated) FC is bigger at lower workloads. In each of the five studies we are interested in the performance of the RV policies, compared to the 'best' FC. The choice of a best coordinated FC as a benchmark, is supported by reporting several FCs with different cycle lengths and offsets. To ease the analysis, most studies concern networks of F4C2 intersections with identical arrival rates. In Sections 8.6 and 8.7.2 we consider a few asymmetric cases. For the infrastructures with F4C2 intersections, we focus on cyclic RV1 policies. Acyclic control and RV2 policies are studied in Section 8.7.2 for the I2F12C4 case, which consists of more combinations per intersection.

Simulation horizon – All reported figures are averages over 100 replications of 72,000 slots, excluding a warming up period of 450 slots (=15 minutes) starting with an empty system. The figures are accurate up to at least 2 or 3 digits, hence we omit confidence intervals. Moreover, the relative performance of the different policies is the objective of the study rather than the absolute figures.

8.5 Study I – I2F4C2 arterial

We put the RV1 policies to the test by simulation of traffic along the simple arterial depicted in Figure 8.4(a): I2F4C2. This simple infrastructure illustrates that green waves can be set under ideal circumstances, but as we will see these are easily lost. The purpose of the simulation study in this section, is to investigate the sensitivity of FC and to check whether the RV policies perform better than coordinated FC.

All policies are simulated, under varying assumptions concerning the car speed, distance between the intersections and the fraction of cars that turn right. Table 8.2 shows an overview of the four scenarios, numbered I-A to I-D, that are studied in the next four sections. In all simulations, we assume identical arrival rates equal to 0.4, and thus the workload of each intersection is 0.8.

Table 8.2: Overview of 4 scenarios for fully-symmetric I2F4C2 ($\rho_1 = \rho_2 = 0.8$; $\lambda_f^i = 0.4$).

Scenario	Short description	Travel distance in meters	Travel distance in seconds	Distribution car speeds (km/h)	% cars that turn right	Section
I-A	To-FC-utmost-ideal	611m	$T_{1,2} = 44$s.	Det(50)	0%	8.5.1
I-B	To-FC-ideal case	500m	$T_{1,2} = 36$s.	Det(50)	0%	8.5.2
I-C	Speed differences	500m	$T_{1,2} = 36$s.	Tri(40,50,60)	0%	8.5.3
I-D	Speed differences and right turning cars	500m	$T_{1,2} = 36$s.	Tri(40,50,60)	25%	8.5.4

A summary of the results and a list of major conclusions is provided in Section 8.5.5.

8.5.1 Scenario I-A – An utmost-ideal case for FC (I2F4C2)

In the to-FC-ideal case, cars travel at identical speed (50 km/h) and do not turn left or right. In an utmost-ideal case, the traveling time $T_{1,2}$ between the intersections equals the locally optimal cycle length D. Then one can impose green waves in both directions as discussed in Section 8.3 with a network cycle identical to the locally optimal cycle.

The locally optimal cycle length of a fully-symmetric F4C2 intersection at a load of 80% is 22 slots (44 seconds), as we have seen in Table 6.2. An utmost ideal distance between the intersections is thus $T_{1,2} = 44$ seconds; or 611 meters as cars travel 50 km/h. When $T_{1,2}$ is 44 seconds, the best network FC is most likely the one with $D = 44$, $d_{i,s} = 20$ and $\psi_2 = 0$ seconds.

Table 8.3: Mean waiting times (in sec.) for I2F4C2 when $\rho = 0.8$ ($\lambda_f^i = 0.4$): a to-FC-utmost-ideal case with distance 611 meters ($T_{1,2} = 44$s.), all cars speeds are 50km/h and all traffic is thru traffic.

		% above	Internal flows	External flows	
Policy	EW	RV1(5)	EW_2^1, EW_4^2	EW_4^1, EW_2^2	EW_1^i, EW_3^i
RV1(5)[a]	12.7		10.9	13.7	13.1
RV1[a]	15.9	+25.4%	17.0	14.9	16.0
FC[a]	13.7	+8.1%	0	18.3	18.2
XC	24.9	+96.0%	25.2	23.5	25.4
XC-1	19.4	+52.8%	19.3	18.3	20.0
XC-2	16.1	+27.2%	15.4	15.7	16.8

[a]Based on FC with $D = 44$, $\psi_2 = 0$, and $d_{i,s} = 20$.

In Table 8.3 we report on the long-run average waiting times EW_f^i (in seconds) at all queues as obtained through simulation. The overall expected waiting time (EW) is reported in the second column. Clearly, RV1(5), the RV1 policy with 5 slots of arrival information, performs best.

Coordinated FC yields an average waiting time per car that is 8.1% above that of RV1(5), although in this case FC implies a green wave at the internal queues Q_2^1 and Q_4^2. Under dynamic control, e.g. under the XC and RV1 policies, green waves are lost. The RV1 policy with no arrival information yields a waiting time that is 25% above that for RV1(5).

Even for this simple symmetric infrastructure with an identical number of flows in each combination, the exhaustive control policies perform bad compared to FC and RV1(5): the overall average waiting time is 27 to 96% higher than that under RV1(5). This shows that RV1(5) is a relatively simple policy that is superior to both FC and XC policies.

Other ideal distances between the intersections

For the given workload $\rho = 0.8$, the locally optimal FC with cycle length 44 seconds was argued to be optimal FC for the I2F4C2 infrastructure when $T_{1,2} = 44$ seconds (or 611 meters). In fact, according to Equation (8.1) $D = 44$ is also optimal when the traveling time traveling time is any multiple of 22 seconds. Hence when traveling at at 50 km/h one experiences a green wave whenever the distance between the two intersections is 306m, 611m, 917m, 1222m, etc. For any other distance the length of a cycle must be adjusted to keep a green wave. The FC that minimizes the overall long-run average waiting time is then no longer trivial.

8.5.2 Scenario I-B – Non-ideal distance: 500 meters (I2F4C2)

When the distance between the two intersections is only 500 meters, then the traveling time equals $T_{1,2} = 36$ seconds when traveling 50 km/h. According to Equation (8.1), a green wave is possible only under cycle lengths 72, 36, 24, 18 seconds. Since the locally optimal cycle equals 44 seconds, a promising network cycle has cycle length D equal to 36, 44 or 72.

In Table 8.4, we present the long-run average waiting times under RV1(5), FC and XC-2. Note that, here and in the remainder, we omit the results for XC and XC-1, since these policies perform much worse than XC-2 as demonstrated already in Table 8.3.

The waiting times under FC are significantly higher, even if one sets green waves in both directions. The best FC has cycle length 36 seconds and shows green waves in both directions. But, the implied long-run average waiting time is now 11.6% above that under RV1(5). Still, coordinated FC performs better than the best XC-2.

Coordinated control with $D = 72$s. and offset $\psi_2 = 36$s., yields a much higher average waiting time than FC with $D = 36$ and $\psi_2 = 0$. Under non-coordinated FC, the cycle length is set to the locally optimal one (44 seconds) and the offset is set to zero, such that cars traveling from west to east experiences the same average waiting as cars traveling in the opposite direction. As expected, green waves are then lost: cars arriving at the intersection have to wait at least 8 seconds, since the head of a platoon arrives at the downstream intersection $(44 - 36 =)$ 8 seconds before a green period starts. Since we assume queued cars to take 7 meters of space per car, the average waiting time (=the average time to spend in each queue) is higher than 8 seconds.

Table 8.4: Mean waiting times (in sec.) for I2F4C2 at $\rho = 0.8$ ($\lambda_f^i = 0.4$): the to-FC-ideal case (cars speed = 50km/h, no cars turn, and distance = 500 m.).

Policy	EW	% above RV1(5)	Internal flows EW_2^1, EW_4^2	External flows EW_4^1, EW_2^2	External flows EW_1^i, EW_3^i
RV1(5)[a]	13.0		12.1	13.6	13.1
RV1[a]	15.3	+17.7%	15.0	15.3	15.5
FC($D = 36, \psi_2 = 0, d_{i,s} = 16$)	14.5	+11.6%	0	19.3	19.4
FC ($D = 44, \psi_2 = 0, d_{i,s} = 20$)	16.2	+24.5%	9.8	18.3	18.3
FC ($D = 72, \psi_2 = 36, d_{i,s} = 34$)	16.1	+23.9%	0	21.5	21.4
XC-2	16.2	+24.4%	15.4	15.7	16.8

[a]Based on FC with $D = 44$, $\psi_2 = 0$, and $d_{i,s} = 20$.

Conclusions on I2F4C2 under to-FC-ideal conditions, but non-ideal distance

Under to-FC-ideal conditions, cars do not turn left or right and do travel at constant speed equal to the speed limit (50km/h).

- In the case under consideration with $\rho = 0.8$ the best FC is coordinated FC with $D = T_{1,2} = 36$ and $\psi_2 = 0$ seconds.
- Compared to the case with ideal distance between the intersections, of the previous subsection, the waiting time under FC has increased by about 6 percent, from 13.7 to 14.5 seconds.
- The waiting time under the RV1 policies are more robust to changes in the distance between the intersections.
- RV1(5) performs better than the best coordinated FC, which in turn is even better than the best exhaustive control policy (XC-2).
- Apparently, the RV policies can easily cope with batch arrivals, although in the computation of the relative values, arrivals occur uniformly over time.

8.5.3 Scenario I-C – Different speeds (I2F4C2)

When cars travel at different speeds, the platoon originating from an upstream intersection falls apart into smaller platoons. Sub-platoons are formed by slow moving traffic that delays fast moving traffic. The time between the first car and the last car arriving at the downstream intersection may exceed the length of the effective green time under coordinated FC. Consequently, some cars will not experience a green wave. Under FC one has to make a trade-off between giving all cars in the (first) platoon a small delay or giving a larger delay to a few cars at the tail of a platoon.

The individual preferred car speeds are drawn from a triangular distribution with mean 50 km/h and minimum and maximum, 40 respectively 60 km/h. Traveling 500 meters to an intersection, will take a car 30 to 45 seconds. Since cars are not allowed to overtake, the average traveling speed will be below 50 km/h.

As the average speed is lower, the best cycle length under coordinated FC is most likely higher than 36 seconds, which was optimal in the previous section. Finding the best cycle length and offset is now considerably more complicated. By a partly enumerate search we investigate a number of configurations of FC.

Search for the best coordinated FC

By simulation of a number of combinations (D, ψ_2), we search for a good cycle length and offset. The results are reported in Figure 8.5. Figure 8.5(a) considers the case where the duration of the green periods for the two combinations are identical. Starting with a short cycle of 12 slots (24 seconds), under which the queues are just stable, both green periods are lengthened by one slot resulting in cycle length 28 seconds. The cycle length is increased by increments of 4 seconds until $D = 92$ seconds. We limit the choices for the offset to $\psi_2 = 0$ and $\psi_2 = D/2$, such that the waiting times in both (horizontal) directions are identical. The best coordinated FC seems to have $D = 44$s. and $\psi_2 = 0$.

In Figure 8.5(b) we have refined the search, by considering cycle lengths that imply green periods that differ in length by one slot. Since the arrival processes at the internal and external queues are not the same, unequal green periods could give a slight improvement even when the arrival rates are identical. In the case under consideration, an improvement stays out: all the green periods are of identical length: $d_{i,s} = 20$ seconds.

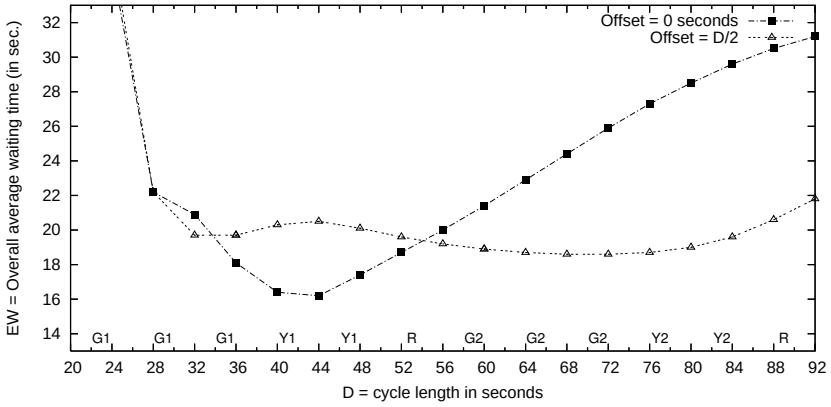
Finally, we consider alternative offsets in Figure 8.5(c), with D fixed to 44 seconds. Since speeds are identical and all traffic is thru traffic, offset 0 is indeed optimal. (The worst choice of ψ_2 is 22 seconds, since $\psi_2 = 22$ implies all cars in the horizontal directions to wait for at least $D - \psi_2 = 18$ seconds.)

The 'best' network FC seems to be the locally optimal cycle: $D = 44$ and $\psi_2 = 0$ seconds.

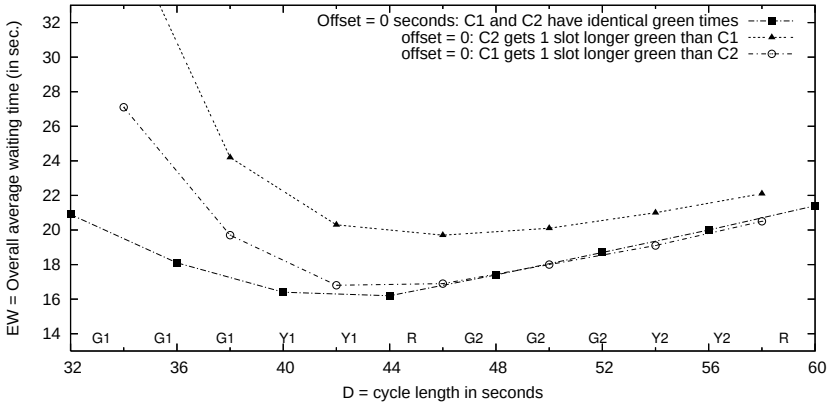
Comparison of RV1 policies and coordinated FC

In Table 8.5 we report the overall long-run average waiting time when speeds do vary. The average waiting time under all policies are higher than when travel speeds were constant and identical, because of the increased uncertainty in car arrivals.

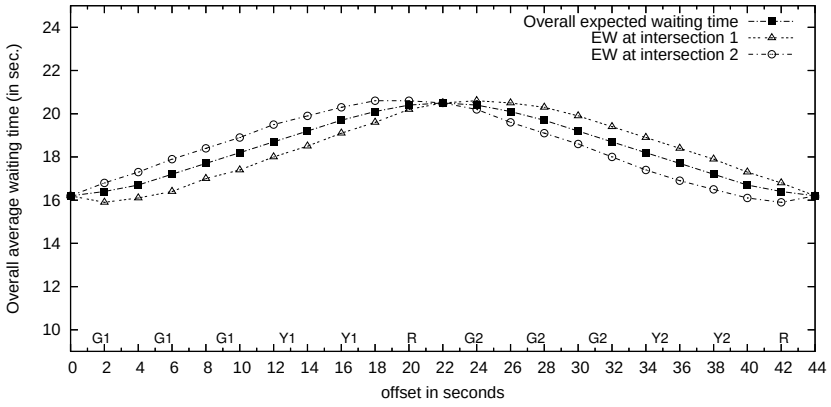
The difference between FC and RV1(5) is larger than when speeds were identical. This is partly due to the fact that the average waiting time under FC is no longer zero at the internal queues: green waves are lost due to the speed differences. The average waiting time at the internal queues is lowest when the cycle length is set higher than the average traveling time between the intersection. A longer cycle time (with offset zero), causes cars to accumulate at the downstream queue and thus 'repairs' the dispersion of the platoons. As a result the green periods at the downstream intersection are used more effectively.



(a) Optimization of FC over D (a multiple of 4 seconds) with $\psi_2 \in \{0, \frac{D}{2}\}$.



(b) Optimization of FC over D (a multiple of 2 seconds) with $\psi_2 = 0$.



(c) Optimization of FC over ψ_2 for $D = 44$.

Figure 8.5: Optimization of FC phased in two directions for I2F4C2 case with different car speeds and all thru traffic.

Table 8.5: Mean waiting times (in sec.) for I2F4C2 at $\rho = 0.8$ ($\lambda_f^i = 0.4$): all thru traffic and car speeds $\sim$ Tri(40,50,60).

Policy	EW	% above RV1(5)	Internal flows EW_2^1, EW_4^2	External flows EW_4^1, EW_2^2	EW_1^i, EW_3^i
RV1(5)[a]	14.2		12.6	15.3	14.3
RV1[a]	16.3	+15.4%	16.0	16.4	16.4
FC ($D = 36, \psi_2 = 0, d_{i,s} = 16$)	18.1	+28.0%	10.2	20.8	20.8
FC ($D = 44, \psi_2 = 0, d_{i,s} = 20$)	16.2	+14.5%	6.1	19.5	19.5
FC ($D = 72, \psi_2 = 36, d_{i,s} = 34$)	18.6	+31.4%	6.5	22.6	22.6
XC-2	17.2	+21.8%	16.9	16.7	17.7

[a]Based on FC with $D = 44$, $\psi_2 = 0$, and $d_{i,s} = 20$.

Conclusions on I2F4C2 when cars do not turn and travel at different speeds

- When speeds vary, the best FC has a cycle length that is higher than the average travel time between the intersections.
- The best coordinated FC performs almost equally well as RV1, which does not use arrival information.
- Adding 5 slots of arrival information to RV1, makes a big difference, even when the information is not accurate due to speed difference.
- Although green waves are lost when speeds vary, the average waiting times under RV1(5) at the internal queues are lower than at the external queues.
- RV1(5) results in the lowest overall average waiting time: the best FC is still 14.5% off.

8.5.4 Scenario I-D – Right turning traffic (I2F4C2)

So far, we did not allow cars to turn right; this to ease the discussion on how to set green waves. In this section we investigate the impact of allowing cars to turn right. We introduce right turning traffic not only to make the problem more realistic but also to demonstrate the complexity of optimizing FC, as discussed already in Section 8.3. Left turns are prohibited, since this would be in conflict with the (opposite) thru traffic of the same combination.

Consider the I2F4C2 infrastructure. Upon passing the stopping line, 25% of the cars decide to turn right, rather than to go 'thru'.[1]

We simulate right turning traffic by means of a probability distribution: with probability $P_f^i(\mathrm{R}) = 0.25$ a car of flow f turns right at intersection i, and with probability $P_f^i(\mathrm{T}) = 0.75$ it goes thru, upon passing the stopping line. In the next sections we consider other probabilities and a case with separate lanes for left turning, right turning, and thru traffic.

Again, we evaluate several configurations of FC to find out which combination of D and ψ_2 is best. For setting good RV1 policies, any good FC satisfies. Therefore, we choose the cycle length (D_i) of intersection i equal to the locally optimal one, since the cycle lengths do not need to be identical for the RV1 policies. One may set the offsets to 0, since under dynamic control the offsets are irrelevant.

Search for the best coordinated FC

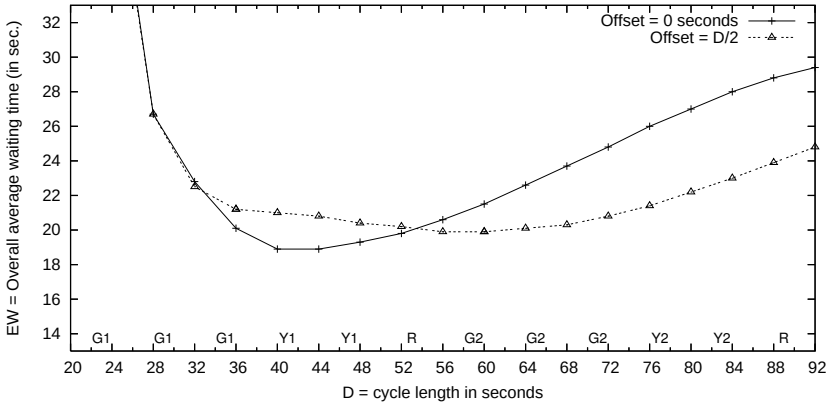
Figure 8.6 shows the search for an optimal FC when cars speeds are generated randomly from a triangular distribution, as in the previous section. The best cycle length that we have found is 40 or 44 seconds with offset 0. In comparison to Figure 8.5, the curves are more flat when cars are allowed to turn right. The upper two curves in Figure 8.6(b) almost coincide. Thus the arrival processes at the internal queues differ little from the arrival process at the external queues. Groups of cars leaving a queue do not form a platoon, since some cars leave the platoon by turning right at the stopping line. In addition, the arrival process is more uniformly distributed, or say more smooth, as the cars arriving at the internal queues originate from more queues, e.g. cars arriving at Q_4^2 originate from either Q_3^1 or Q_4^1.

Consequently, the minimal average waiting time under FC is somewhat higher and choosing a suboptimal value for the offset has less impact on the performance of FC.

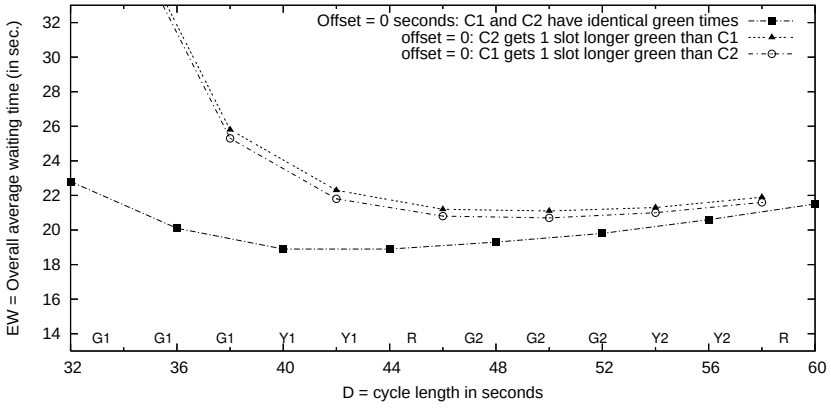
Comparison of RV1 policies and coordinated FC

In Table 8.6, we report the long-run average waiting times. Since the arrival of cars at the internal queues are more evenly spread over time, the average waiting time per car differs not much between the queues. Although the arrival process thus has a lower coefficient of variation, the average waiting times per car under FC and the RV1 policies are higher

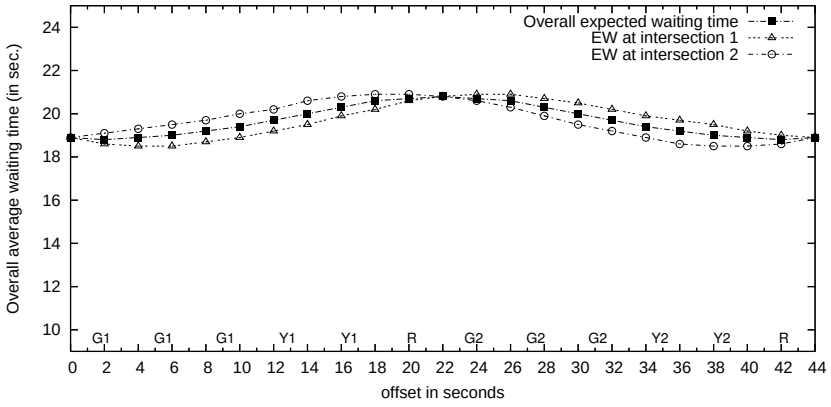
[1]The typical F4C2 intersection, at which left turns are forbidden, can be found in a number of American and Canadian cities, where (during rush hours) cars have to make three right turns around a block instead of making a left turn.



(a) Optimization of FC over D (a multiple of 4 seconds) with $\psi_2 \in \{0, \frac{D}{2}\}$.



(b) Optimization of FC over D (a multiple of 2 seconds) with $\psi_2 = 0$.



(c) Optimization of FC over ψ_2 for $D = 44$.

Figure 8.6: Optimization of FC phased in two directions for I2F4C2 case with different car speeds and 25% of cars turns right.

Table 8.6: Mean waiting times (in sec.) for I2F4C2 at $\rho = 0.8$ ($\lambda_f^i = 0.4$): 25% of traffic turns right and cars travel at different speeds.

Policy	EW	% above RV1(5)	Internal flows EW_2^1, EW_4^2	External flows EW_4^1, EW_2^2	EW_1^i, EW_3^i
RV1(5)[a]	14.7		14.4	14.8	14.8
RV1[a]	16.5	+12.2%	16.7	16.3	16.6
FC[a]	18.8	+27.8%	16.7	19.6	19.5
XC	25.4	+72.6%	25.4	24.9	25.7
XC-1	20.0	+36.0%	20.1	19.6	20.2
XC-2	16.7	+13.1%	16.6	16.4	16.8

[a]Based on FC with $D = 44$, $\psi_2 = 0$, and $d_{i,s} = 20$.

than those reported in Table 8.5, since it becomes harder to anticipate future arrivals. XC-2 yields higher waiting times when the arrival process is more evenly spread over time: XC-2 performs best when successive platoons are far apart.

Again we conclude that RV1(5) is superior to FC and XC-2. XC-2 is close to RV1, but yields an overall average waiting time that is 13% above that of RV1(5). The best FC is about 28% worse than RV1(5).

Remark – Although not reported we have seen similar results when speeds would be identical. The difference between RV1(5) and FC is then even bigger, since RV1(5) is more sensitive to speed differences than FC.

Table 8.7: Summary of results of all four scenarios for symmetric I2F4C2 with load 0.8.

Scenario	I-A	I-B	I-C	I-D
Section	8.5.1	8.5.2	8.5.3	8.5.4
Travel distance (in meters)	611	**500**	**500**	**500**
Distribution car speeds (km/h)	Det(50)	Det(50)	**Tri(40,50,60)**	**Tri(40,50,60)**
% cars that turn right	0%	0%	0%	**25%**
Best network FC: $(D, \psi_2, d_{i,s})$	(44, 0, 20)	(36, 0, 16)	(44, 0, 20)	(44, 0, 20)
Policy	Average waiting time per car			
RV1(5)	12.7	13.0	14.2	14.7
RV1	15.9	15.3	16.3	16.5
FC	13.7	14.5	16.2	18.8
XC-2	16.1	16.2	17.2	16.7

8.5.5 Conclusions for Scenario I-A to Scenario I-D

We summarize in Table 8.7 the main results for all four scenarios I-A to I-D.

Although the results relate to the symmetric I2F4C2 arterial at workload 0.8, we stipulate some conclusions that may hold in general:

1. RV1(5) is favored above the best coordinated FC, because the average waiting time per car is much lower under RV1(5) and the optimization of FC can be difficult:
   - Finding a locally-good FC that act as initial policy to the RV policies is much easier than finding a well-coordinated FC.
   - Imposing a green wave in two directions requires identical cycle lengths at all intersections and offsets to be equal to 0 or half the cycle length.
   - Green waves in two opposite directions exist only under ideal conditions, and are easily lost when car speeds vary or part of the traffic turns right.
2. Speed differences and right turning traffic smooth the arrival process at internal queues, and cause an increase of the average waiting time at the internal queues under both coordinated FC and RV1(5):
   - Platoons get dispersed, when cars travel at different speeds and when cars turn right at the upstream intersection.
   - When platoons fall apart into two (or more) sub-platoons, the best coordinated FC has a cycle length larger than average traveling time: the head of a platoon gets delayed to reduce a potentially excessive waiting time of cars at the very tail of the platoon.
   - RV1(5) is more sensitive to speed differences, since it uses the estimated arrival times of cars.
3. RV1 with a few slots arrival information yields a much lower average waiting time per car than the best FC and XC-2.

In the next section we investigate how we may steer the RV policies to satisfy secondary criteria such as reducing the waiting time at the internal queues. Further we investigate a skew case where most of traffic travels from west to east and one may set a green wave in 1 direction. In Sections 8.7 and 8.8 we consider more general infrastructures.

8.6 Study II – Progression along arterial

Although the principal criterion in this study is to minimize the long-run average waiting time per car, the progression along the arterial may be important. Therefore we investigate whether the RV1 policies allow to decrease the average waiting times at the internal queues. Three scenarios are considered as tabulated in Table 8.8.

Table 8.8: Scenarios with respect to the progression at I2F4C2 arterial.

Scenario	Description	Section
II-A	Internal queues have higher weight	8.6.1
II-B	Internal queues have more slots of arrival information	8.6.2
II-C	High arrival rates in the West-to-East direction	8.6.3

In all three scenarios, 25% of the cars turns right and individual (preferred) car speeds are drawn from a triangular distribution with mean 50 km/h and minimum and maximum 40 and 60 km/h respectively. In scenarios II-A and II-B, the arrival rates are identical ($\lambda_f^i = 0.4$), and we mainly investigate options to reduce the waiting time at the internal queues.

Scenario II-C is interesting for two reasons. Firstly, the asymmetry in the arrival rates may make a green wave in only one direction favorable, even though not all cars traveling in that direction will benefit. Secondly, non-coordinated FC, with locally optimal cycle lengths, may perform quite well, since the loads of the two intersections differ.

8.6.1 Scenario II-A – Higher waiting costs at internal queues (I2F4C2)

The RV policies are based on relative (cost) values of states under FC. In the evaluation of the Markov chain under FC, we incur 1 unit cost for each car that is queued at the start of a slot, independent of the queue at which the car is waiting. By accounting higher costs for cars waiting at the internal queues, one may reduce the waiting time at these queues. These costs per car per slot serve as weights of the flows.

In Table 8.9 we report on the waiting times, when the waiting costs at the internal queues, Q_2^1 and Q_4^2, are tripled. For easy comparison, we copy at the second line the results for RV1(5) of the previous section, in which all flows had identical weights. The exhaustive

Table 8.9: Mean waiting times (in sec.) for I2F4C2 at $\rho = 0.8$ ($\lambda_f^i = 0.4$): waiting costs at internal queues is tripled.

Policy	EW	% above RV1(5)	Internal flows EW_2^1, EW_4^2	External flows EW_4^1, EW_2^2	External flows EW_1^i, EW_3^i
RV1(5)[a] – identical costs	14.7		14.4	14.8	14.8
RV1(5)[a]	15.4	+4.6%	9.3	15.9	18.2
RV1[a]	17.6	+19.3%	13.1	15.5	20.9
FC[a]	18.8	+27.8%	16.7	19.6	19.5

[a]Based on FC with $D = 44$, $\psi_2 = 0$, and $d_{i,s} = 20$.

control policies are not included, as the performance of these policies is not affected by changes in the cost structure.

In this case, the locally optimal cycle is also optimal for coordinated FC: the optimal cycle length is 44 seconds, just as in Section 8.5.4. The waiting time at the internal queues under RV1(5) is reduced from 14.4 to 9.3 seconds, since another set of relative values is used. The overall average waiting time per car under RV1(5) is only 4.6% higher than when all queues have identical weights, while the waiting time at the internal queues is reduced by 35%.

The best FC yields the same waiting time as before, since the length of the green periods have not changed. The average waiting time at the internal queues is $16.7 - 9.3 = 7.4$ seconds, or almost 80%, higher than under RV1(5) with the modified cost structure.

Conclusions

- RV1(5) is more reactive to changes in the cost structure than FC. The cost structure reflects the weights of each flow.
- By changing the weights of the flows, one can improve the progression along the arterial under RV1(5).
- RV1(5) yields a much lower average waiting time at the internal queues than under coordinated FC, which aims at minimizing the long-run average waiting time per car.

8.6.2 Scenario II-B – More slots of information on internal arrivals (I2F4C2)

Instead of changing the weights of the flows, one may question whether changing the number of slots of arrival information to be used at each flow affects the performance of the RV policies. In Section 7.5, we have demonstrated, for an isolated F12C4 intersection, that considering more than 5 slots of arrival information for all flows hardly gives any improvement. In a network setting, it may be favorable to extend the number of slots of arrival information for the internal flows, from say 5 to 10 slots. By simulation we check whether the resulting RV1(·) policies yield a reduction of the waiting time at the internal queues. . From the second line of Table 8.10, we conclude that more than 5 slots of arrival information does not necessarily yield better decisions under RV1(·).

Next, we test whether ignoring arrival information at the external queues helps to increase the progression of the internal flows. As reported on the third line of Table 8.10, the average waiting time at the internal queues has decreased slightly by 0.3 seconds, while at the external queues the waiting time has increased by more than 1.5 seconds.

For completeness, we have copied the results of the case where we use no arrival information at all. As we have seen before the average waiting time per car is then about 12% higher than when 5 slots of arrival information is used for all flows.

Conclusion – Although using a few slots of arrival information significantly reduces the overall average waiting time, arrival information is hardly an effective tool for reducing the average waiting time at the internal queues. Changing the cost parameters seems to be much more effective.

Table 8.10: Mean waiting times under RV1(·) (in sec.) for I2F4C2 at $\rho = 0.8$ ($\lambda_f^i = 0.4$): different amounts of arrival information.

# slots arrival information			% above	Internal flows	External flows	
internal	external	EW	RV1(5)	EW_2^1, EW_4^2	EW_4^1, EW_2^2	EW_1^i, EW_3^i
5	5	14.7		14.4	14.8	14.8
10	5	14.8	+0.5%	14.5	14.7	15.0
5	0	15.9	+7.8%	14.1	16.9	16.3
0	0	16.5	+12.2%	16.7	16.3	16.6

8.6.3 Scenario II-C – High arrival rates in the West-to-East direction (I2F4C2)

When investigating the progression along the arterial, an interesting scenario is when more cars travel from west to east than in the other directions. Coordinated control in this direction may then be of high importance to reduce the overall average waiting time per car. We study the I2F4C2 case with $\lambda_f^i = 0.2$ at the external, except at Q_4^1, at which three times as many cars arrive ($\lambda_4^1 = 0.6$). The load of intersection 1, ρ_1, equals thus 0.8. Since cars turn right with probability 0.25, the arrival rate at Q_4^2 equals $0.25 \cdot 0.2 + 0.75 \cdot 0.6 = 0.5$. The load of intersection 2, ρ_2, is thus only 0.7.

Since intersection 2 has a lower load than intersection 1, the locally optimal cycle length of the two intersections differ: $D_1 = 46$ and $D_2 = 28$ seconds. For coordinated FC the cycle lengths as well as the length of the green periods must be identical for the two intersection. Given the asymmetric arrival rates, one may favor a green wave primarily at Q_4^2, although a green wave cannot be obtained for all cars. Therefore we set the cycle length to $D_i = 46$ at both intersections. The duration of the effective green period is 12 and 30 seconds for respectively the combinations 1 and 2 of both intersections. The best offset, ψ_2, is 40 (or 38) seconds, as depicted in Figure 8.7.

This case is not only of interest to study the progression in one direction, but also because of the asymmetry in the workload of the two intersections.

In Table 8.11 we report the long-run average waiting time per car under the various

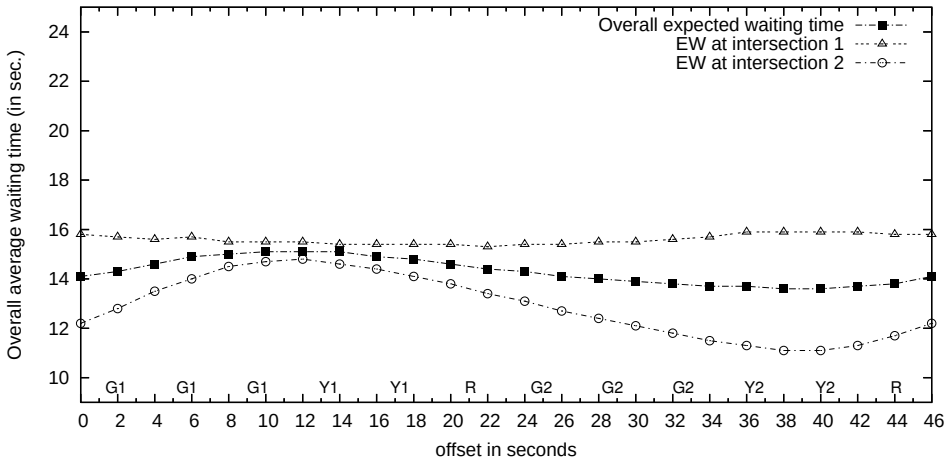


Figure 8.7: Optimization of FC over offset for $D = 46$: skew I2F4C2 case with different car speeds and 25% turns right.

Table 8.11: Mean waiting times (in sec.) for asymmetric I2F4C2 at $\rho_1 = 0.8$ and $\rho_2 = 0.7$ ($\lambda_f^i = 0.2$, except $\lambda_4^1 = 0.6$ and $\lambda_4^2 = 0.5$): different car speeds.

		% above			Internal flows		External flows			
Policy	EW	RV1(5)	EW_1	EW_2	EW_2^1	EW_4^2	EW_4^1	EW_2^2	EW_1^1, EW_3^1	EW_1^2, EW_3^2
RV1(5)[a]	8.7		10.2	7.1	4.0	5.5	7.9	4.1	16.7	10.5
RV1[a]	10.7	+23.0%	12.3	9.0	4.1	8.1	10.2	4.1	19.5	12.5
FC $D_1 \neq D_2$[a]	13.5	+55.2%	15.6	11.3	4.4	12.1	14.3	4.1	23.1	13.9
FC $D_i = 46$[b]	13.6	+56.3%	15.9	11.1	6.4	4.2	14.3	4.3	23.2	23.1
XC-2	10.8	+24.1%	12.2	9.2	4.4	10.1	12.5	4.6	15.7	10.4

[a](Relative values of) FC with $D_1 = 46$, $D_2 = 28$, and $d_{1,1} = 12$, $d_{1,2} = 30$, $d_{2,1} = 8$, $d_{2,2} = 16$.
[b]FC with $D_i = 46, d_{i,1} = 12, d_{i,2} = 30$, and $\psi_2 = 40$

policies. With respect to the overall average waiting time, coordinated FC hardly improves non-coordinated FC with locally optimal cycle lengths. However, the waiting time at Q_4^2 under coordinated FC is only a third of that under non-coordinated FC: 4.2 seconds versus 12.1 seconds.

RV1(5) is a very good dynamic alternative to FC. The overall average waiting time is only 8.7 seconds, which is a third (4.9 seconds) below that under coordinated FC. Under RV1(5), the waiting time at Q_4^2 is only 1.3 seconds higher, while the waiting time at all queues is much lower. Though the I2F4C2 infrastructure consists of simple intersections, XC-2 results in an average waiting time that is 24% above RV1(5).

8.6.4 Conclusions

RV1(5) results in a very low overall average waiting time per car, and allows to steer on a good progression in an arterial. The average waiting times at the internal queues can be reduced, by increasing the weights (=waiting costs per car) at these queues. Evaluating an horizon of future arrivals for more than 5 slots at these queues, does not improve the progression.

For an asymmetric arterial, where most traffic travels from west to east and the load of the two intersections differ, one may aim at setting a green wave under FC in the busiest direction. The best FC that we found appears to be more than 50% off from the overall average waiting time under RV1(5).

8.7 Study III and IV – More arterials

In this section, we report on Study III and Study IV, which relate to two different arterials:

Study III	–	I4F4C2	=	an arterial with four unequally spaced F4C2 intersections,
Study IV	–	I2F12C4	=	an arterial with two intersections of the F12C4-type.

The first arterial is of interest both for having more than two intersections and for having non-identical distances between the intersections. The second arterial is of interest for two reasons. Firstly, the number of flows in a combination is no longer identical and thus simple rules may not suit very well, as we have seen in the previous chapters. Secondly, through the arterial of F12C4 intersections one may test acyclic control policies in a network setting.

8.7.1 Study III - More intersections along an arterial (I4F4C2)

Thus far we have considered only arterials of 2 intersections. In practice one often deals with an arterial with multiple intersections at non-identical distance of each other. In Figure 8.8 we depict such an infrastructure with four intersections along an arterial.

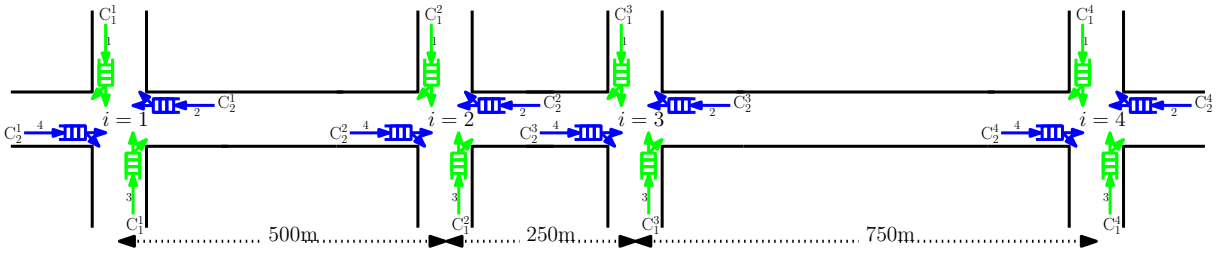


Figure 8.8: Arterial with four simple intersections: I4F4C2.

As in Scenario II-C, we assume non-identical arrival rates: much more cars travel from west to east than in the opposite direction. For the case that we study in this section, the arrival rates λ_f^i are 0.2, but the west-to-east traffic is three times as busy: $\lambda_4^i = 0.6$. Cars in the busy traffic flow, from west to east, turn right with probability 0.2. To keep the load of all intersections identical, we assume that at the other queues 60% of the cars turn right. The load of each intersections equals thus 80%.

Selection of coordinated FC

Given the insights from the previous sections, we synchronize FC by setting the cycle length and green times identical for all intersections, and carefully choosing the offsets ψ_i. The cycle length is set to the locally optimal one: $D = 46$ with $d_{i,1} = 12$ and $d_{i,2} = 30$ seconds. The distances between the intersections 1 and 2, 2 and 3, and, 3 and 4, is 500, 250, and 750 meters respectively. Optimal offsets under to-FC-ideal circumstances are $\psi_2 = 36$, $\psi_3 = 54$, and $\psi_4 = 108$ seconds. In fact, these offsets are the traveling times starting at intersection 1, when driving at constant speed of 50 km/h[2]. As a fraction of cars turns right, and as cars travel at different speeds, a perfect green wave cannot be set to all cars in the busy flow. Therefore we consider several sets of values for the offsets.

Average waiting time

In Table 8.12 we report several configurations of FC. The best FC yields an average waiting time that is 46% higher than that under RV1(5). We have seen similar results for the case of identical speeds. At the internal queues the waiting times are much lower than at the side streets. Compared to RV1(5), coordinated FC performs not so well, since the green periods are fixed, and are thus not adjusted to the actual number of cars queued. Furthermore, the coordination is complicated by the fact that, on average, a fifth of the cars arriving at Q_4^i (with $i \geq 2$), originate from Q_3^{i-1}.

Table 8.12: Mean waiting times (in sec.) for asymmetric I4F4C2 case ($\rho = 0.8$; for $f = 1, 2, 3$: $\lambda_f^i = 0.2$ and $P_f^i(\mathrm{R}) = 0.6$; $\lambda_4^i = 0.6$ and $P_4^i(\mathrm{R}) = 0.2$; and car speeds $\sim$ Tri(40,50,60).

		% above	West-to-East				Side streets
Policy	EW	RV1(5)	EW_4^1	EW_4^2	EW_4^3	EW_4^4	$EW_1^i = EW_3^i$
RV1(5)[a]	10.6		7.9	8.2	8.4	8.5	17.2
RV1[a]	12.9	+22.2 %	10.2	10.9	10.8	11.4	20.4
FC $\psi = (32, 46, 96)$[a]	15.6	+47.7 %	14.3	14.2	13.9	17.1	23.0
FC $\psi = (36, 54, 108)$[a]	15.5	+46.4 %	14.4	13.7	13.4	16.2	23.1
FC $\psi = (40, 62, 120)$[a]	15.6	+47.9 %	14.3	13.7	14.4	15.2	23.0
FC $\psi = (44, 70, 132)$[a]	16.0	+51.7 %	14.3	14.4	15.3	15.3	23.1
XC-2	12.7	+19.9 %	12.5	12.8	12.8	13.1	16.5

[a]Based on FC with $D = 46$, $d_{i,1} = 12$ and $d_{i,2} = 30$.

[2]According to the definition of ψ_i in Equation (8.2), the offsets should be $\psi_2 = 36$, $\psi_3 = 8$, and $\psi_4 = 16$ seconds, instead of $\psi_2 = 36$, $\psi_3 = 54$, and $\psi_4 = 108$. The latter choices give the same results, but emphasizes more the relation with the traveling time to intersection 1.

RV1(5) gives the lowest average waiting time. The average waiting time is lowest at the busiest queues. Apparently, these queues get high weights, likely because they contribute much to the total workload. The waiting time at the internal queues in the west-to-east direction is even more than 25% lower than under coordinated FC. Also the average waiting time per car at the side streets is lower under RV1(5) compared to the FC policies. Exhaustive control gives a slightly lower waiting time at the external queues, but at the expense of high average waiting times in the west-to-east direction. The overall average waiting time per car is about 20% above that of RV1(5).

8.7.2 Study IV – Arterial with more combinations (I2F12C4)

The I2F12C4 infrastructure, depicted in Figure 8.9, has separate lanes for right turning, left turning and thru traffic. At each internal queue, traffic may originate from three different queues of the upstream intersection. Thus, we cannot expect to set green waves to all cars arriving at the internal queue.

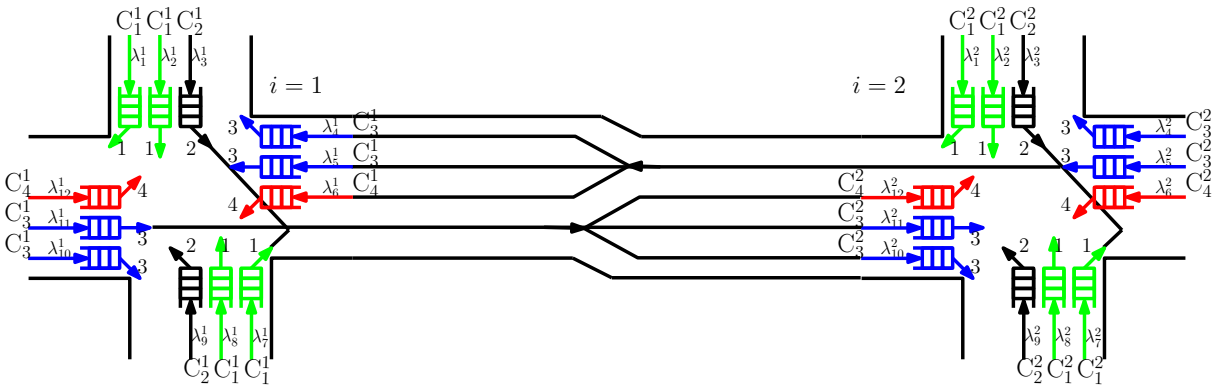


Figure 8.9: Typical example of arterial with two signalized intersections: I2F12C4.

The decision on when to switch from green to one combination, to green to another combination, is complicated by the asymmetry in the number of flows per combination. Further, under acyclic control, one should also decide which of the four combinations gets green next: thin combinations may get under-served, since serving a thick combination maybe more beneficial.

Some modeling assumptions and data

We adopt the modeling assumptions of the previous section(s), a.o.

- individual (desired) car speeds are drawn from Tri(40,50,60),
- the approach to and the distance between the intersections are 500 meters, and
- arrival information is available of cars that are up to 5 slots away from the end of the queue.

The arrival rates λ_f^i are set to 0.1 for all right and left turning flows. Thru traffic is three times as busy: $\lambda_2^i = \lambda_5^i = \lambda_8^i = \lambda_{11}^i = 0.3$ for each intersection i. The workload of each intersection is thus $\rho^i = 0.8$.

Upon leaving a queue a car chooses a lane according to fixed probabilities, as illustrated in Figure 8.10. Cars leaving Q_f^i proceed to a downstream approach and select with probability $P_f^i(\mathrm{L})$ the left lane, w.p. $P_f^i(\mathrm{T})$ the thru-lane and w.p. $P_f^i(\mathrm{R}) = 1 - P_f^i(\mathrm{L}) - P_f^i(\mathrm{T})$ the right lane. Cars stick to the lane that they have selected upon crossing the upstream intersection. Although the routing probabilities may depend on i and f, we choose them identical for all cars: $P_f^i(\mathrm{L}) = P_f^i(\mathrm{R}) = 0.2$, and $P_f^i(\mathrm{T}) = 0.6$.

Figure 8.11 shows how cars proceed from one intersection to another, and how the lane selection affects the dispersion of platoons. Therefore we have marked fictitious segments or cells, that cars can cross in one slot (when traveling 50km/h). A '1' in a cell indicates

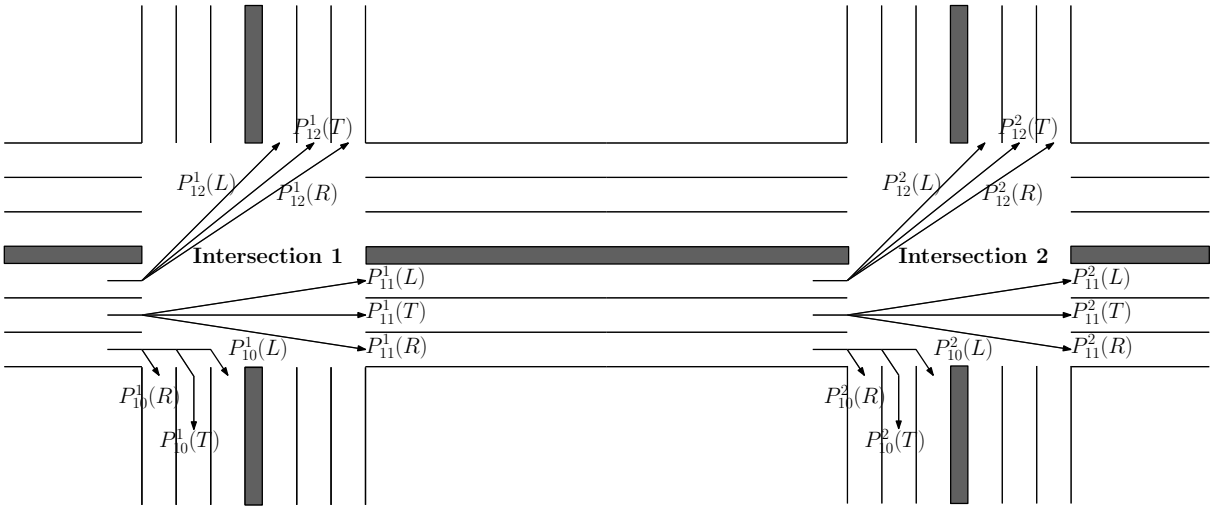


Figure 8.10: Routing probabilities in an arterial with two signalized intersections:

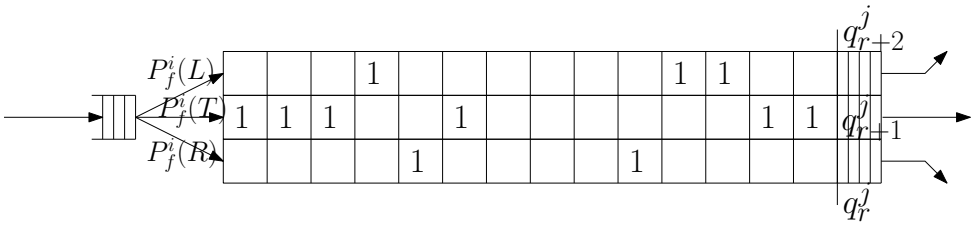


Figure 8.11: Routing probabilities and arrival information over three lanes.

that a car is present at that segment. The segments closest to the queues contains cars that are (at most) 1 slot away from the tail of the queue. Cars driving at the next-to closest segment, are still two slots away from the tail of queue. The segments not necessarily correspond to physical marked segments of the road, but are introduced to model the arrival times. Since cars leaving a single queue, have to choose between three lanes, cars do not travel in a single perfect platoon. When cars speeds differ, a platoon may be partly repaired by slow moving traffic as a car is not allowed to change lanes to overtake slow moving traffic on its lane.

Optimization of FC

The local optimization of FC for each intersection in isolation can be done by means of the Optimal-fixed-cycle algorithm (of Appendix D). In Figure 8.12 we show, for varying values of the cycle length D, the average waiting time per car under the best FC found by the algorithm. Apparently, the best FC for the F12C4 intersection, under the given arrival rates, is 84 seconds. The plot indicates also alternatives that are close to optimal. The local optima for a single intersection in isolation, are considered in finding a best coordinated FC.

For coordinated FC, one chooses both a cycle length D and an offset ψ_2, such that the overall average waiting time is minimal. According to the insights obtained in the previous sections, the offset should be zero or $D/2$, such that one experiences, in both horizontal directions, identical average waiting times per car. When the offset equals $D/2$, then the offset should be roughly equal to the average traveling time, between the two intersections. When driving at 50 km/h, the traveling time between the intersections is 36 seconds. According to Equations (8.1) and (8.2), $D = 72$ is a reasonable cycle length, since it is closest to the locally optimal FC (with cycle length 84 seconds). The FC with $D = 72$ and $\psi_2 = 36$, is, however, no local optimum in Figure 8.12.

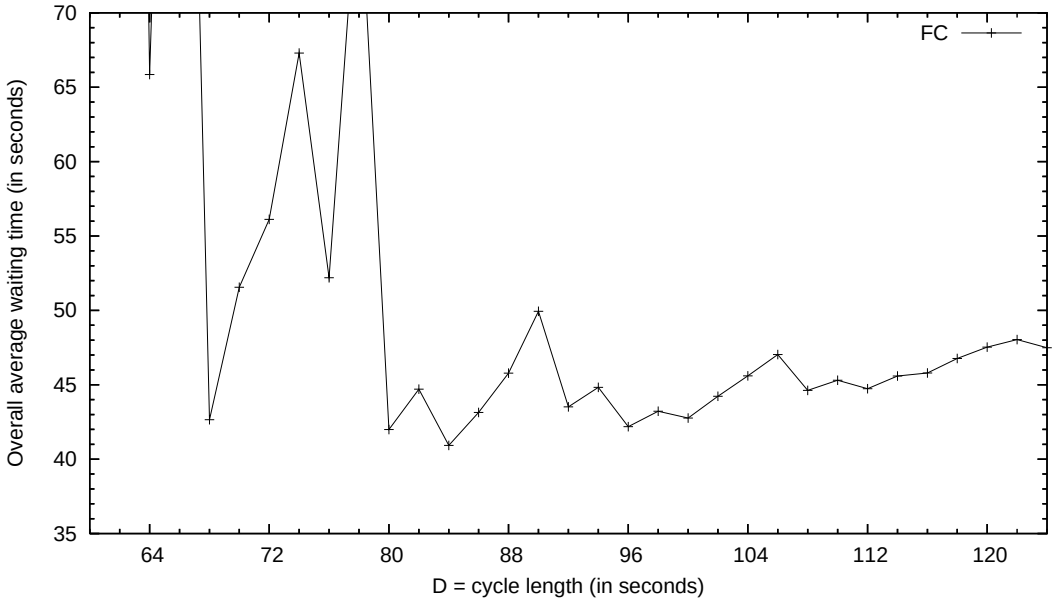


Figure 8.12: Application of the Optimal-fixed-cycle algorithm to find optimal value of the cycle length for F12C4 with non-identical arrival rates ($\lambda^i = \frac{1}{10}(1,3,1,1,3,1,1,3,1,1,3,1)$).

Therefore, we report, in Table 8.13, the average waiting time under FC with different cycle lengths, ranging from $D = 68$ to 88 seconds with increments of 4 seconds. In all configurations of FC, the offset is set to $\psi_2 = D/2$. The effective green times are reported in the second and third column. It appears that in this case the best coordinated FC has a cycle length equal to the locally optimal one: $D = 84$ seconds. The reported overall average waiting times, does not differ much from those of a single intersection in isolation. Apparently, the arrival pattern at the internal queues, do differ not much from the Binomial arrival pattern. This is due to the fact that cars need to select a lane after crossing an intersection and to the fact that their traveling speeds differ, causing platoons to disperse.

Cyclic control

In Table 8.14 we report on the average waiting time both under cyclic and acyclic policies. We split the overall average waiting time in the average waiting time at the lanes of the internal and of the external queues.

We first discuss the top half of Table 8.14, which contains the results under cyclic control. RV1(5), which is reported in the first row, is selected as a base for comparing all other

Table 8.13: Mean waiting times ((in sec.)) under coordinated FCs for I2F12C4 at $\rho_i = 0.8$ and thru traffic three times as busy than left and right turning traffic (i.e. $\lambda_f^i = 0.1$, except $\lambda_2^i = \lambda_5^i = \lambda_8^i = \lambda_{11}^i = 0.3$); car speeds $\sim$ Tri(40,50,60).

	Dep. times ($d_{i,f}$)		*EW*	Internal lanes			External lanes		
FC	C_1, C_3	C_2, C_4	overall	Left	Thru	Right	Left	Thru	Right
$D = 68, \psi_2 = 34$	22	8	43.6	62.8	39.2	10.6	65.8	46.8	18.6
$D = 72, \psi_2 = 36$	24	8	44.8	95.3	28.7	10.1	100.4	38.9	19.1
$D = 76, \psi_2 = 38$	26	8	50.8	45.4	55.2	11.2	47.0	65.7	21.1
$D = 80, \psi_2 = 40$	26	10	42.8	54.6	38.7	12.2	55.6	49.1	21.6
$D = 84, \psi_2 = 42$	*28*	*10*	*41.8*	*67.1*	*33.5*	*13.7*	*68.8*	*42.6*	*22.1*
$D = 88, \psi_2 = 44$	30	10	44.2	85.9	32.3	15.4	89.0	39.4	22.6
$D = 92, \psi_2 = 46$	30	12	45.2	51.3	45.4	17.7	53.9	51.5	24.6
$D = 96, \psi_2 = 48$	32	12	44.1	59.2	42.0	19.6	61.5	46.1	25.2

policies. RV1, which ignores any available arrival information, yields a somewhat higher overall average waiting time: +6.1%. Seemingly, considering arrival information is more important at the simple F4C2 intersections of the previous sections than at the more complex F12C4 intersections. The best coordinated FC results in an average waiting time per car that is 29% above that under RV1(5). The best cyclic exhaustive control policy (XC-2) yields a much higher average waiting time: 18.1% above RV1(5).

Table 8.14: Mean waiting times (in sec.) for I2F12C4 at $\rho_i = 0.8$ and thru traffic three times as busy than left and right turning traffic (i.e. $\lambda_f^i = 0.1$, except $\lambda_2^i = \lambda_5^i = \lambda_8^i = \lambda_{11}^i = 0.3$); car speeds $\sim$ Tri(40,50,60).

		% above	Internal lanes			External lanes		
Policy	EW	RV1(5)	Left	Thru	Right	Left	Thru	Right
Cyclic policies:								
RV1(5)[a]	32.3		44.5	29.9	17.5	45.9	32.3	20.9
RV1[a]	34.3	+6.1%	47.1	33.9	20.8	47.5	34.4	21.4
FC ($D = 84, \psi_2 = 42$)	41.8	+29.3%	67.1	33.5	13.7	68.8	42.6	22.1
XC-2	38.4	+18.9%	49.2	38.3	27.0	48.6	38.9	27.5
Acyclic policies:								
RV1M(5)[a]	31.7	−1.7%	48.7	29.3	18.2	48.9	30.2	19.6
RV1M[a]	33.4	+3.3%	51.2	31.8	19.8	51.4	31.9	19.9
XA-2	37.7	+16.8%	73.8	29.4	21.2	74.7	31.3	22.8

[a]Based on FC with $D = 84$, $\psi_2 = 0$, $d_{i,1} = d_{i,3} = 28$ and $d_{i,2} = d_{i,4} = 10$.

Looking at the average waiting times at the different lanes, we observe that, on average, left turning cars are queued longest. Right turning traffic instead, enjoys a relatively long green period and thus has to wait shortest, since it is part of thick combinations. The waiting time for thru traffic is not very high, as the thru traffic plays a dominant role for having the highest arrival rate.

The highest average waiting times under cyclic control are under FC, in particular to left turning traffic. Under RV1(5) the average waiting times at all queues, except at Q_6^1 and Q_{12}^2, are much lower than under any of the other policies.

Acyclic control

When the control of the lights is no longer restricted to cyclic policies, one may select any combination to serve next after an all red phase. Under the acyclic exhaustive control policies that we have considered, one switches, after an all red slot, to the combination that takes longest until it is exhausted: the expected time to exhaustion is estimated based on the actual queue lengths and the constant arrival and departure rates.

Under exhaustive control we now observe much higher average waiting times to left turning traffic: at these queues XA2 results in an average waiting time that is 50% higher than under XC-2. Under the acyclic RV1 policies the average waiting times are much lower.

Regarding the overall average waiting time the difference between RV1(5) and the acyclic RV policies is only 1.7%. Since the difference is very small, cyclic policies may be preferred for psychological reasons. Although not reported, we have not seen a great improvement by RV2M and RV2M(5).

Conclusions

- Even for the more complex intersection with 4 combinations, cyclic control performs almost as good as an acyclic control policy.
- Adding arrival information to an RV policy gives a much smaller improvement for F12C4 intersections than for F4C2 intersections.
- RV1 and RV1(5) greatly improve the best FC: the waiting time at most internal and external queues are significantly lower under RV1(5).
- Under the RV1 policies, the difference between internal and external queues is very small.

8.8 Study V – Network of 9 intersections (I3x3F4C2)

8.8.1 Problem and data

In this final section, we demonstrate the application of the RV rules to a network of intersections on a grid. We consider the network depicted in Figure 8.13, which we refer to as the I3x3F4C2 infrastructure. The nine intersections of the F4C2 type are numbered 1 to 9 (row-by-row and from left to right). The distance between any two neighboring intersections is 500 meters.

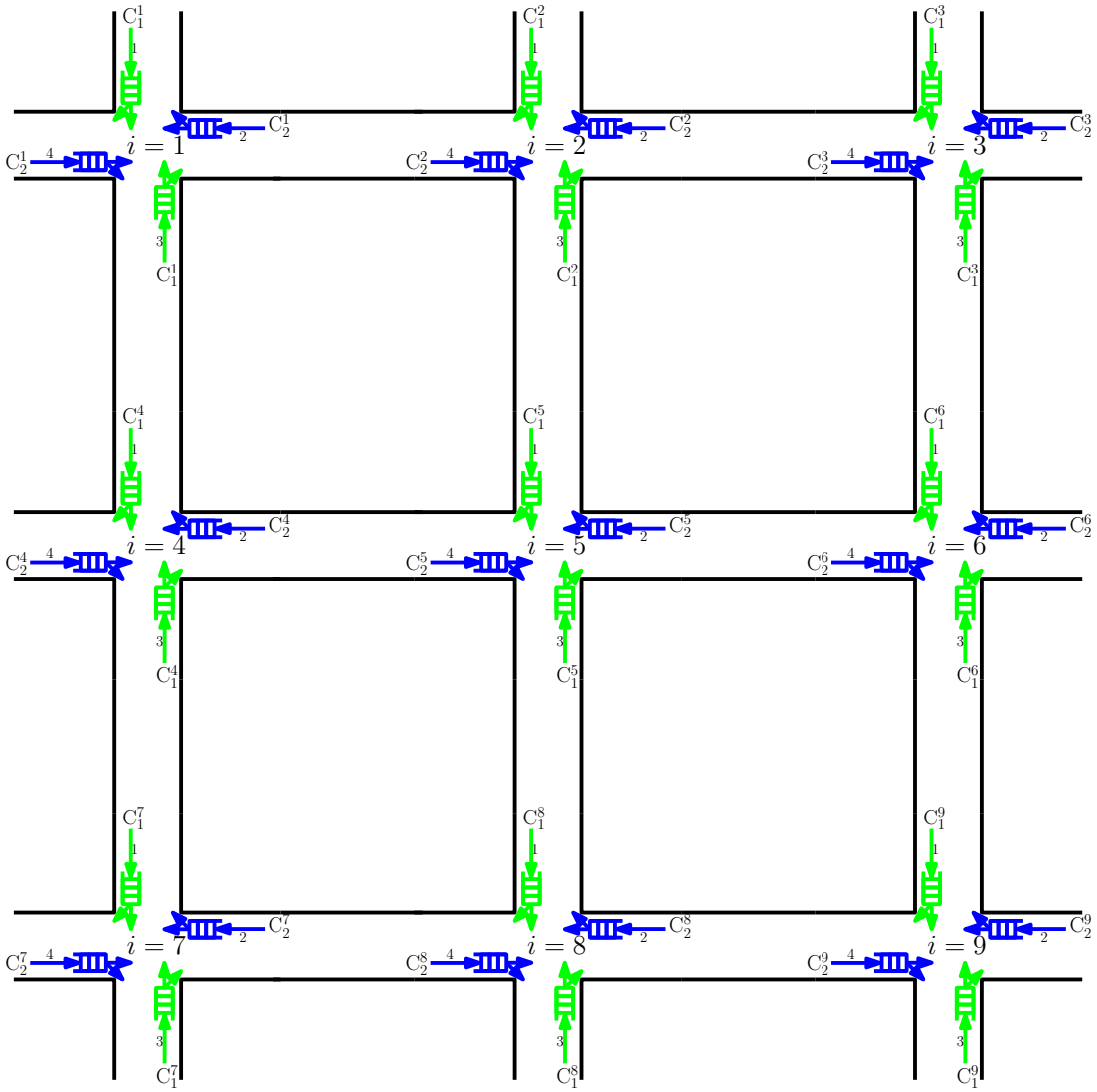


Figure 8.13: Network of 9 simple intersections: I3x3F4C2.

Under to-FC-ideal conditions (identical car speeds and thru traffic only), green waves may be set in both directions. Then the waiting time at intersection 5 can be zero under coordinated FC, as intersection 5 is the center of the network and is fed only by traffic leaving the neighboring intersections. However, we believe that such conditions are not very realistic. Therefore we assume desired car speeds to vary according to a Triangular distribution with mean 50 km/h and minimum and maximum speeds 40 and 60 km/h respectively.

We assume flow-dependent arrival rates that are identical to all intersections: $(\lambda_1^i, \lambda_2^i, \lambda_3^i, \lambda_4^i) = (0.32, 0.16, 0.16, 0.48)$. The load of each intersection is thus 80%. At each intersection, the busiest flow is flow 4 that has on average 3 times as many cars as flows 2 and 3. The load of flow 1 is two times that of flows 2 and 3.

A fraction of the cars may turn right. For the given infrastructure there is no left turning traffic and thus $P_f^i(\mathrm{T}) = 1 - P_f^i(\mathrm{R})$ for all flows f. The flow-dependent routing probabilities are such that the arrival rates (λ_f^i) are preserved. Hence $P_f^i(\mathrm{R}) = P_f(\mathrm{R})$ must satisfy the set of balance equations (8.3) – (8.6) for flow 1 to flow 4 at all intersections.

$$0.32 = 0.32 \cdot (1 - P_1(\mathrm{R})) + 0.48 \cdot P_4(\mathrm{R}) \tag{8.3}$$

$$0.16 = 0.16 \cdot (1 - P_2(\mathrm{R})) + 0.32 \cdot P_1(\mathrm{R}) \tag{8.4}$$

$$0.16 = 0.16 \cdot (1 - P_3(\mathrm{R})) + 0.16 \cdot P_2(\mathrm{R}) \tag{8.5}$$

$$0.48 = 0.48 \cdot (1 - P_4(\mathrm{R})) + 0.16 \cdot P_3(\mathrm{R}) \tag{8.6}$$

$$P_1(\mathrm{R}),\ P_2(\mathrm{R}),\ P_3(\mathrm{R}),\ P_4(\mathrm{R}) \in [0, 1] \tag{8.7}$$

One of the (infinitely many) solutions is $P_1(\mathrm{R}) = 0.15$, $P_2(\mathrm{R}) = 0.3$, $P_3(\mathrm{R}) = 0.3$, and $P_4(\mathrm{R}) = 0.1$. Then on average 17% of the cars turns right. This case is considered throughout the next subsections.

Since the routing probabilities are memoryless, a car may pass an intersection multiple times before actually leaving the network. Given the simulation model, we have to accept this phenomenon, but this can be avoided easily when the routes that cars will travel are set upon the generation of car arrivals at the borders of the network.

8.8.2 Optimization of coordinated FC

For the computation of the relative values required for the RV rules, we simply optimize FC for an intersection as if it is in isolation. A locally optimal uncoordinated FC can be found by applying the Optimal-fixed-cycle algorithm. Figure 8.14 shows the incremental search process: only the best increments of the cycle length by 1 slot are accepted and reported in the figure. Due to the asymmetry in the arrival rates, the local optima are unequally spread over the range of considered cycle lengths. The best cycle length seems to be $D = 44$ seconds. The duration of the effective green periods of the two combinations are then 16 and 24 seconds, for C_1 respectively C_2.

The best coordinated FC likely has a cycle length not too far from this optimum value. To be sure to compare the RV policies to the best possible coordinated FC, we check a whole range of cycle lengths D and offsets $(\psi_1, \psi_2, \psi_3, \ldots, \psi_9)$. The Optimal-fixed-cycle algorithm helps in finding a set of promising cycle lengths. Related to each cycle length D the algorithm gives for each combinations s a best effective green time $d_{i,s}$.

The number of choices for the offsets, can be limited since the infrastructure is well-structured: the traveling time between any two adjacent intersections is the same. Therefore we only consider offsets that satisfy: $(\psi_1, \psi_2, \ldots, \psi_9) = (0, 1, 2,\ 1, 2, 3,\ 2, 3, 4)\psi$. A

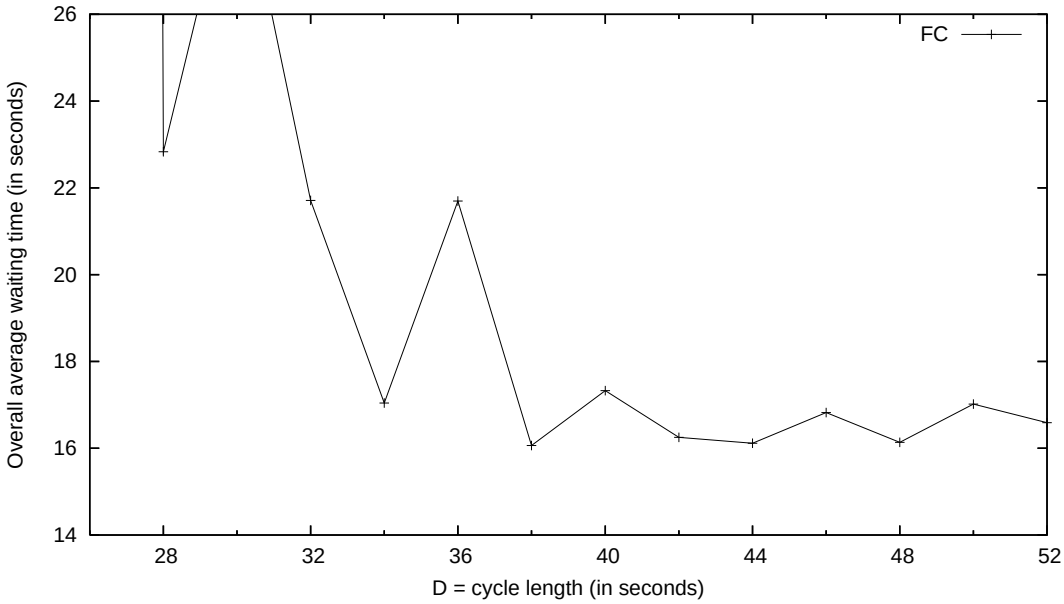


Figure 8.14: Application of the Optimal-fixed-cycle algorithm to determine (by simulation) optimal value of the cycle length D for F4C2 with non-identical arrival rates $(\lambda_1^i, \ldots, \lambda_4^i) = (0.32, 0.16, 0.16, 0.48)$); car speeds $\sim$ Tri(40,50,60).

wide range of values of ψ is considered: $\psi \in \{34, 36, \ldots, 48\}$ seconds. All pairs (D, ψ) are evaluated by simulating the entire network. The best pair (D, ψ) appears to be $(44, 42)$ seconds, with effective green periods of $d_{i,1} = 16$ and $d_{i,2} = 24$ seconds for C_1^i and C_2^i respectively.

8.8.3 Comparison of policies

In Table 8.15 we report the long-run average waiting times at an intersection per car under the various policies. To judge the quality of each policy, we primarily focus on the overall average waiting time, EW (which is the average waiting time that a car experiences at an (arbitrary) intersection). Since intersection 5 is fed by internal arrivals only, we also study the waiting time (EW^5) at this intersection. The progression at this central intersection is investigated by looking at the average waiting time at each queue of this central intersection: EW_1^5 to EW_4^5.

The overall average waiting time (per car per intersection) under the best coordinated FC appears to be about 29% above that under RV1(5). The best exhaustive control policy, XC-2, yields a lower waiting time than coordinated FC, but is still 21% above that of RV1(5). The use of arrival information turns out to be crucial to the RV policies: with no slots of arrival information RV1 performs a bit worse than coordinated FC.

In the case reported in Table 8.15, the control of the lights is dominated by the queue lengths of Q_1^i and Q_4^i. The system approximately behaves as a polling system, since each combination consists of only 1 thick flow next to a thin flow. For polling models with a similar cost structure, exhaustive control is optimal. It is thus not surprising that RV1

Table 8.15: Mean waiting times (in sec.) for skew I3x3F4C2 at $\rho_i = 0.8$ ($\lambda_1^i = 0.32$, $\lambda_2^i = \lambda_3^i = 0.16$ and $\lambda_4^i = 0.48$); 17% of cars turns right and car speeds $\sim$ Tri(40,50,60).

		% above	Waiting time at intersection 5				
Policy	EW	RV1(5)	EW^5	EW_1^5	EW_2^5	EW_3^5	EW_4^5
RV1(5)[a]	9.6		9.4	11.8	6.0	10.1	8.6
RV1[a]	12.7	+33%	13.1	17.2	6.1	11.9	13.2
FC, coordinated[b]	12.3	+29%	10.2	14.5	7.4	8.7	8.8
XC-2	11.6	+21%	11.7	14.0	6.0	10.2	12.6

[a]Based on FC with $D = 44$, $d_{i,1} = 16$ and $d_{i,2} = 24$ seconds.
[b]FC with $D_i = 44$, $\psi = 42$, $d_{i,1} = 16$ and $d_{i,2} = 24$ seconds.

does not beat XC-2. (As we have seen in Chapter 6, XC-2 performs well, in particular, for cases where a single flow dominates all other flows in the same combination. RV1 performs very well for both simple cases and more complex cases with multiple thick flows in the same combination or with identical arrival rates per flow.)

With respect to the progression at the internal lanes leading to intersection 5, RV1(5) performs better than coordinated FC, except at flow 3. C_1^5, which contains flow 3, is the thinnest combination of the two combinations. Since the thickest combination, C_2^5, dominates over C_1^5, the highest average waiting time per car is observed at the thickest flow of C_1^5: flow 1. Analogously, the lowest average waiting time is observed at the thinnest flow of the thickest combination: flow 2.

Conclusions for I3x3F4C2

For I3x3F4C2, with varying car speeds and with on average 17% of the cars turns right, we conclude that RV1(5) is favored above coordinated FC for three reasons:

1. RV1(5) yields the lowest overall average waiting time per car,
2. the average waiting time per car under RV1(5) is lower at most of the internal queues than under coordinated FC, and
3. setting an initial FC for RV1(5) is much easier than finding a good coordinated FC.

8.8.4 Sensitivity regarding the fraction of cars that turns right

We consider three different scenarios to investigate the impact of routing probabilities on the average waiting time per car at an arbitrary intersection under the different policies. The three scenarios, labeled V-A to V-C, are presented in Table 8.16. In Scenario V-A no cars turn right. Scenario V-B is the one studied in the previous sections. In Scenario V-C twice as many cars turn right as in Scenario V-B.

Table 8.16: Scenarios for fraction of cars that turns right in I3x3F4C2 network.

		Routing probabilities			
Scenario	Description	P_1(R)	P_2(R)	P_3(R)	P_4(R)
V-A	No cars turn right	0	0	0	0
V-B	About 17% of the cars turns right	0.15	0.3	0.3	0.1
V-C	About 34% of the cars turns right	0.30	0.6	0.6	0.2

Table 8.17: Mean waiting times (in sec.) for skew I3x3F4C2 at $\rho^i = 0.8$ ($\lambda_1^i = 0.32, \lambda_2^i = \lambda_3^i = 0.16$ and $\lambda_4^i = 0.48$): different percentages of right turning traffic.

	Scenario V-A		Scenario V-B		Scenario V-C	
Policy	No cars turn right		17% turns right		34% turns right	
RV1(5)[a]	8.9		9.6		10.0	
RV1[a]	12.9	+44%	12.7	+33%	12.6	+27%
FC, coordinated	8.8[b]	−0.9%	12.3[c]	+29%	14.4[d]	+44%
XC-2	11.6	+30%	11.6	+21%	11.6	+16%

[a]Based on FC with $D_i = 44$, $d_{i,1} = 16$ and $d_{i,2} = 24$ seconds.
[b]FC with $D_i = 42$, $\psi = 42$, $d_{i,1} = 16$ and $d_{i,2} = 22$ seconds.
[c]FC with $D_i = 44$, $\psi = 42$, $d_{i,1} = 16$ and $d_{i,2} = 24$ seconds.
[d]FC with $D_i = 48$, $\psi = 40$, $d_{i,1} = 18$ and $d_{i,2} = 26$ seconds.

In Table 8.17 we present the overall average waiting time per car for all three scenarios. As one may expect the waiting time is lowest when all traffic is thru traffic. Setting coordinated FC is then relatively easy, but since car speeds differ it is not possible to set a green wave to all cars arriving at the internal queues. When no cars turn right, the best coordinated FC performs slightly better than RV1(5), since platoons of cars are less dispersed and all cars arriving at an internal queue originate from the same queue.

When the percentage of right turning traffic increases, the impact of the arrival information reduces, as becomes clear from comparing RV1(5) to RV1. The gap between FC and RV1(5) grows when more traffic turns right, since it becomes much harder for FC to coordinate the lights. (This statement may no longer hold when more than half of the cars turn right, since then the optimal coordinated FC would have a different offset.) XC-2 is not sensitive to changes in the fraction of cars that turn right. The relative performance of XC-2, compared to RV1(5), improves when more cars turn right.

For the given network of nine F4C2 intersections, XC-2 may beat RV1 but not RV1(5). XC-2 performs quite well, since the arrival rates are very skew such that one flow dominates in each combination. The system then behaves like a polling system. However when the arrival rates of flows in the same combination would be close close to each other, or when the intersections are more complex, the difference between RV1(5) and XC-2 becomes bigger as we have seen before. We believe that RV1(5) is a promising policy that gives very good results and that is often superior to all other strategies.

8.9 Conclusions

In this chapter we have learned that controlling the traffic lights in a network of intersections is a challenging task. In general an optimal dynamic, state-dependent solution, which minimizes the average waiting time per car per intersection, cannot be obtained.

We have paid considerable attention to how to coordinate the traffic lights at different intersections, since coordinated control contributes to the minimization of the average waiting time per car. With respect to coordinated FC, we draw the following conclusions:

- Under FC so-called green waves can be set in multiple directions, but only under ideal conditions and for a limited number of cycle lengths.
- Green waves are easily lost when
  - platoons break due to different car speeds, and
  - cars may turn right, as internal arrivals originate then from different queues.
- Finding a well-coordinated FC requires an extensive search for a good network cycle $(D, d_{i,s}, \psi_i)$. This search can be streamlined by using experience, good intuition and insights for the specific infrastructure.

To reduce the long-run average waiting time per car, dynamic control based on local information may be preferred over coordinated FC. Centralized coordinated dynamic control may be quite complicated: when the traveling time between any two neighboring intersections is T seconds, it requires to coordinate the current decision with the decisions taken T seconds ago at the neighboring intersections. Even when the historic information on the state of neighboring lights is available to the controller, it might not be very helpful in predicting car arrivals, as cars speeds may vary. Then it might be better to use accurate information on the current state of the traffic lights and accurate estimates of arrival times at each intersection.

Therefore we propose the one-step policy improvement algorithm, RV1(5), which was already introduced in Chapter 7. Coordinated FC may outperform RV1(5) under (utmost) ideal conditions for FC. As often the conditions are not ideal, RV1(5) is preferred for three reasons:

- RV1(5) is easy to configure, since the optimization of the underlying FC is quite simple: the Optimal-fixed-cycle algorithm provides the locally optimal FCs for each intersection in isolation.

- RV1(5) looks ahead a number of slots for which estimated arrival times of cars are known, and thus considers the most accurate information available to synchronize the control of the lights.
- RV1(5) gives great improvements over coordinated FC and XC-2, especially for a number of complex cases where cars speeds may differ and where:
  - the intersections have unequal load, or
  - the intersections are unequally spaced, or
  - multiple flows are combined in the same combination, or
  - the arrivals at an internal queue originate from different upstream queues, or
  - all flows within the same combination have virtually identical arrival rates, such that none of the flows dominate.

For most of the network scenarios that we have studied, we have seen that even the best coordinated FC may yield an average waiting time per car that is 28–56% higher than that under RV1(5). In some cases XC-2 performs better than coordinated FC, but in most cases the waiting time is still about 20% above that under RV1(5).

Chapter 9

Epilogue

The topic of this thesis is large structured Markov decision problems (MDP). With large we mean 'too large to solve the MDP to optimality'. With 'structured' we indicate that many MDPs exhibit some structure that can be exploited in the search for an approximate solution. Many real-life sequential decision problems can be formulated as an MDP, but too often it is believed that the MDP's state space is too large to compute an optimal strategy. The tendency for multi-dimensional MDPs to become too large to solve, is often called the *curse-of-dimensionality*. Nevertheless, the formulation of a problem as an MDP maybe very helpful in obtaining an approximate solution using existing techniques.

Two approaches that are widely used in many mathematical programs for solving large problems are aggregation and decomposition. The possibilities to aggregate or to decompose an MDP strongly depend on the specific properties of the problem that define the problem structure. Another approach to solve high-dimensional MDPs, is approximate dynamic programming [117]. Although having the same objective, we consider approximate dynamic programming as another line of research.

For two entirely different applications, we have shown that aggregation and decomposition can be applied successfully to solve MDPs that arise in real life. The two applications may be inspiring for solving similar problems. In the first part of the thesis, we have focused on an application in which aggregation is used to reduce the number of states in the MDP. In the second part decomposition of the state space is used to obtain an approximate solution through a one-step-policy improvement algorithm.

A production-inventory-problem of perishables with a short fixed shelf life

Although the platelet production-inventory problem (PPP) is a high-dimensional problem, the problem exhibits enough structure to formulate an MDP problem and solve it. Solving the MDP of the PPP at a Dutch blood bank, requires the aggregation of states such that an SA algorithm can be executed in a reasonable amount of time. Aggregation is needed at several levels. First of all, the blood groups are to be aggregated as if blood platelet pools (BPPs) are of a single universal blood group. Secondly, BPPs are aggregated in batches when the PPP plays at a large scale. Finally, to come up with a simple rule, the stock consisting of batches of different age categories are aggregated into a single number. By simulation the structure of the optimal strategy is investigated and approximated by simple rules.

Whether the state aggregation at all three levels is possible depends on the context of the problem as well as on the actual data. In cases where other standards would apply concerning the compatibility of the different blood groups, or when the supply of buffy coats is limiting, the aggregation of blood groups into a single universal blood group may not be justified. When blood groups have to be matched strictly, aggregation at this first level is not justified. Then one may decompose the problem and solve the ordering problem for each blood group in isolation of all others.

In some cases the other two aggregation steps are not required or not justified. On the one hand, when the demand for BPPs is very low, there may be no need for aggregating BPPs into batches. On the other hand, when the coefficient of variation of the demand is high, e.g. because demand levels are low, a simple order-up-to S rule may be too far from optimal. Then the SDP-Simulation approach can be used for deriving more advanced stock-level-dependent and stock-age-dependent ordering rules.

For the PPP with realistic data from a Dutch blood bank, aggregation at all three levels is possible and needed. For the given data, it appears that an order-up-to S rule roughly fits to the optimal ordering decisions. Even for smaller cases, such as that of a medium-large hospital, the order-up-to S rule performs quite well. The inclusion of fixed order costs is no problem, although it gives rise to order-up-to rules with thresholds for ordering, such as (s, S)-policies.

The SDP-Simulation approach does not only give insight in the structure of an optimal policy, but also translates it, when possible, into a simple rule and yields the corresponding (nearly) optimal parameters values. The approach seems to be fruitful for applying it to other production-inventory settings that involve perishables with a short fixed shelf life.

The software developed during this PhD project has resulted in an application named TIMO, which is currently implemented at one of the Dutch blood banks. The application shows the usefulness of Stochastic Dynamic Programming and aggregation techniques in solving problems that seem to be intractable at first sight. The application of the SDP-Simulation approach is not straight-forward and requires insight into the problem to deal with the involved modeling steps.

The dynamic control of traffic lights: a queueing problem

The dynamic control of traffic lights is a typical queueing problem, where the intersection can be seen as a server which is able to serve multiple, but not all, queues at a time. When switching, from serving one set of queues to serving another set, a fixed switch-over time applies. During the first part of the switch-over time the service of a queue might continue, e.g. as long as the light stays yellow, but the switching process cannot be interrupted. At the last part of a switching phase, no queues get served, e.g. an all red phase applies to clear the intersection.

For the dynamic control of traffic lights optimal decisions end green periods and specify the order in which queues are served, such that the long-run average waiting time is minimized. Input to the control are the number of cars waiting at each queue as well as the state of the traffic lights. Given the potential number of cars waiting at each queue, the number of states explodes rapidly when the number of queues is increased. Whereas an optimal policy for a small intersection, with say only 4 flows is tractable, one has to rely on approximate solutions for intersections with many queues.

The approximation that we discuss is a one-step policy improvement over a well-structured policy that allows decomposition of the state space. For the control of traffic lights, the control according to a fixed cycle (FC) allows for such a decomposition. Under FC the waiting time at each flow can be evaluated flow-by-flow after formulating a Markov chain. The relative values (RV) of the states under FC are input to a one-step policy improvement algorithm.

The most simple improvement rule is the policy that we called RV1: every time slot FC may be interrupted. Therefore all feasible jumps within the cycle are considered and evaluated using the relative values under FC. In the evaluation of a time jump it is assumed that the FC resumes after the jump as if it is not interrupted in the future.

RV1 appears to significantly improve FC and other control policies such as anticipative exhaustive control. There are simple infrastructures for which the difference between

anticipative exhaustive control and RV1 is small. For more complex intersections, where multiple (non-conflicting) flows receive green at the same time, RV1 gives an easy way out of the complex sequential decision problem.

A problem similar to the control of traffic lights may be observed in a production-inventory setting. A production facility, or production department, produces various products. Suppose it is able to work at only a few product series at a time, and switching between series of different products requires a switch-over time, e.g. for cleaning and setting-up machines and instructing operators and other staff. One has to decide (dynamically) on the length of a product run, and which product run to set-up next. On the one hand, the decisions should balance the probabilities of getting out of stock for one or more products: short product runs may therefore be favored. On the other hand, short production runs for each product may be very inefficient, as switching implies switching costs and some set-up time. Optimal decisions on the order and the length of the production runs depend, amongst other, on the actual stock-levels of all products and on the demand distribution.

An approximation of this production-inventory problem, is by imposing a special policy that imposes additional structure to the problem such that the state space can be decomposed. Such a policy may be a cyclic production schedule, in which the order as well the length of production runs are fixed. A one-step policy improvement approach for this problem is considered by Bruin and Van der Wal ([22] and [23]).

Aggregation or Decomposition?

It is hard to formulate general statements on when aggregation or decomposition is possible, as it depends strongly on both the problem and the data. Without claiming any generality, we give some thoughts at a high level.

In general decomposition of a problem implies the distinction of subproblems that can be solved one-by-one. When solving an MDP, the state space can sometimes be decomposed when one restricts to a special class of well-structured policies. Depending on the problem, the state space can also be decomposed by imposing additional modeling assumptions. Decomposition of the state space into subspaces requires that each subspace contains all relevant information of the system's state to evaluate a related subproblem.

Whereas decomposition may help the evaluation of a policy, a one-step policy improvement algorithm is used to find a better policy. Usually, the improved policy is not well-structured, and does not allow the decomposition of the state space. Consequently, the improved policy can be evaluated by simulation only. A second improvement step is

not possible, as the relative values of the states under the improved policy cannot be determined.

In an MDP where the different dimensions of the state space are directly correlated, decomposition of the state space is not possible. For example, in the PPP the element x_1 at some day, strongly depends on the value of x_2 at the previous day: BPPs with a residual shelf life of 2 days that are not issued on a day, have the next day a residual shelf life of one day. For the PPP the dimensions r and $r+1$ of successive state vectors, $\mathbf{x}$, are too much correlated to allow the decomposition of the stock-state space into subspaces.

The vector $\mathbf{x}$ in the PPP can take a large number of values in each dimension. As the elements refer to a number of BPPs, aggregation is possible by counting them in batches of say 4. For an arbitrary problem, this way of aggregating states may not be possible, as it strongly depends on the interpretation of the elements in the vector. Another level of aggregation is the aggregation of multiple dimensions into one dimension, e.g. BPPs of 8 different blood groups are modeled as if their are from a single universal blood group. This way of aggregation leaves information out of the state space. This is only possible when this information is (expected to be) not relevant for deriving optimal decisions.

Conclusions – The formulation of a sequential decision problem as an MDP maybe very helpful in finding an approximately optimal solution, even when the problem seems to be too large to solve. Whether aggregation or decomposition can be applied to obtain a good approximation of an optimal MDP policy, depends strongly on the structure of the problem as well as on the data. Looking for problem structure or imposing additional structure helps in approximating a high-dimensional MDP. Whether the approximation is justified is to be checked by simulation. Simulation is often the only technique that is capable of evaluating the resulting policy at a detailed level.

Appendix A

Abbreviations, notations and conventions

.

Throughout this thesis the following conventions and notations apply. At a few places notations may differ, e.g. for typographical reasons or to stay consistent with notations that are common in the literature.

$\mathbf{x}$	vectors are printed in bold face small case letters.
$\mathbf{A}$	matrices are in bold capital letters.
A_{ij}, x_i	elements of matrices and vectors are printed as scalar variables in italic case, with indexes in subscript.
$x = \sum_i x_i$	the sum of all elements in vector $\mathbf{x}$ is denoted by the scalar x.
$\mathbf{1}$	a vector of elements equal to 1.
$\mathbf{e}_i$	the i-th unit vector: all elements are zero, but the i-th element equals 1.
$\mathcal{X}$	sets are printed using calligraphic letters.
$\lvert\mathcal{X}\rvert$	denotes the number of elements in set $\mathcal{X}$.
$\bigcup_{i=1}^{m} \mathcal{X}_i$	the union of the m sets $\mathcal{X}_1, \ldots, \mathcal{X}_m$. $= \{x \mid x \in \mathcal{X}_1 \vee x \in \mathcal{X}_2 \vee \ldots \vee x \in \mathcal{X}_m\}$
$\prod_{i=1}^{m} \mathcal{X}_i$	the multi-dimensional state space by combining the m sets $\mathcal{X}_1, \ldots, \mathcal{X}_m$. $= \{\mathbf{x} = (x_1, x_2, \ldots, x_m) \mid \forall i \in \{1, 2, \ldots, m\} : x_i \in \mathcal{X}_i\}$.
$\mathcal{X}(x_1 = a)$	denotes the subspace of $\mathcal{X}$, where x_1 takes value a.

$I(q)$ $= \begin{cases} 1 & \text{if } q = \text{true}, \\ 0 & \text{if } q = \text{false}. \end{cases}$
e.g. $I(a > 0)$ is 1 if $a > 0$ and 0 if $a \leq 0$

a^+ the positive part of a: a^+ is a when $a > 0$ and 0 otherwise.

$|a|$ the absolute value of a.

$\lfloor a \rfloor$ the floored value of a (round down).

$\lceil a \rceil$ the ceiled value of a (round up).

When the value of a is limited from above and below, the bounds are denoted by:

$\overline{a}$ or A the upper bound set to a: depending on the context, the upper bound of a parameter or variable is denoted in uppercase or the over-line-notation is used.

$\underline{a}$ the lower bound set to a.

Variables are denoted by a single letter to keep notations clear and short. To improve the readability of the models, the letter relates to the definition of the variable. Variable names thus depend on the context, but in case of conflicting or confusing notations other names are chosen.

Appendix B

Relative values in periodic MCs

The content of this section is a reformulation of a technical note by Van der Wal [147].

When a policy π is periodic, a state $\mathbf{x}$ of the underlying Markov chain belongs to one of the D periodic classes. The expected direct costs to incur in a state are stored in the vector $\mathbf{c}$ and the transition probabilities under policy π are stored in matrix $\mathbf{P}$.

The system visits the classes in the order $1, 2, \ldots, D, 1, 2$, etc. and the long run average costs to incur in a slot is class dependent: $g_1, g_2, \ldots, g_D$. Therefore the value vector in a SA algorithm can not be used directly as a relative value vector, as was argued already in Section 1.3.2. In this section, we show how the bias terms are computed for periodic policies. These bias terms are relative values that can be used in a PI algorithm or in a one-step policy improvement algorithm.

The bias terms $\widetilde{\mathbf{v}}^\pi = \mathbf{v}$ and the gain $g^\pi = g$ are the unique solution to the following set of equations:

$$\begin{aligned} \mathbf{v} &= \mathbf{c} + \mathbf{P}\mathbf{v} - \mathbf{g}, && \text{(B.1)} \\ \mathbf{P}\mathbf{g} &= \mathbf{g}, && \text{(B.2)} \\ \mathbf{P}^*\mathbf{v} &= \mathbf{0}. && \text{(B.3)} \end{aligned}$$

where $\mathbf{g} = g \cdot \mathbf{1}$ and

$$\mathbf{P}^* \equiv \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \mathbf{P}^i = \lim_{N\to\infty} \mathbf{P}^N \frac{1}{D} \sum_{d=0}^{D-1} \mathbf{P}^d. \qquad \text{(B.4)}$$

By iteratively substituting $\mathbf{v} = \mathbf{c}^\pi + \mathbf{P}^\pi\mathbf{v} - g\mathbf{1}$ in the righthand-side of Equation (B.1), one gets after n iterations:

$$\mathbf{v} = \sum_{i=0}^{n-1} \mathbf{P}^i(\mathbf{c} - \mathbf{g}) + \mathbf{P}^n\mathbf{v} = \sum_{i=0}^{n-1} \mathbf{P}^i\mathbf{c} - n \cdot \mathbf{g} + \mathbf{P}^n\mathbf{v}.$$

$\sum_{i=0}^{n-1} \mathbf{P}^i\mathbf{c}$ is the total expected costs over an horizon of n slots when strategy π is applied. In the evaluation of the MC by the SA algorithm, these costs are denoted by $\mathbf{V}_n^\pi$. Hence

$$\mathbf{v} = \mathbf{V}_n^\pi - n\mathbf{g} + \mathbf{P}^n\mathbf{v}. \tag{B.5}$$

Summing Eq. (B.5) over $n = N$ to $N + D - 1$, and next diving both sides by D yields:

$$\mathbf{v} = \frac{1}{D} \sum_{n=N}^{N+D-1} \mathbf{V}_n^\pi - \frac{N + N + D - 1}{2} \cdot \mathbf{g} + \frac{1}{D} \sum_{n=N}^{N+D-1} \mathbf{P}^n\mathbf{v}.$$

Taking the limit for N to ∞ results in:

$$\mathbf{v} = \lim_{N\to\infty} \left(\frac{1}{D} \sum_{n=N}^{N+D-1} \mathbf{V}_n^\pi - N\mathbf{g} \right) - \frac{D-1}{2}\mathbf{g}, \tag{B.6}$$

as $\lim_{N\to\infty} \sum_{n=N}^{N+D-1} \mathbf{P}^n\mathbf{v} = \mathbf{0}$, given Equations (B.3) and (B.4).

Since relative values are unique up to an additive constant, one may subtract any constant vector from it. For example, one may value all states against a fixed reference state, say state $\mathbf{z}$:

$$\lim_{N\to\infty} \frac{1}{D} \sum_{d=0}^{D-1} \left(\mathbf{V}_{N+d}^\pi - V_{N+d}^\pi(\mathbf{z}) \cdot \mathbf{1} \right). \tag{B.7}$$

Alternatively, the value vector can be set to the bias terms: the expected difference in receiving the total expected costs over an infinitely long horizon compared to receiving every slot the class dependent average costs:

$$\lim_{N\to\infty} \frac{1}{D} \sum_{d=0}^{D-1} \left(\mathbf{V}_{N+d}^\pi - (N + d) \cdot \cdot \mathbf{g} \right). \tag{B.8}$$

Appendix C

Simulation-based search for order-up-to levels

Finding an optimal order-up-to level S^* to an ordinary order-up-to S rule for a stationary problem with period 1 is easy when the cost function appears to be convex. Commonly used procedure are binary search and Fibonacci search. When the cost function is not convex but may have multiple local optima, then an optimal order-up-to level is obtained by complete enumeration over a range of order-up-to levels. If no explicit expression for the cost function in terms of S exist, simulation is used to evaluate a chosen value of S.

The optimization problem that we are studying is periodic with period 7 and an optimal order-up-to S rule consist of five parameters: $(S_1, S_2, \ldots, S_5)$; one for each of the five working days. Production volumes are set according to $\min(S_d - x, U_d)$, with x the total stock level and U_d is the production capacity, which is most studies assumed to be (virtually) infinite. Finding optimal values for $(S_1, S_2, \ldots, S_5)$ is much more difficult, since depending on the costs structure there are many local optima.

Finding nearly optimal parameter values for the 2D-rule seems to be even more complicated since it consists of ten parameters for problems with five working days a week. Despite these difficulties we present two fast incremental search algorithms based on simulation. We do not guarantee these algorithms to find the best optimal order-up-to levels, but they have shown to be successful to a number of cases that we have considered. In fact we have also observed cases where the incremental search, should be inverted, such that one starts at high order-up-to values and successively decrease the levels. Since these search algorithm are not the focus of this thesis, we did not studied the conditions under which the algorithm does execute well.

For the description of the algorithm we restrict ourselves to the PPP with production on Monday to Friday.

C.1 Ordinary order-up-to S rule

For the order-up-to S rule with one order-up-to level per working day, we present the 1D-local-search algorithm. First we introduce some notation.

(Notation) Let $\mathbf{e}_d$ denote the 5-dimensional vector with all components equal to 0 except for element d which is 1, e.g., $\mathbf{e}_2 = (0, 1, 0, 0, 0)$. Vector $\mathbf{1}$ is the 5-dimensional vector with all elements equal to 1: $\mathbf{1} = (1, 1, 1, 1, 1)$.

1D-local-search algorithm

- *Initialize.* Define $\mathbf{S} = (0, 0, 0, 0, 0)$
- *Step 1.* Replace $\mathbf{S}$ by the best improvement $\mathbf{S} + \mathbf{1}$ until no further improvement is found,
- *Step 2.* Replace $\mathbf{S}$ by the best improvement $\mathbf{S} + \mathbf{e}_d$ until no further improvement is found over the last 5 iterations,
- *Step 3.* Replace $\mathbf{S}$ by the best improvement $\mathbf{S} - \mathbf{e}_d$ until no further improvement is found over the last 5 iterations.

Starting in with an initial choice of $\mathbf{S}$ we apply an incremental search; to test for improvements the order-up-to S rule is evaluated through a few, say 10, short simulation runs. The production volumes follow in principle from the order-up-to levels, but production will not exceed the production capacity on some day. The search algorithm thus anticipates that in practice the production capacity may be restrictive.

In the 1D-local-search algorithm the result of Step 1 is a set of order-up-to levels identical for each of the five working days. Applying the order-up-to S rule with these order-up-to levels implies that the production volume on some day equals the demand on the previous day. The production level on Monday is the demand over the last three days. Due to the perishability and the periodicity some of the order-up-to levels should be higher and maybe some of them should be set lower. This is done through steps 2 and 3.

In each iteration of Step 2 the production on some day d is raised by 1, consequently the production on day $d + 1$ is lowered by 1. (When the production on Friday is raised by 1, the available stock on Monday may be 1 higher and thus the production volume on Monday is 1 lower.) In Step 3 the production on day d is lowered by 1, consequently the production on day $d + 1$ is raised by 1.

Initial S – Instead of taking the null-vector as an initial choice of **S** one could set **S** to the mean demand over the next days until the next production is released. S_1 is then the mean demand over Monday and Tuesday, S_5 is the sum of the mean demands over Friday, Saturday, Sunday and Monday. When the initial value of **S** is still very low, e.g. because Step 1 is skipped, we should be careful in setting the order in which increments are evaluated in Step 2 and 3. One should enforce fair tie-breaking when increments on different days are equally well.

Simulation – Each increment is evaluated through 10 simulation runs. When the current value of **S** is still far from optimal, the simulation runs can be very short to evaluate whether an increment of **S** is an improvement. When an increment does not result in an improvement over all 10 runs then the length of simulation run is increased to obtain a more accurate evaluation. The initial run length that we have chosen is 250 weeks, which is successively increased to 1,000, and 4,000 weeks. When 10 runs of 4,000 weeks do not result in an improvement Step 2 of the search algorithm is executed. Before an increment in Step 2 is selected one has to evaluate all 5 possible increments. Step 2 is terminated and Step 3 is started when none of the 5 increments result in an improvement.

Running time – The running time of the search procedure depends on the scale of the problem. For the down-scaled problem the search last about one minute. For the full size problem the search may take 15 minutes to half an hour, depending on the length of the simulation runs. When production and demand happens on a very large scale, the simulation-based search will take much longer than solving the MDP formulation after scaling the problem.

C.2 2D-order-up-to S rule

For the 2D-order-up-to S rule one has to optimize over ten different parameters $\left(\mathbf{S}^{\mathrm{T}}, \mathbf{S}^{\mathrm{Y}}\right)$: $\left(S_1^{\mathrm{T}}, S_2^{\mathrm{T}}, S_3^{\mathrm{T}}, S_4^{\mathrm{T}}, S_5^{\mathrm{T}}; S_1^{\mathrm{Y}}, S_2^{\mathrm{Y}}, S_3^{\mathrm{Y}}, S_4^{\mathrm{Y}}, S_5^{\mathrm{Y}}\right)$. Parameter S_1^{T} denotes the order-up-to level for the total stock, S_1^{Y} is the order-up-to level related to the 'young' products in stock. The optimization over ten parameters seems to be a complicated task, but the 1D-local-search algorithm of the previous section is helpful to find initial values. Again an incremental search, with increments and decrements of the order-up-to levels commonly results in good parameter values. Again, we do not guarantee the algorithm to converge to the best parameter set, since under different cost structures, it may end-up in a local optimum.

2D-local-search algorithm

- *Initialize.* First execute the 1D-local-search algorithm. Let $\mathbf{S}^*$ be the best policy found. Set $\mathbf{S}^{\mathrm{T}} := \mathbf{S}^*$ and $\mathbf{S}^{\mathrm{Y}} := \mathbf{S}^*$.
- *Step 1.* This overestimates the levels for 'young', thus replace $\mathbf{S}^{\mathrm{Y}}$ by the best improvement $\mathbf{S}^{\mathrm{Y}} - \mathbf{e}_d$ until no further improvement is found.
- *Step 2.* Replace $\mathbf{S}^{\mathrm{T}}$ by the best $\mathbf{S}^{\mathrm{T}} - \mathbf{e}_d$ for $d = 1, \ldots, 5$ until no further improvement is found.
- *Step 3.* Replace $\mathbf{S}^{\mathrm{Y}}$ by the best $\mathbf{S}^{\mathrm{Y}} + \mathbf{e}_d$ until no further improvement is found.

Starting with $\mathbf{S}^{\mathrm{T}}$ and $\mathbf{S}^{\mathrm{Y}}$ equal to the order-up-to levels for the ordinary order-up-to S rule, one successively decreases in Step 1 the order-up-to levels for young. Since under the 2D-rule the production volume in some states is determined by $S_d^{\mathrm{Y}} - x^{\mathrm{Y}}$, one may decrease S_d^{T}, as in Step 3. Further fine-tuning $\mathbf{S}^{\mathrm{Y}}$ may result in an increase of $\mathbf{S}^{\mathrm{Y}}$, since $\mathbf{S}^{\mathrm{T}}$ may be lowered in the previous step. Fine-tuning the 10 parameters is time-consuming since it requires relatively long simulation runs for an accurate evaluation.

Appendix D

Optimization of fixed cycle

D.1 Determining the minimal cycle length

Under FC the green periods are fixed to say $\alpha_1, \ldots, \alpha_C$, where C is the number of combinations. The switch-over times are fixed to γ slots. During the first β slots the lights show yellow and cars still leave the queue (whenever present). Departures happen at rate 1 per slot per queue (that has a green or yellow signal).

A bottleneck stream of a combination is a queue that brings the highest workload over all streams in that combination. Further let λ_c denote the arrival intensity at the bottleneck stream of combination c.

The length of a cycle D is fixed given the sum of the green periods and the switch-over times. The what-we-call Minimal-cycle-length algorithm provides the minimal cycle length D and minimal green times $\alpha_1, \ldots, \alpha_C$, under which FC is just stable.

Minimal-cycle-length algorithm

1. $D' \leftarrow C \cdot (1+\gamma)$; $\quad$ {*suppose a minimum green time of 1 slot*}
2. repeat
3. $\quad D \leftarrow D', \quad D' \leftarrow C \cdot \gamma$;
4. $\quad \forall c \in \{1, \ldots, C\}$:

$$\alpha_c \leftarrow 1 + (\lceil \lambda_c D - (1+\beta) \rceil)^+, \qquad \{\textit{minimal green time given } D\}$$

$$D' \leftarrow D' + \alpha_c;$$

5. until $D' = D$;
6. $\rightarrow$ minimum cycle length is D slots, with green times $\alpha = \alpha_1, \ldots, \alpha_C$.

D.2 Incremental search

Starting at the shortest cycle possible the algorithm, which we call the Optimal-fixed-cycle algorithm, increases the cycle length successively by 1 slot. Every time that a better cycle is found the best one found so far is update. Especially in the beginning of the search, an increase by one may result in a very bad or even instable cycle. Therefore we continue increasing the cycle length until all green times are increased proportionally to the workload the respective flows bring to the intersection. Hence after each improvement M successive increments are considered, with $M = \sum_c \lambda_c / \lambda_g$ and $g = \arg\min_c \lambda_c$, the 'less- busiest' combination. If an improvement stays out the search is terminated.

We formalize the algorithm for finding a nearly optimal FC in pseudo-code using the notations from the previous section. We introduce $EW(D, \alpha)$ being the expected overall average waiting time under FC with cycle length D and green times $\alpha = \alpha_1, \ldots, \alpha_C$. The function EW can be evaluated by simulation or by solving a Markov chain. The vector $\mathbf{e}_c$ is the c-th unit vector with all elements set to 0, except for a 1 at the c-th position; u, m, b and g are auxiliary variables.

Optimal-fixed-cycle algorithm

1. Set cycle length D and green times α by the Minimal-cycle-length algorithm, such that FC is just stable.
2. $D^* \leftarrow D, \quad \alpha^* \leftarrow \alpha, \quad n \leftarrow 0; \qquad \{n = \#\ \textit{iterations no improvement}\}$
3. $g \leftarrow \arg\min_c \lambda_c,$

   $M \leftarrow \sum_c \lambda_c/\lambda_g; \qquad \{M = \textit{maximal}\ \#\ \textit{iterations no improvement}\}$
4. repeat
5. $\quad w \leftarrow EW(D, \alpha), \quad n \leftarrow n + 1;$
6. $\quad \forall c \in \{1, \ldots, C\} : \qquad \{\textit{select green period}\}$

   $\quad$ if $EW(D + 1, \alpha + \mathbf{e}_c) \leq w$

   $\quad\quad$ then $\quad w \leftarrow EW(D + 1, \alpha + \mathbf{e}_c), \quad b \leftarrow c;$
7. $\quad D \leftarrow D + 1, \quad \alpha \leftarrow \alpha + \mathbf{e}_b; \qquad \{\textit{lengthen green period combination } b\}$
8. $\quad$ if $EW(D^*, \alpha^*) > EW(D, \alpha) \qquad \{\textit{an improvement}\}$

   $\quad\quad$ then $\quad D^* \leftarrow D, \quad \alpha^* \leftarrow \alpha, \quad n \leftarrow 0;$
9. until $n = M$;
10. $\rightarrow$ best cycle length is D^* slots, with green times $\alpha_1^*, \ldots, \alpha_C^*$.

Appendix E

Extrapolating tabulated functions

A tabulated function is a function whose values are known and tabulated for a finite number of points on a grid, rather than set by an analytical expression.

E.1 1-dimensional tabulated functions

Consider a tabulated function $V(x) : \mathbb{N} \to \mathbb{R}$, for which the function values are known for $x \in \{0, 1, \ldots, B\}$. Through quadratic extrapolation based on three of the known values one can estimate the function value $V(y)$ for $y \in \{B+1, B+2, \ldots\}$.

E.1.1 Indirect extrapolation through a quadratic function

One approach for quadratic extrapolation is to compute the coefficients of the quadratic function $f(x)$ through the three points $V(B)$, $V(B-1)$ and $V(B-2)$, and estimate $V(y)$ by $f(y)$. We call this the indirect approach, since it requires the computation of the coefficients a, b, and c of the function $f(x) = a \cdot y^2 + b \cdot y + c$. The computation of these coefficients follow from the next theorem:

Theorem E.1.1. *Given a function $f(x) = ay^2 + by + c$, with unknown coefficients a, b, and c, but known function values $f(B)$, $f(B-1)$, and $f(B-2)$, for some $B \in \mathbb{R}$, then a, b, and c are set by:*

$$a = [f(B) - 2f(B-1) + f(B-2)]/2,$$
$$b = [f(B) - f(B-2)]/2 - 2a(B-1), \text{ and}$$
$$c = f(B) - aB^2 - bB.$$

Proof. The coefficients follow from the following observations:

- Coefficient c follows directly from the definition of $f(B)$.
- Coefficient b follows from the first order derivative of the curve $f(x) = ax^2 + bx + c$ through the points $(B, f(B))$, $(B-1, f(B-1))$ and $(B-2, f(B-2))$: $f'(x) = 2ax + b$. It is easily shown that $f'(x) = [f(x+1) - f(x-1)]/2$ for any x. Hence $2ax + b = [f(x+1) - f(x-1)]/2$ and thus $b = [f(x+1) - f(x-1)]/2 - 2ax$. Once a is known b is computed by evaluating the expression for $x = B - 1$.
- Coefficient a follows from the second order derivative of $f(x)$: $f''(x) = 2a$. We claim $f''(x) = [f(x+1) - f(x)] - [f(x) - f(x-1)]$ for all x. (The proof follows from straight forward calculus.) Hence $a = [f(x+1) - 2f(x) + f(x-1)]/2$ for any x. Evaluating this expression for $x = B - 1$ gives the value of a.

□

Indirect extrapolation formula

The suggested extrapolation formula for $V(y)$ is thus:

$$V(y) \approx a \cdot y^2 + b \cdot y + c \tag{E.1}$$

with

$$\begin{aligned} a &= [V(B) - 2V(B-1) + V(B-2)]/2, \\ b &= [V(B) - V(B-2)]/2 - 2a(B-1), \text{ and} \\ c &= V(B) - aB^2 - bB. \end{aligned}$$

If V indeed appears to be a quadratic function than this approximation is exact.

E.1.2 Alternative: direct quadratic extrapolation

An alternative approximation of $V(y)$ extrapolates V using (directly) the known function values $V(B)$, $V(B-\Delta)$ and $V(B-2\Delta)$ (with $\Delta = y - B$), rather then first computing the coefficients of a (quadratic) function:

$$V(y) \approx 3 \cdot V(B) - 3 \cdot V(B-\Delta) + V(B-2\Delta) \tag{E.2}$$

This extrapolation works fine whenever $V(B)$, $V(B-\Delta)$ and $V(B-2\Delta)$ are known: thus when $B \leq y \leq \lfloor \frac{3B}{2} \rfloor$. (For $y \leq B$ the extrapolation is meaning less, since the function values are known for $x \in \{0, 1, \ldots, B\}$. For $y > \lfloor \frac{3B}{2} \rfloor$ one may apply a trick by recursively extrapolating function values.)

The proof that the approximation indeed corresponds to quadratic extrapolation comes from the proof of the following theorem.

Theorem E.1.2. *Given a quadratic function $f(x) = ax^2 + bx + c$ with $a, b, c \in \mathbb{R}$, then for any $y, B \in \mathbb{R}$ and $\Delta = y - B$ the following relation holds:*

$$f(y) = 3 \cdot f(B) - 3 \cdot f(B-\Delta) + f(B-2\Delta) \tag{E.3}$$

Proof. For $y = B$, and thus $\Delta = 0$, the theorem clearly holds. A formal proof requires straight forward calculus to show the following equality (in which the constant c has already been canceled):

$$a(B+\Delta)^2 + b(B+\Delta) = 3(aB^2+bB) - 3(a(B-\Delta)^2 + b(B-\Delta)) + a(B-2\Delta)^2 + b(B-2\Delta)$$

By simplifying the expressions and further cancelation of terms that are on both sides of the $=$ sign, one proofs the equality. □

Example

Suppose we know the function values of a tabulated function $V(x)$ for $x = 0, 1, \ldots, 5 (= B)$, as given in the table below, and we are interested in an estimate of $V(7)$.

x	0	1	2	3	4	5	6	7
$V(x)$	5	10	19	32	49	70	?	?

Suppose that we are not aware of nor interested in the exact relationship between x and $V(x)$, then

1. we can apply the direct approach:
$$V(7) \approx 3 \cdot V(5) - 3 \cdot V(5-2) + V(5-4) = 3 \cdot 70 - 3 \cdot 32 + 10 = 124.$$

2. we may apply the indirect approach and we thus first compute the value of the coefficients a, b, and c:
$$\begin{aligned} a &= [V(5) - 2V(5-1) + V(5-2)]/2 = [70 - 2 \cdot 49 + 32]/2 = 2, \\ b &= [V(5) - V(5-2)]/2 - 2a(5-1) = [70 - 32]/2 - 8 \cdot 2 = 3, \text{ and} \\ c &= V(5) - 5^2 a - 5b = 70 - 25 \cdot 2 - 5 \cdot 3 = 5. \end{aligned}$$

   Hence $V(y) \approx 2y^2 + 3y + 5$ and $V(7) \approx 2 \cdot 7^2 + 3 \cdot 7 + 5 = 124$. the same results in more computations.

3. we can compute $V(7)$ recursively by using the direct approach for Δ fixed to 1 (independent of B):
$$V(7) \approx 3 \cdot V(6) - 3 \cdot V(6-1) + V(6-2) = 3 \cdot V(6) - 3 \cdot 70 + 49,$$

   which requires the computation of $V(6)$:
$$V(6) \approx 3 \cdot V(5) - 3 \cdot V(5-1) + V(5-2) = 3 \cdot 70 - 3 \cdot 49 + 32 = 95.$$

Evaluation

Since all given points are on the curve of the quadratic function $f(x) = 2x^2 + 3x + 5$, the results of the three quadratic extrapolations do coincide. In general this will only be the case for the last two extrapolations that are based on the last three known function values, while the points used for the first extrapolation depends on the overflow Δ.

Although the direct approach needs the value of Δ it seems to be a very efficient approach, since it does not require the computation of the coefficients a, b, and c. Unless Δ needs to be computed anyway, it becomes worth to compute the coefficients once and store them in memory when many extrapolations have to be executed.

A great advantage of the direct approach is that it can easily be extended to extrapolate a multi-dimensional tabulated function.

E.2 n-dimensional tabulated functions

Consider a tabulated function $V(\mathbf{x}) = V(x_1, \ldots, x_n)$ for which the (real) values are known on the n-dimensional $(0, B)$-grid. Hence $V(\mathbf{x})$ is known when all elements of vectors $\mathbf{x}$ are in $\{0, 1, \ldots, B\}$.

Through quadratic extrapolation one can estimate the function value $V(\mathbf{y})$ when $\mathbf{y}$ shows 'overflow' in one or more dimensions: when one or more elements exceed B. The extrapolation formula is displayed in the following equation:

$$V(\mathbf{y}) \approx 3 \cdot V(\mathbf{z}) - 3 \cdot V(\mathbf{z} - \boldsymbol{\Delta}) + V(\mathbf{z} - 2\boldsymbol{\Delta}) \tag{E.4}$$

where $\mathbf{z} = (\min\{B, y_1\}, \ldots, \min\{B, y_F\})$ and $\boldsymbol{\Delta} = (y_1 - z_1, \ldots, y_F - z_F)$.

When the function increases quadratically in each direction at which overflow might happen, than the approximation is exact.

Example

The following points all lay on the hyper curve of the quadratic form $(2x_1 + 3x_2)^2 + 7$.

x_1	0	0	0	1	1	1	2	2	2	3	1	3
x_2	0	1	2	0	1	2	0	1	2	1	3	3
$V(x_1, x_2)$	7	16	43	11	32	71	23	56	107	?	?	?

Suppose we are not aware of the exact form the curve through the points and we wish to estimate the last 3 missing value in the table below:

$V(3, 1)$, $V(1, 3)$ and $V(3, 3)$ can be estimated using Equation (E.4):

- $V(3, 1) \approx 3 \cdot V(2, 1) - 3 \cdot V(1, 1) + V(0, 1) = 88$
- $V(1, 3) \approx 3 \cdot V(1, 2) - 3 \cdot V(1, 1) + V(1, 0) = 128$
- $V(3, 3) \approx 3 \cdot V(2, 2) - 3 \cdot V(1, 1) + V(0, 0) = 232$

The estimates indeed correspond to the values set by the quadratic form.

Bibliography

[1] Adan, I. J. B. F., Boxma, O. J., and Resing, J. A. C. Queueing models with multiple waiting lines. *Queueing Systems 37* (2001), 65–98.

[2] Adan, I. J. B. F., van Eenige, M. J. A., and Resing, J. A. C. Fitting discrete distributions on the first two moments. *Probability in the Engineering and Informational Sciences 9* (1999), 623–632.

[3] Akcelik, R. *Transportation Research Record*. No. 1457. TRB, National Research Council, Washington DC, 1994, ch. Estimation of Green Times and Cycle time for Vehicle-Actuated Signals, pp. 63–72.

[4] Archibald, B., and Silver, E. (s, S) policies under continuoous review and discrete compound poisson demands. *Management Science 24* (1978), 899–908.

[5] Arrow, K. J. The genesis of "optimal inventory policy".

[6] Arrow, K. J., Blackwell, D., and Girshick, M. A. Bayes and minimax solutions of sequential decision problems. *Econometrica 17* (1949), 213–244.

[7] Arrow, K. J., Harris, T., and Marschak, J. Optimal inventory theory. *Econometrica 19*, 3 (1951), 250–272.

[8] Axsäter, S. *Inventory Control*. International Series in Operations Research & Management Science. Springer, 2006.

[9] Bar-Lev, S. K., and Perry, D. A discrete time markovian inventory model for perishable commodities. *Stochastic Analysis and Applications 7*, 3 (1989), 243–259.

[10] Bell, C. Improved algorithms for inventory and replacement stock problems. *SIAM Journal on Applied Mathematics 18* (1970), 558–566.

[11] Bell, M. G. H. A probabilistic approach to the optimisation of traffic signal settings in discrete time. In *Proceedings of the 11-th International Symposium on Transportation and Traffic Theory* (1990), pp. 619–632.

[12] Bell, M. G. H. Future directions in traffic signal control. *Transportation Research Part A 26A*, 4 (1992), 303–313.

[13] Bell, M. G. H., and Brookes, D. Discrete time adaptive traffic signal control: the calculation of expected delays and stops. *Transportation Research Part C 1*, 1 (1993), 43–55.

[14] Bellman, R. *Dynamic Programming*. Princeton University Press, 1957.

[15] Bellman, R. *Applied Dynamic Programming*. Princeton University Press, 1962.

[16] Beun, M., and van Dijk, N. M. Personal communications and notes, 1996.

[17] Bhulai, S. Dynamic routing policies for multi-skill call centers. Technical Report WS2004-11, Vrije Universiteit Amsterdam, 2004.

[18] Bhulai, S. Dynamic routing policies for multi-skill call centers. *To appear in Probability in the Engineering and Informational Sciences* (2008).

[19] Blake, J. T., Thompson, S., Smith, S., Anderson, D., Arellano, R., and Bernard, D. Using dynamic programming to optimize the platelet supply chain in nova scotia. In *Proceedings of the 29th Meeting of the European Working Group on Operational Research Applied to Health Services* (2003), C. R. O. M. Dlouhý. Prague, Ed., pp. 47–65.

[20] Boxma, O. J., Levy, H., and Weststate, J. A. Efficient visit frequencies for polling tables: Minimization of waiting costs. *Queueing Systems: Theory and Applications 9* (1991), 133–162.

[21] Brown, G., Lu, J., and Wolfson, R. Dynamic modelling of inventory subject to obsolescence. *Management Science 11* (1964), 51–63.

[22] Bruin, J., and van der Wal, J. A dynamic control strategy for multi-item production systems. Unpublished paper, Eindhoven University of Technology, The Netherlands, 2008.

[23] Bruin, J., and van der Wal, J. A dynamic control strategy for multi-item production systems with backlog. Unpublished paper, Eindhoven University of Technology, The Netherlands, 2008.

[24] Bulinskaya, E. Some results concerning optimum inventory policies. *Theory of Probabibilty and Its Applications 9* (1964), 389–403.

[25] Camp, W. E. Determining the production order quantity. *Management Engineering 2* (1922), 17–18.

[26] Chazan, D., and Gal, S. A markovian model for a perishable product inventory. *Management Science 23*, 5 (1977), 512–521.

[27] Cohen, M. A. Analysis of single critical number ordering policies for perishable inventories. *Operations Research 24*, 4 (1976), 726–741.

[28] Cohen, M. A., and Pekelman, D. LIFO inventory systems. *Management Science 24*, 11 (1978), 1150–1162.

[29] Cohen, M. A., and Pierskalla, W. P. Perishable inventory theory and its application to blood bank management. Tech. rep., Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois, 1974.

[30] Cohen, M. A., and Pierskalla, W. P. Management policies for a regional blood bank. *Transfusion 15*, 1 (1975), 58–69.

[31] Cohen, M. A., and Pierskalla, W. P. Target inventory levels for a hospital blood bank or a decentralized regional blood banking system. *Transfusion 19*, 4 (1979), 444–454.

[32] Cowan, R. An improved model for signalised intersections with vehicle-actuated control. *Journal of Applied Probability 15*, 2 (1978), 384–396.

[33] Darroch, J. N. On the traffic light queue. *The Annals of Mathematical Statistics 35*, 1 (1964), 380–388.

[34] Darroch, J. N., Newell, G. F., and Morris, R. W. J. Queues for a vehicle-actuated traffic light. *Operations Research 12*, 6 (1964), 882–895.

[35] Deniz, B., Scheller-Wolf, A., and Karaesmen, I. Managing inventories of perishable goods: the effect of substitution. *revised and resubmitted to Management Science* (2004).

[36] Deuermeyer, B. *Inventory control Policies for Multi-Type Production Systems with applications to Blood Component Management.* unpublished Ph.D. dissertation, Northwestern University, 1976.

[37] Dion, F., and Yagar, S. Real-time control of signalised networks – different approaches for different needs. *Road Monotoring and Control (IEE)* (April 1996).

[38] Dreyfus, S. E. Computational aspects of dynamic programming. *Operations research 5*, 3 (1957), 409–415.

[39] Dunne, M. C., and Potts, R. B. Algorithm for traffic control. *Operations Research 12*, 6 (1964), 870–881.

[40] Dvoretzky, A., Kiefer, J., and Wolfowitz, J. The inventory problem: I. Case of known distributions of demand. *Econometrica 20*, 2 (1952), 187–222.

[41] Dvoretzky, A., Kiefer, J., and Wolfowitz, J. On the optimal character of the (s, S) policy in inventory theory. *Econometrica 21*, 4 (1953), 586–596.

[42] Emmons, H. *The optimal use of radioactive pharmaceuticals in medical diagnosis.* unpublished Ph.D. dissertation, John Hopkins University, Baltimore, 1968.

[43] Erlenkotter, D. Ford whitman harris and the economic order quantity model. *Operations Research 38*, 6 (1990), 937 – 946.

[44] Fairfield, R. P., and Kingsman, B. G. Control theory in production/inventory systems: a case study in a food processing organization. *The Journal of the Operational Research Society* (1993).

[45] Federgruen, A., Groenevelt, H., and Tijms, H. C. Coordinated replenishments in a multi-item inventory system with compound poisson demands. *Management Science 30*, 3 (1984), 344–357.

[46] Federgruen, A., and Zipkin, P. An efficient algorithm for computing optimal (s, S) policies. *Operations Research 34* (1984), 1268–1285.

[47] Fries, B. Optimal ordering policy for a perishable commodity with fixed lifetime. *Operations Research 1* (23), 46–61.

[48] Gartner, N. H. OPAC: A demand-responsive strategy for traffic signal control. Transportation Research Record 906, U.S. Dept. Transp., Washington, DC, 1983.

[49] Ghare, P. M., and Shrader, G. F. A model for exponentially decaying inventory. *Journal of Industrial Engineering 14* (1963), 238–243.

[50] Goh, C. H., Greenberg, B. S., and Matsuo, H. Two-stage perishable inventory models. *Management Science 39*, 5 (1993), 633–649.

[51] Goyal, S. K., and Giri, B. C. Recent trends in modeling of deteriorating inventory. *European Journal of Operational Research 134*, 1 (2001), 1–16.

[52] Hadley, G., and Whitin, T. M. An optimal final inventory model. *Management Science 7* (1961), 179–183.

[53] Hadley, G., and Whitin, T. M. Generalizations of the optimal final inventory model. *Management Science 8* (1962), 454–457.

[54] Haijema, R., and van der Wal, J. Dynamic Traffic Light Regulation: Phase I – Intersections in Isolation. working paper AE13/2005, Department of Quantitative Economics, University of Amsterdam, August 2005.

[55] Haijema, R., and van der Wal, J. An MDP decomposition approach for traffic control at isolated signalized intersections. *Probability in the Engineering and Informational Sciences 22*, 4 (2008), 587–602.

[56] Haijema, R., van der Wal, J., and van Dijk, N. M. Blood platelet production: a high-dimensional perishable inventory problem. In *Operations Research Proceedings 2004* (2005), H. Fleuren, D. den Hertog, and P. Kort, Eds., Springer, pp. 84–92.

[57] Haijema, R., van der Wal, J., and van Dijk, N. M. Blood platelet production: Optimization by dynamic programming and simulation. *Computers & Operations Research 34*, 3 (March 2007), 760–779. submitted in 2004, online publication in May 2005, doi:10.1016/j.cor.2005.03.023.

[58] Haijema, R., van Dijk, N. M., van der Wal, J., and Smit Sibinga, C. Blood platelet production with breaks: Optimization by SDP and Simulation. *International Journal of Production Economics* (2008). accepted for publication, online available through doi:10.1016/j.ijpe.2006.11026.

[59] Harris, F. W. How many parts to make at once. *Factory, The Magazine of Management 10* (1913), 135–136,152.

[60] Harris, F. W. How much stock to keep on hand. *Factory, The Magazine of Management 10* (1913), 240–241, 281–284.

[61] Haviv, M., and Ritov, Y. Externalities, tangible externalities, and queuing disciplines. *Management Science 44* (1998), 850–858.

[62] Hax, A. C., and Candea, D. *Production and Inventory Management.* Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

[63] Henry, J. J., Farges, J. L., and Tuffal, J. The PRODYN real time traffic algorithm. In *Proceedings of the 4-th IFAC-IFIP-IFORS Conference on Control in Transportation Systems* (1983), D. Klampt and R. Lauber, Eds., VDI/VDE: Dusseldorf, Germany, pp. 307–311.

[64] Hesse, S. M., Coullard, C., Daskin, M. S., and Hurter, A. P. A case study in platelet inventory management. In *Proceedings of the Sixth Annual Industrial Engineering Research Conference, Miami Beach, Florida* (1997).

[65] Higashikubo, M., Hinenoya, T., and Takeuchi, K. Traffic queue length measurement using an image processing sensor. *Sumitomo Electric Technical Review 43* (1997), 64–68.

[66] Hinloopen, B. Head blood processing, Sanquin blood bank The Netherlands division North-East. Personal communications, 2003-2004.

[67] Hofri, M., and Ross, K. W. On the optimal control of two queues with server set-up times and its analysis. *SIAM Journal of Computing 16* (1987), 399–420.

[68] Howard, R. A. *Dynamic Programming and Markov Processes.* MIT Press, Cambridge University, 1960.

[69] Howard, R. A. *Dynamic Probabilistic Systems – Volume II: Semi-Markov and Decision Processes.* John Wiley and Sons, 1971.

[70] Hunt, P. B., Robertson, D. L., and Bretherton, R. D. The SCOOT on-line traffic signal optimization technique. *Traffic Engineering and Control 23* (1982), 190–192.

[71] Jennings, J. B., and Kolesar, P. Comments on a blood-bank inventory model of Pegels and Jelmert. *Operations Research 21*, 3 (1973), 855–858.

[72] Johnson, E. On (s, S) policies. *Management Science 15* (1968), 80–101.

[73] Jones, R. L. The blood supply chain, from donor to patient: a call for greater understanding leading to more effective strategies for managing the blood supply. *Transfusion 43*, 2 (2003), 132–134.

[74] Katsaliaki, K., and Brailsford, S. C. Using simulation to improve the blood supply chain. *The Journal of the Operational Research Society 58* (2007), 219–227.

[75] Katz, A. J., Carter, C. W., Saxton, P., Blutt, J., and Kakaya, R. M. Simulation analysis of platelets production and inventory management. *Vox Sang 44* (1983), 31–36.

[76] Kopach, R., Balcioğlu, B., and Carter, M. Tutorial on constructing a red blood cell inventory management system with two demand rates. *European Journal of Operational research* (2006). doi:10.1016/j.ejor.2006.01.051.

[77] Kortbeek, N., van der Wal, J., van Dijk, N. M., Haijema, R., and de Kort, W. Blood bank production and issuing optimization: Strategies for younger platelets. *under review with International Journal of Production Economics* (submitted in 2008).

[78] Kraiselburd, S. The evolution of real newsboy contracts in the USA. working paper WP06-16, Instituto de Empresa, María de Molina 12 - 5 , Madrid, Spain, january 2006.

[79] Krishnan, K. R. Joining the right queue: a markov decision rule. In *Proc. of 26th IEEE Conference on Decision and Control, Los Angeles* (1987), IEEE, New York, pp. 1863–1868.

[80] Krishnan, K. R., and Ott, T. J. State-dependent routing for telephone traffic: theory and results. In *Proc. of 25th IEEE Conference on Decision and Control, Athens, Greece* (1986), IEEE, New York, pp. 2124–2128.

[81] Krishnan, K. R., and Ott, T. J. Joining the right queue: a markov decision rule. In *Proc. of 26th IEEE Conference on Decision and Control, Los Angeles* (1987), IEEE, New York, pp. 1863–1868.

[82] Ledman, R. E., and Groh, N. Platelet production planning to ensure availability while minimizing outdating. *Transfusion 24*, 6 (1984), 532–533.

[83] Lee, J., Lee, K., Seong, K., Kim, C., and Lee-Whang, H. Traffic control of intersection group based on fuzzy logic. In *Proceedings of the 6-th International Fuzzy Systems Association World Congress* (1995), pp. 465–468.

[84] Li, M. T., and Gan, A. C. Signal timing optimization for oversaturated networks using TRANSYT-7F. presented at the 78th Annu. Meeting Transportation Research Board, 1999.

[85] Lian, Z., and Liu, L. A discrete-time model for perishable inventory systems. *Annual of Operations Research 87* (1999), 103–116.

[86] Lowrie, P. The sydney coordinated adaptive control system – principles, methodology, algorithms. In *IEE Conference Publication* (1982), no. 207.

[87] Luk, J. Y. K. Two traffic-responsive area traffic control methods: SCAT and SCOOT. *Traffic Engineering and Control 25* (1984), 14–22.

[88] Mauro, V., and DiTaranto, C. UTOPIA. In *Proceedings of the 6-th IFAC-IFIP-IFORS Conference on Control, Computers and Communications in Transport* (1989), Parijs.

[89] McCullough, J., Undis, J., and Allen (Jr), J. W. Platelet production and inventory management. In *Platelet Physiology and Transfusion* (1978), D. M. Mallory, Ed., Washington, DC: American Association of Blood Banks,, pp. 17–38.

[90] McNeill, D. R. A solution to the fixed cycle traffic light problem for compound Poisson arrivals. *Journal of Applied Probability 5* (1968), 624–635.

[91] Mei, R. D. V. D. Polling systems with switch-over times under heavy load: Moments of the delay. *Queueing Systems: Theory and Applications 36*, 4 (2000), 381–404.

[92] Mercer, A., and Tao, X. Alternative inventory and distribution policies of a food manufacturer. *The Journal of the Operational Research Society 47*, 6 (1996), 755–765.

[93] Michalopoulos, P. G. Vehicle detection video through image processing: The AUTOSCOOP system. *IEEE Transactions on Vehicular Technology 40*, 1 (1991), 21–29.

[94] Miller, A. J. Settings for fixed-cycle traffic signals. *Operations Research 14*, 4 (December 1963), 373–386.

[95] Morse, P. M., and Kimball, G. E. *Methods of Operations Research.* MIT Press, Cambridge, MA, 1951.

[96] Murphy, S. Platelets from pooled buffy coats: an update. *Transfusion 45* (2005), 634–639.

[97] Nahmias, S. A comparison of alterrnative approximations for ordering perishable inventory. *INFOR 13*, 2 (1975), 175–184.

[98] Nahmias, S. Optimal ordering policies for perishable inventory - II. *Operations Research 23*, 4 (1975), 735–749.

[99] Nahmias, S. Higher order approximations for the perishable inventory problem. *Operations Research 25*, 4 (1977), 630–640.

[100] Nahmias, S. Perishable inventory theory: A review. *Operations Research 30* (1982), 680–708.

[101] Nahmias, S., and Pierskalla, W. P. Optimal ordering policies for product that perishes in two periods subject to stochastic demand. *Naval Research Logistics Quarterly 20* (1973), 207–229.

[102] Nandakumar, P., and Morton, T. E. Near myopic heuristic for the fixed-life perishableilty problem. *Management Science 39*, 12 (1993), 1490–1498.

[103] Newell, G. F. Approximation methods for queues with applications to the fixed-cycle traffic light. *SIAM Review 7*, 2 (1965), 223–240.

[104] Newell, G. F. The rolling horizon scheme of traffic control. *Transportation Research Part A 32* (1998), 39–44.

[105] Norman, J. M. *Heuristic procedures in dynamic programming.* Manchester university Press, Manchester, 1972.

[106] Ohno, K. Computational algorithm for a fixed-cycle traffic light problem for compound poisson arrivals. *Transportation Science 12* (1978), 29–47.

[107] Ott, T. J., and Krishnan., K. R. Separable routing: A scheme for state-dependent routing of circuit switched telephone traffic. *Annals of Operations Research 35* (1992), 43–68.

[108] Papageorgiou, M., Diakaki, C., Dinopoulou, V., Kotsialos, A., and Wang, Y. Review of road traffic control strategies. In *Proc. of the IEEE* (2003), vol. 91, IEEE, pp. 2043–2067.

[109] Park, B., Messer, C. J., and II, T. U. *Transportation Research Record.* No. 1683. TRB, National Research Council, Washington DC, 1999, ch. Traffic Signal Optimization Program for Oversaturated Conditions: A Genetic Algorithm Approach, pp. 133–142.

[110] Parlar, M. Optimal ordering policies for a perishable and substituatble products: a Markov decision model. *INFOR 23*, 2 (1985), 182–195.

[111] Pegels, C. C., and Jelmert, A. E. An evaluation of blood inventory policies: a markov chain application. *Operations Research 18*, 6 (1970), 1087–1098.

[112] Perry, D., and Posner, M. J. M. An $(S-1, S)$ inventory system with fixed shelf life and constant lead time. *Operations Research 46* (1998), S65–S71.

[113] Pierskalla, W. P. Optimal issuing policies in inventory management — I. *Management Science 13*, 5 (1967), 395–412.

[114] Pierskalla, W. P., and Roach, C. D. Optimal issuing policies for perishable inventory. *Management Science 18* (1972), 603–614.

[115] Pinson, S. D., Pierskalla, W. P., and Schaefer, B. A computer simulation anaysis of blood bank inventory policies. Technical report, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois, 1972.

[116] Porche, I., and Lafortune, S. Adaptive look-ahead optimization of traffic signals. *Journal of Intelligent Transportation Systems 4*, 3&4 (1999), 209–254.

[117] Powell, W. B. *Approximate Dynamic Programming: Solving the Curses of Dimensionality.* Wiley Series in Probability and Statistics. Wiley, 2007.

[118] Prastacos, G. P. Blood inventory management: an overview of theory and practice. *Management Science 30*, 7 (1984), 777–800.

[119] Prinsen, L. H. A. Personal communications with Luc Prinsen. Goudappel-Coffeng, September 2006.

[120] Puterman, M. L. *Markov decision processes: Discrete Stochastic Dynamic Programming.* Wiley Series in Probability and Mathematical Statistics, 1994.

[121] Reiman, M. I., and Wein, L. M. Dynamic scheduling of a two-class queue with set-ups. Technical report, Sloan School of Management, MIT, Cambridge (MA), 1994.

[122] Robertson, D. I. TRANSYT method for area traffic control. *Traffic Engineering and Control 10* (1969), 276–281.

[123] Sasieni, M. Dynamic programming and inventory problems. *Operations Research 11*, 1/2 (1960), 41–49.

[124] Sassen, S. A. E., Tijms, H. C., and Nobel, R. D. A heuristic rule for routing customers to parallel servers. *Statistica Neerlandica 51*, 1 (1997), 107–121.

[125] Scarf, H. The optimality of (S, s) policies in the dynamic inventory problem. In *Proceedings of the First Stanford Symposium on Mathematical Methods in the Social Sciences.* (1960), K. Arrrow, P. Suppes, and S. Karlin, Eds., Stanford University Press, Standord.

[126] Schmidt, C. P., and Nahmias, S. $(S-1, S)$ policies for perishable inventory. *Management Science 31* (1985), 719–728.

[127] Sen, S., and Head, L. Controlled optimization of phases at an intersection. *Transportation Science 31* (1997), 5–17.

[128] Shelby, S. G. Single intersection evaluation of real-time adaptive traffic control algorithms. In *Proceedings of the 84rd Annual Meeting of The Transportation Research Board (TRB'04)* (January 2004).

[129] Silver, E. A. A control system for coordinated inventory replenishment. *International Journal of Production Research 12* (1974), 647–670.

[130] Silver, E. A., Pyke, D. F., and Peterson, R. *Inventory Management and Production Planning and Scheduling.* Wiley, 1998.

[131] Sipper, D., and Bulfin, R. L. *Production: Planning, Control and Integration.* Industrial Engineering and Management Science. McGraw-Hill, 1998.

[132] Sirelson, V., and Brodheim, E. A computer planning model for blood platelet production and distribution. *Comput. Methods Programs Biomed. 35* (1991), 279–291.

[133] Smit Sibinga, C. T. Director of sanquin consulting services and advisor of the WHO Collaborating Center for Blood Transfusion and the WFH International Hemophilia Training Centre. Personal communications, 2003-2004.

[134] Smith, D. K. Dynamic programming and inventory management: What has been learnt in the last generation? In *Proceedings of the 1999 ISIR Workshop on Inventory Management. (Exeter, 2000)* (2000).

[135] Taale, H. Comparing methods to optimise vehicle actuated signal control. In *Proceedings of the 11-th International Conference on Road Transport Information and Control* (2002), Institute of Electrical Engineers, pp. 114–119.

[136] Takagi, H. Queueing analysis of polling models. *ACM Computing Surveys 20*, 1 (1988), 5–26.

[137] Takagi, H. *Stochastic Analysis of Computer and Communication Systems*. North-Holland, Amsterdam, 1990, ch. Queueing analysis of polling models: an update, pp. 267–318.

[138] Takagi, H. *Frontiers in Queueing: Models, Methods and Problems*. CRC Press, Boca Raton, 1997, ch. Queueing analysis of polling models: progress 1990-1994, pp. 119–146.

[139] Takagi, H. *Performance Evaluation: Origins and Directions*, vol. 1769 of *Lecture Notes in Computer Science*. Springer, Berlin, 2000, ch. Analysis and application of polling models, pp. 423–442.

[140] Tan, K. K., Khalid, M., and Yusof, R. Intelligent traffic lights control by fuzzy logic. *Malaysian Journal of Computer Science 9*, 2 (1995).

[141] Tekin, E., G urler, U., and Berk, E. Age-based vs. stock level control policies for a perishable inventory system. *European Journal of Operational Research 134* (2001), 309–329.

[142] Theodorović, D., Varadarajan, V., Popović, J., Chinnaswamy, M. R., and Ramaraj, S. Dynamic programming – neural network real time traffic adaptive signal control algorithm. *Annals of Operations Research 143* (2006), 123–131.

[143] Thorpe, T. Vehicle traffic light control using SARSA. Master's thesis, Department of Computer Science, Colorado State University, 1997.

[144] Tijms, H. C. *A first course in stochastic models.* Wiley, 2003.

[145] van den Broek, M. S. *Traffic Signals: Optimizing and analyzing traffic control systems.* Master thesis, Eindhoven University of Technonolgy, 2004.

[146] van den Broek, M. S., van Leeuwaarden, J. S. H., Adan, I. J. B. F., and Boxma, O. J. Bounds and approximations for the fixed cycle traffic light queue. *Transportation Science 40*, 4 (2006), 484–496.

[147] van der Wal, J. Relative values of states in periodic markov chains. Technical note, Dep. of Quantitative Economics, Univ. of Amsterdam, The Netherlands, 2005.

[148] van der Wal, J., and Schweitzer, P. J. Iterative bounds on the equilibrium distribution of a finite markov chain. *Probability in the Engineering and Informational Sciences 1* (1987), 117–131.

[149] van Dijk, N. M., Haijema, R., van der Wal, J., and Smit Sibinga, C. Blood platelet production: a formal approach for practical optimization. *Transfusion* (January 2009).

[150] van Donselaar, K., van Woensel, T., Broekmeulen, R., and Fransoo, J. Inventory control of perishables in supermarkets. *International Journal of Production Economics 104* (2006), 462–472.

[151] van Katwijk, R. T., de Schutter, B., and Hellendoorn, J. Traffic adaptive control of a single intersection: A taxonomy of approaches. Technical report 06-013, Delft University of Technology, 2006.

[152] van Katwijk, R. T., de Schutter, B., and Hellendoorn, J. Look-ahead traffic-adaptive signal control. Technical report 07-020, Delft University of Technology, 2007.

[153] van Leeuwaarden, J. S. H. Delay analysis for the fixed cycle traffic light queue. *Transportation Science 40*, 2 (2006), 189–199.

[154] van Zyl, G. J. J. *Inventory Control for Perishable Commodities.* unpublished Ph.D. dissertation, University of North Carolina, 1964.

[155] Veihola, M., Aroviita, P., Linna, M., Sintonen, H., and Kekomki, R. Blood donors and blood collection: variation of platelet production and discard rates in 17 blood centers representing 10 european countries from 2000-2002. *Transfusion 46* (2006), 991–995.

[156] Veinott, A., and Wagner, H. Computing optimal (s, S) inventory policies. *Management Science 11* (1965), 525–552.

[157] Verhoeven, G. Logistiek, een nieuwe tak van sport binnen de bloedbank. *AENORM 11*, 44 (2004), 16–21.

[158] Verhoeven, G. Automatisch voorraadbeheer: bloedproducten als pakjes koekjes. *Bloedbeeld, a magazine of Stichting Sanquin Bloedvoorziening*, 4 (2006), 17–18.

[159] Verhoeven, G., and Smid, M. Logistiek, een nieuwe tak van sport binnen de bloedbank. *NVB Bulletin, Nederlandse Vereniging voor Bloedtransfusie* (december 2003), 21–23.

[160] Website. ITS, University of Leeds, http://www.its.leeds.ac.uk/projects/smartest (Last visited at 2008-06-25), 1999.

[161] Website. http://www.didyouknow.cd/trafficlights.htm (Last visited at 2007-07-19), 2007.

[162] Webster, F. V. Traffic signal settings. Road Research Technical Paper 39, Road Research Laboratory, London, U.K., 1958.

[163] Whitaker, B. I., and Sullivan, M. The 2005 nationwide blood collection and utilization survey report. Tech. rep., US Department of Health and Human Services, and the American Association of Blood banks (AABB), 2005.

[164] Wiering, M., van Veenen, J., Vreeken, J., and Koopman, A. Intelligent traffic light control. technical report UU-CS-2004-029, Institute of information and computing sciences, Utrecht University, 2004.

[165] Wijngaard, J. Decomposition for dynamic programming in production and inventory control. *Engineering and Process Economics 4* (1979), 385–388.

[166] Wilson, R. H. A scientififc routine for stock control. *Havard Business Review 13*, 1 (1934), 116–128.

[167] Winands, E. M. M. On polling systems with large setups. *Operations Research Letters 35* (2007), 584–590.

[168] Yu, X.-H., and Recker, W. W. Stochastic adaptive control model for traffic signal systems. *Transportation Research Part C 14*, 4 (2006), 263–282.

[169] Zheng, Y.-S., and Federgruen, A. Finding optimal (s, S) policies is about as simple as evaluating a single policy. *Operations Research 39*, 4 (1991), 654–665.

[170] Zheng, Y.-S., and Federgruen, A. Errata: Finding optimal (s, S) policies is about as simple as evaluating a single policy. *Operations Research 40*, 1 (1992), 192.

[171] Zhou, D., and Pierskalla, W. P. Perishable product inventory policies with emergency replenishments. Tech. rep., Anderson School of Management, UCLA, 2002.

Summary in English

Introduction

Quite a number of optimization problems arising in practice involve sequential decision making and can be formulated as a Markov decision problem (MDP). However, the size of an MDP for many problems that arise in practice is often too large to determine an optimal policy. Fortunately, the problem's state space has some additional structure.

In this thesis we illustrate how to exploit the problem structure for two different applications in order to find good approximate solutions. The first problem that we treat is the production-inventory management of blood platelet pools (BPPs), a typical example of a perishable product with a shelf life of only 4-7 periods. The second problem is the dynamic control of traffic lights. The motivation to study these two problems stems from the fact that both problems are high-dimensional MDPs. The structure of the state space is different, consequently we apply different approaches to the two problems.

Part I: Chapters 2 to 4 – Inventory management of BPPs

Blood platelets are of life saving importance: as small particles in the blood stream they help repair damaged blood vessels and prevent excessive bleeding. Some patients need to be transfused with platelets to keep the concentration of platelets in their blood stream at a safe level. To meet the uncertain demand at hospitals blood banks produce BPPs, also called thrombocytes concentrates, by pooling platelets from 5 different voluntary donors into a storage bag: the platelet pool. A platelet pool is the most perishable and the most expensive blood product processed at blood banks. BPPs are processed from a side product that is in The Netherlands fortunately plenty available after processing red blood cell concentrates (RBCs). Dutch blood banks usually process BPPs only of the blood groups O and A, since for platelet transfusion the blood groups of most patients are to some degree compatible with other blood groups.

Despite the compatibility of blood groups Veihola et al. [155] report in 2006 European figures for 17 blood centers (spread over 10 European countries) that the waste of BPPs due

to outdating is considerable: 7-25% of the BPPs is not transfused because of outdating. In the USA [163] the fraction of outdated BPPs was 18% over 2004 . Outdating is the result of the short shelf life in combination with the uncertain demand and 'sub-optimal' replenishment strategies for inventories held at hospitals and/or blood banks. In The Netherlands the inventory of BPPs is mostly centralized at four blood banks. Shortages due to out-of-stock are to be kept at a low level, but can be resolved by meeting demand with BPPs from another blood bank. We have investigated (nearly) optimal production orders for one of the Dutch blood banks, where production happens 5 days a week (Monday to Friday), and production and laboratory tests take 1 day.

When modeling the problem as an MDP we firstly aggregate all blood groups into a single, say universal blood group. Later this modeling assumption is checked by a simulation study in which we distinguish patients and donors of eight different blood groups according to the ABO-system and the Rhesus-D factor (negative or positive). Depending on regulations by law and clinical studies BPPs have a maximal shelf life of 4 to 7 days, consequently one distinguishes BPPs of 4, 5, 6 or 7 age categories. The state vector of the MDP contains next to the day of the week also the number of BPPs in each age category.

On the one hand the high-dimensionality of the state vector does not allow an analytical approach, due to the complexity of modeling the transitions. On the other hand a numerical approach seems to be doomed to fail due to the large number of states that may occur. To allow numerical approximation of an optimal strategy by stochastic dynamic programming (SDP) we scale the problem by aggregating demand as if it happens in multiples of k pools. When restricting production orders to be also in multiples of k pools, the state space is at a more coarse grid. Consequently, we have to repair the transition law by transforming the demand distributions.

After truncating the state space to reduce the potential number of states even more, an optimal ordering strategy is computed for a given issuing policy. The optimal ordering strategy may be too complicated for practical use by blood bank managers. Therefore we check by simulation whether the optimal strategy resembles a simple rule that performs nearly optimal and that is applicable to the unscaled problem.

For data obtained for one of the Dutch blood banks it appears that a simple ordering rule with fixed order-up-to levels for each working day performs very well: outdating can be reduced to only a few percent or even less than 1 percent depending on the issuing discipline. Shortages occur even less often and the age of the pools upon transfusion may be improved as well. These conclusions are supported by an extensive sensitivity study using different data sets. Outdating is expected to be higher when the problem plays at

a much smaller scale, when demand is much more uncertain or when all donor blood is needed to produce BPPs. The combination of SDP and Simulation may still be helpful in finding an approximate solution, as we have shown how more advanced order-up-to rules can be read from frequency tables obtained by Simulation.

Next to a description of the methodology, Chapters 3 and 4 of the thesis offer a detailed validation through a sensitivity analysis using data from a Dutch blood bank. Since production breaks have a great impact on the outdating of BPPs, we also present an SDP-Simulation approach to solve the problem around short holiday breaks, during which production is hampered not only on Saturday and Sunday. Also the ordering of BPPs at a smaller blood bank or a hospital is considered, where set-up costs for ordering platelets may apply. The suggested SDP-Simulation approach can also be adopted to determine ordering quantities at other blood banks and to solve similar problems in other industries that deal with perishable products that have a short limiting shelf life. The software developed during this thesis project is made more user-friendly and named TIMO (for Thrombocytes Inventory Management Optimizer). TIMO currently runs at Sanquin's blood bank division South East.

Part II: Chapters 5 to 8 – Dynamic control of traffic lights

Traffic lights are introduced to make road traffic safer and more efficient by controlling the waiting time of car drivers. The problem of minimizing the long-run average waiting time over all F traffic flows at a single intersection can be formulated as a Markov decision problem with state description, next to the state of the lights, the number of cars present at each queue: $\mathbf{q} = (q_1, q_2, \ldots, q_F)$.

All cases that we consider are under-saturated cases: in the long-run all traffic can be handled by the intersection. The potential number of states grows exponentially in the number of queues (F). Solving the MDP requires approximate solutions when the intersection consists of a larger number of queues. When the problem of a single intersection requires an approximate solution, then one certainly has to rely on approximations for the control of a network of intersections.

Whereas in the previous application we were looking for problem structure under an optimal policy as computed after scaling, we now add structure to the problem by imposing a well-structured policy: the fixed cycle control (FC). Under FC queues get green in a fixed order and the effective green times are fixed. The state of the traffic lights is thus simply the position in the cycle or the time slot. By adjusting the position in the cycle the green periods of the flows are extended or ended dynamically based on the number

of cars queued. Therefore we need to know the relative appreciation or relative value of each time slot and for each possible $(q_1, q_2, \ldots, q_F)$. The state space under FC can be decomposed, such that one can evaluate each traffic flow in isolation of the others. The relative values of the states occurring under FC can be computed by summing the related relative values of the states for each flow. By a one-step policy improvement algorithm we obtain an improved strategy that hopefully performs nearly optimal. The improved strategy is called the RV rule (with RV for relative value).

The RV rule is put to the test and compared against basic control policies such as FC and exhaustive control. Therefore a simulation study is executed with different intersections under varying circumstances. The results are promising: in all (fully) symmetric cases considered the long-run average waiting time over all flows is reduced by at least 20 percent and in some cases even more than 70 percent compared to FC and (pure) exhaustive control (see Chapter 6).

Modern technology allows not only to track the number of cars at each queue, but it also detects the position of the cars approaching the intersection. Estimates of the expected arrival times of these cars can be included in the state description. The state space of each flow consists now of the state of the traffic light, the number of cars waiting at it's queue and a vector containing the arrival information. Assuming in the decision model that cars proceed at a constant and identical speed and that they do not change lanes, the RV rule can be easily extended to include arrival information. Some test by simulation show that including arrival information over only a few slots, say the next 5 slots, shows already an improvement of the control rule (see Chapter 7). Even when car speeds differ in the simulation model, the RV rule gives good results.

To solve the problem for a network of intersections (see Chapter 8), we decompose the control of the network by controlling the intersections in isolation. Starting with a good FC for each intersection in isolation, decisions are taken locally based on the RV rule. The synchronization of the local decisions at the intersections may be partly accomplished by using a few slots of arrival information. A more rigid synchronization can be achieved under FC by setting so-called offsets: in some to-FC-ideal cases green waves may be set in multiple directions (to maximize progression). An optimal synchronized FC is in general hard to find, but may result in low long-run average waiting times over all flows. For the RV rule we need just a locally-good FC, not necessarily an optimal synchronized one.

In a detailed simulation model in which cars may drive at constant or different speeds and queued cars take a length of 7 meters each, we put the resulting RV policies with arrival information to the test for several network cases. Although, the average waiting

times under synchronized FC can be very low when green waves apply, the RV rule with arrival information yields lower overall long-run average waiting times in all cases that we have studied. In the network cases with nine intersections that we have considered, the waiting time under the synchronized FC is still more than 40% above that of the RV rule with arrival information.

Adding structure to the problem by starting with a good FC is thus very beneficial in order to get to an improved approximate solution.

Conclusion

In the thesis we have studied two applications that may look so different, but that have in common that the underlying MDP is too large to solve. For both applications an approximate decision model is created and tested by (discrete) Simulation.

For the platelet production-inventory problem we have exploited the problem structure as present in the Dutch case: first blood groups are aggregated, next the pools are aggregated into batches to down-scale the problem and finally it appears that the age-categories of the pools can be aggregated to derive a simple rule for practical use. After re-scaling, this simple rule can be used in practice. For the dynamic control of traffic lights we suggest a one-step policy improvement algorithm. This algorithm can be applied only to a well-structured policy for which the relative values of the states can be computed. Under FC control the state space is well-structured and can be decomposed such that relative values can be computed for each flow in isolation. The control of a network of intersection also relies on decomposition.

Both decision models arose by exploiting the problem structure and rely on numerical techniques. For the platelet production-inventory problem we have looked for structure in the problem and data under consideration. For the dynamic control of traffic lights we have structured the problem by imposing a well-structured strategy for further improvement. For both applications simulation models are developed that contain more realistic features than can be incorporated in the decision models. Results show that the decision models perform very well, at least for the cases that are studied. Outdating of platelet pools can be reduced significantly and the waiting time at signalized intersections can be much lower than under the selected benchmark heuristics.

MDPs can thus be approximated successfully when exploiting the structure of the problem to come to approximate solutions. A uniform way of tackling high-dimensional MDPs may not exist, hence each problem requires a problem specific approach. This thesis helps in gaining insight in how to do so.

Nederlandse samenvatting

In dit proefschrift staan grote gestructureerde Markov beslisproblemen (MBP) centraal. Met 'groot' wordt bedoeld dat het aantal toestanden in het MBP model te groot is om in een acceptabele rekentijd een optimale oplossing te bepalen. Met 'gestructureerd' wordt bedoeld dat de toestandsruimte van een MBP een probleem-specifieke structuur kan hebben die benut kan worden in het oplosproces.

Het grote aantal toestanden is vaak het gevolg van de dimensionaliteit van de toestandsruimte. Voor iedere toestand is er een optimaal besluit, of optimale actie, te bepalen. Met behulp van (stochastisch) dynamisch programmeren kunnen de nodige berekeningen efficient plaatsvinden. Voor veel vraagstukken uit de praktijk, is het aantal toestanden echter dermate groot dat men zich tevreden heeft te stellen met een benadering van de optimale oplossing. Daartoe dient men de aanwezige probleemstructuur te benutten.

Voor een tweetal optimalisatievraagstukken uit de praktijk wordt geïllustreerd hoe de kennis en oplostechnieken van MBPs ingezet kunnen worden om tot benaderingen van een optimale oplossing te komen. Het eerste vraagstuk betreft het productie-voorraadbeheer van bloedplaatjespools (BPPs). Het tweede vraagstuk betreft de drukte-afhankelijke regeling van verkeerslichten. De twee vraagstukken vergen een verschillende aanpak om tot een benadering te komen daar ze verschillen in probleemstructuur.

Deel I – Voorraadbeheer van bloedplaatjespools (H2 tot H4)

Bloedbanken en ziekenhuizen houden bloedproducten op voorraad om aan de slecht voorspelbare vraag naar transfusies te kunnen voldoen. Hoge voorraadniveaus garanderen een hoge beschikbaarheid, echter leiden vaak in de praktijk tot hoge vervalpercentages. De duurste en tevens kortst-houdbare bloedproducten zijn trombocytenconcentraten. De meeste trombocytenconcentraten worden verkregen door de plaatjes van vijf volbloeddonaties samen te voegen tot een bloedplaatjespool (BPP).

Zowel in Europa (Veihola e.a. [155]) als in de Verenigde Staten (Whitaker and Sullivan [163]) wordt gemiddeld 15 à 20% van de geproduceerde BPPs weggegooid, voornamelijk

als gevolg van het verlopen van de uiterste houdbaarheidsdatum van de BPPs. Door voorraadniveaus verstandiger vast te stellen, kan het verval gereduceerd worden zonder dat dit ten koste gaat van de beschikbaarheid van BPPs.

In dit proefschrift wordt onderzocht hoe een optimale productie- of bestelregel er uitziet, en in welke mate een makkelijke regel gehanteerd kan worden in de praktijk. Daartoe wordt een MBP model opgesteld, waarin het onderscheid naar bloedgroepen in eerste instantie achterwege gelaten wordt. Het uitgiftebeleid ten aanzien van BPPs wordt middels eenvoudige regels (FIFO, LIFO, of mengvormen) gemodelleerd. In 2004 kenden BPPs in Nederland een houdbaarheid van maximaal vijf dagen. Het aantal BPPs op voorraad dient daarom opgesplitst te worden in vijf leeftijdscategorieën. De voorraad wordt in een MBP model aldus middels een vector aangeduid: $\mathbf{x} = (x_1, \ldots, x_5)$, met x_r is het aantal BPPs met een resterende houdbaarheid van r dagen. Als er iedere dag hoogstens 9 BPPs geproduceerd worden, dan zijn er reeds 10^5 mogelijke toestanden ten aanzien van de voorraad te beschouwen. Daarnaast maakt de dag van de week (Maandag t/m Zondag) onderdeel uit van de toestandsruimte. Huidige ontwikkelingen staan het toe om BPPs 7 dagen op voorraad te houden, het aantal voorraadtoestanden is dan 10^7.

De productievolumes liggen in de praktijk van de Nederlandse bloedbanken aanzienlijk hoger dan 9, waardoor het aantal toestanden al snel te groot wordt om nog een optimale oplossing te bepalen. De toestandsruimte van het MBP kan nog enigzins gereduceerd worden doordat er in weekeinden niet geproduceerd wordt en door toestanden met een extreem voorraadniveau uit te sluiten in de berekeningen. Zelfs dan, is veelal het aantal toestanden te groot om een optimale oplossing te berekenen.

De structuur van de toestandsruimte staat het toe om toestanden te aggregeren. Door de vraag en de toegestane productievolumes te modelleren in batches van, zeg, 4 pools, kan het aantal toestanden gereduceerd worden met een factor 4^m, waarbij m de maximale houdbaarheid is, gemeten in dagen. Als de maximale houdbaarheid 5 dagen is ($m = 5$), wordt de toestandsruimte gereduceerd met ruwweg een factor 1000. Schaling van het MBP vergt echter het zorgvuldig aanpassen van de toestandsovergangen en met name van de overgangskansen.

Eventueel na schaling, kan er een optimale strategie berekend worden. De optimale strategie kan vrij complex zijn daar deze afhankelijk is van de leeftijd van de op voorraad liggende BPPs. Op basis van de optimale bestelstrategie, wordt indien mogelijk een eenvoudige regel afgeleid. Daartoe wordt gekeken welke toestanden vaak voorkomen onder de optimale strategie en welke bestelvolumes daarbij horen. Met dit doel wordt een frequentietabel opgesteld door simulatie van de optimale strategie.

Hieruit blijkt dat voor data van één van de Nederlandse bloedbanken, een aanvulregel met vaste aanvulniveaus voor iedere werkdag goed aansluit bij de optimale strategie. De prestatie van de aanvulregel kan geevalueerd worden door numerieke evaluatie van de onderliggende Markov ketens of middels simulatie. Het blijkt dat voor de gegeven dataset de regel inderdaad vrijwel optimaal presteert. Voor praktisch gebruik worden de aanvulniveaus van deze regel teruggeschaald, zodat voor het ongeschaalde probleem optimale productievolumes benaderd kunnen worden.

De aanvulregel is vervolgens toegepast in een simulatiemodel waarin patiënten en donoren van verschillende bloedgroepen onderscheiden worden. Hieruit blijkt het gerechtvaardigd te zijn om het onderscheid van bloedgroepen in het MBP model achterwege te laten. De bloedgroepen zijn immers in ruime mate uitwisselbaar en in de Nederlandse situatie blijven er voldoende bloedplaatjes over na het verwerken van de volbloeddonaties tot rode bloedcelconcentraten (RBCs).

Vele gevoeligheidsanalyses zijn verricht en alternatieve bestelregels zijn afgeleid met het oog op situaties bij andere bloedbanken en ziekenhuizen, alwaar:

- een ander uitgifte beleid gehanteerd kan worden,
- de vraag opgesplitst kan worden in een categorie voor patiënten die de jongste BPPs op voorraad krijgen en een categorie die de oudste plaatjes krijgen,
- een kortere of langere maximale houdbaarheid van BPPs van toepassing kan zijn,
- een andere afweging gemaakt kan worden tussen verval en beschikbaarheid,
- de vraag meer of minder onzeker kan zijn,
- de problematiek zich op een kleinere schaal kan afspelen, of
- er vaste kosten gemoeid kunnen gaan met het opstarten van een productierun of met het plaatsen en afleveren van een bestelling.

In eerste instantie is het model opgezet voor een periodiek stationair MBP. Later is de methode uitgebreid opdat niet-stationaire perioden, zoals rondom vakantiedagen, in de planningshorizon opgenomen kunnen worden. Uit de gedetailleerde simulatieresultaten blijkt dat het verval van BPPs aanzienlijk teruggebracht kan worden van 15 à 20% in de huidige praktijk tot slechts een paar procent of zelfs minder dan 1%. De beschikbaarheid van BPPs hoeft zelden een probleem te zijn.

Bloedbankmanagers zijn dermate positief over de ontwikkelde methode, dat nu een gebruikersvriendelijke versie van de ontwikkelde programmatuur draait bij een van de Nederlandse bloedbanken.

Deel II – Dynamische regeling van verkeerslichten (H5 tot H8)

Verkeerslichten zijn geïntroduceerd om het wegverkeer veiliger en efficiënter te maken. Sommige verkeersstromen kunnen tegelijkertijd groen genieten, mits ze elkaar niet kruisen. Verkeersdeelnemers in de andere verkeersstromen zullen dan echter moeten wachten totdat zij groen krijgen. De wachttijd wordt beheerst door een softwarematig gestuurde regeling van de verkeerslichten. Informatie over het aantal wachtende auto's op iedere rijbaan kan aan de softwarematige regelaar meegegeven worden. Stel er zijn F rijbanen, het aantal wachtende auto's kan dan worden aangeduid met de vector $\mathbf{q} = (q_1, q_2, \ldots, q_F)$.

De uitdaging om de lange-termijn-gemiddelde wachttijd voor automobilisten te minimaliseren op een enkel kruispunt kan geformuleerd worden als een MBP. Het MBP model met $(x, \mathbf{q})$ als toestandsbeschrijving, waarbij x de toestand ('kleur') van de verkeerslichten aangeeft. In eerste instantie veronderstellen we een stationair aankomstpatroon van auto's. In het MBP worden andere verkeersstromen, zoals voetgangers, fietsers en het openbaar vervoer buiten beschouwing gelaten. Niettemin worden uitbreidingen besproken in Hoofdstuk 6.

Het potentieel aantal wachtrijtoestanden groeit exponentieel met het aantal wachtrijen. Ter illustratie, als er 4 rijbanen samenkomen op een kruispunt, en op iedere rijbaan maximaal 9 auto's zullen wachten is het aantal mogelijke vectoren $\mathbf{q}$ reeds $10^4 = 10.000$. Als er echter 12 rijbanen zijn, dan zijn er reeds 1 biljoen toestanden mogelijk. Voor een optimale strategie heeft men voor iedere mogelijke toestand een optimaal besluit ten aanzien van het verkeerslicht te berekenen.

De optimale oplossing van een MBP met tientallen miljoenen toestanden kan niet in afzienbare tijd bepaald worden. Daarom zal men benaderingen moeten accepteren. Als het probleem van een enkel kruispunt een benadering vergt, dan heeft men voor het regelen van een netwerk van kruispunten zeker benaderingen nodig. In de praktijk treft men veelal heuristsche regels aan. De vaste cyclus, afgekort met FC voor Fixed Cycle, is wellicht de meest bekende: iedere stroom krijgt gedurende een vaste tijd groen en de stromen worden in een vaste cylische volgorde afgewerkt. Daarnaast zijn er drukte-afhankelijke regelingen zoals de cyclische uitputtende regeling, XC voor cyclic exhaustive control, waarbij een licht op groen blijft totdat er geen verkeer meer is nabij de stopstreep van de betreffende rijbaan.

FC is een interessante basisstrategie om tot een benadering van een optimale regeling te komen. FC doorloopt een cyclus van zeg D tijdslots, die we kunnen nummeren van 1 tot D. De lengte van een tijdslot is gelijk gesteld aan 2 seconden. Stel dat verkeersstroom 1 groen heeft gedurende slots 1 t/m 5, dan zijn gedurende slots 6 en 7 de lichten oranje, en gedurende slots 8 t/m D zal stroom 1 rood licht hebben. De kleur van de verkeerslichten kan onder FC dus eenvoudigweg bepaald worden op basis van de positie in de cyclus: het nummer van het tijdslot. De 'roodtijd' van een stroom is dus onafhankelijk van het aantal auto's dat staat te wachten op de andere rijbanen. De gemiddelde wachttijd per auto kan voor FC bepaald worden door de onderliggende Markov keten numeriek te evalueren. Een toestandsbeschrijving van de Markov keten voor stroom f bevat het slotnummer $t \in \{1, \ldots, D\}$ en het aantal auto's dat op de rijbaan staat te wachten q_f.

Middels een één-stapsverbeteralgoritme, kan een betere strategie dan FC bepaald worden. Daartoe zijn de relatieve waarden van toestanden voor een goede vaste cyclus nodig. Daar de toestandsruimte onder FC slechts twee-dimensionaal is, kunnen voor alle verkeersstromen f de relatieve waarden van alle toestanden (t, q_f) snel berekend worden. Hierbij moet wel rekening gehouden worden dat de Markov keten voor FC periodiek is, en dat in de berekening van de relatieve waarden de in principe oneindige horizon wordt afgekapt. De relatieve waarde van een toestand $(t, q_1, \ldots, q_F)$ onder FC, is de som van F relatieve waarden: één voor iedere verkeersstroom.

De verbeterde strategie heet RV (voor 'Relative Value'). RV kent geen vaste groen perioden en hoeft ook niet per se cyclisch te zijn. RV doorbreekt FC door de positie in de cyclus aan te passen, waardoor de groenperiodes van de verkeersstroom dynamisch verlengd, verkort, of beëindigd worden. Naast RV zijn er andere strategiën afgeleid op basis van de relatieve waarden onder FC.

De enige manier om de kwaliteit van RV te bepalen is door een aantal kruispunten te simuleren onder verschillende omstandigheden, bijvoorbeeld ten aanzien van de aankomstintensiteiten. RV is aldus vergeleken met eenvoudige regels zoals FC en een aantal varianten van de uitputtende regeling (XC). Voor infrastructuren met slechts een klein aantal stromen, zeg $F \leq 4$, kan de optimale oplossing voor het MBP berekend worden. De resultaten van een simulatiestudie zijn veelbelovend: in alle (volledig) symmetrische gevallen die beschouwd zijn, is voor alle stromen de gemiddelde wachttijd op lange termijn voor RV 20 procent lager dan voor FC. In Hoofdstuk 6 blijkt dat de gemiddelde wachttijd voor RV zelfs meer dan 70 procent lager kan zijn dan voor FC en XC.

Moderne technologie is in staat om de positie te bepalen van auto's die op weg zijn naar een kruispunt. Schattingen van de verwachte aankomsttijden van deze auto's kunnen

in de toestandbeschrijving worden opgenomen. De toestandsruimte per stroom, bestaat dan uit de toestand van het verkeerslicht, het aantal wachtende auto's en een vector met de aankomstinformatie voor de komende, zeg, vijf tijdslots (10 seconden). Het aantal toestanden neemt daarmee drastisch toe. Een optimale oplossing op basis van een MBP model kan dan niet bepaald worden. Daarom is de RV regel uitgebreid met aankomstinformatie. In verscheidene simulaties blijkt dat het toevoegen van 10 seconden aankomstinformatie, leidt tot een significante verbetering van RV (zie Hoofdstuk 7). Zelfs als de aankomstinformatie niet perfect is doordat auto's in het simulatiemodel met verschillende snelheden kunnen rijden, geeft RV met aankomstinfo goede resultaten.

In een netwerk van kruispunten worden lichten vaak statisch geregeld middels een netwerk FC, daar de lichten dan in principe op elkaar afgesteld kunnen worden zodanig dat een groene golf ervaren kan worden in 1 of meerdere richtingen. Vaak gaan groene golven verloren doordat auto's niet allemaal even hard rijden en doordat auto's vertraagd worden door andere auto's die nog staan te wachten. Vanuit het oogpunt van minimalisatie van de gemiddelde wachttijd kan het beter zijn om ook in een netwerk de lichten lokaal en drukte-afhankelijk te regelen. De RV regels met en zonder aankomstinformatie worden in Hoofdstuk 8 gesimuleerd voor verschillende soorten verkeersaders en netwerken.

De gemiddelde wachttijd wordt vergeleken met de best-gevonden netwerk cyclus (FC). Het blijkt dat een RV regel met aankomstinformatie een gemiddelde wachttijd geeft die veelal lager is dan die voor de gesynchroniseerde netwerk FC. Door afslaand verkeer en snelheidsverschillen blijkt de gemiddelde wachttijd in een netwerk van negen kruispunten voor FC meer dan 40% hoger te zijn dan die voor RV met aankomstinformatie. Een ander voordeel van RV is dat het bepalen van een goede basis cyclus veel makkelijker is dan het bepalen van de beste netwerk FC.

Conclusie

De twee praktische optimalisatievraagstukken hebben de neiging dermate omvangrijk te zijn, dat een optimale strategie niet berekend kan worden. Het aantal toestanden in het onderliggende MBP model is (te) groot doordat de zogenaamde toestandsruimte multidimensionaal is. De structuur van de toestandsruimte voor de twee problemen is verschillend, waardoor een verschillende aanpak nodig is.

In Deel I wordt de structuur van een voorraadprobleem bij bloedbanken bestudeerd en blijkt de *aggregatie* van toestanden mogelijk te zijn: toestanden worden geaggregeerd door bloedgroepen samen te nemen, door de vraag naar en de productie van BPPs te

modelleren in batches, en tenslotte door leeftijdsgroepen samen te nemen in een vaste aanvulregel.

In Deel II wordt strucuur aan het verkeerslichtenprobleem toegevoegd door een speciale strategie als uitgangspunt te nemen. *Decompositie* van de toestandsruimte blijkt dan mogelijk waardoor een *één-stapsverbeteralgoritme* toegepast kan worden. Decompositie vindt plaats op twee niveaus: voor de regeling van een netwerk van kruispunten wordt het netwerk eerst gedecomponeerd in de afzonderlijke kruispunten, en per kruispunt wordt het rekenwerk opgesplitst in rekenwerk voor iedere verkeersstroom afzonderlijk.

Grote gestructureerde Markov beslisproblemen kunnen aldus succesvol opgelost worden door de structuur van het probleem te benutten. Een uniforme aanpak van hoog-dimensionale Markov beslisproblemen bestaat wellicht niet; elk probleem vraagt om een probleemspecifieke aanpak. Dit proefschrift geeft inzicht dat kan bijdragen tot de oplossing van andere grote MBPs.

The Tinbergen Institute is the Institute for Economic Research, which was founded in 1987 by the Faculties of Economics and Econometrics of the Erasmus Universiteit Rotterdam, Universiteit van Amsterdam and Vrije Universiteit Amsterdam. The Institute is named after the late Professor Jan Tinbergen, Dutch Nobel Prize laureate in economics in 1969. The Tinbergen Institute is located in Amsterdam and Rotterdam. The following books recently appeared in the Tinbergen Institute Research Series:

393. G.I.J.S. VAN DE KUILEN, *The economic measurement of psychological risk attitudes.*
394. E.A. MOOI, *Inter-organizational cooperation, conflict, and change.*
395. A. LLENA NOZAL, *On the dynamics of health, work and socioeconomic status.*
396. P.D.E. DINDO, *Bounded rationality and heterogeneity in economic dynamic models.*
397. D.F. SCHRAGER, *Essays on asset liability modeling.*
398. R. HUANG, *Three essays on the effects of banking regulations.*
399. C.M. VAN MOURIK, *Globalisation and the role of financial accounting information in Japan.*
400. S.M.S.N. MAXIMIANO, *Essays in organizational economics.*
401. W. JANSSENS, *Social capital and cooperation: An impact evaluation of a women's empowerment programme in rural India.*
402. J. VAN DER SLUIS, *Successful entrepreneurship and human capital.*
403. S. DOMINGUEZ MARTINEZ, *Decision making with asymmetric information.*
404. H. SUNARTO, *Understanding the role of bank relationships, relationship marketing, and organizational learning in the performance of people's credit bank.*
405. M.Â. DOS REIS PORTELA, *Four essays on education, growth and labour economics.*
406. S.S. FICCO, *Essays on imperfect information-processing in economics.*
407. P.J.P.M. VERSIJP, *Advances in the use of stochastic dominance in asset pricing.*
408. M.R. WILDENBEEST, *Consumer search and oligopolistic pricing: A theoretical and empirical inquiry.*
409. E. GUSTAFSSON-WRIGHT, *Baring the threads: Social capital, vulnerability and the well-being of children in Guatemala.*
410. S. YERGOU-WORKU, *Marriage markets and fertility in South Africa with comparisons to Britain and Sweden.*
411. J.F. SLIJKERMAN, *Financial stability in the EU.*
412. W.A. VAN DEN BERG, *Private equity acquisitions.*
413. Y. CHENG, *Selected topics on nonparametric conditional quantiles and risk theory.*
414. M. DE POOTER, *Modeling and forecasting stock return volatility and the term structure of interest rates.*
415. F. RAVAZZOLO, *Forecasting financial time series using model averaging.*
416. M.J.E. KABKI, *Transnationalism, local development and social security: the functioning of support networks in rural Ghana.*

417. M. POPLAWSKI RIBEIRO, *Fiscal policy under rules and restrictions.*
418. S.W. BISSESSUR, *Earnings, quality and earnings management: the role of accounting accruals.*
419. L. RATNOVSKI, *A Random Walk Down the Lombard Street: Essays on Banking.*
420. R.P. NICOLAI, *Maintenance models for systems subject to measurable deterioration.*
421. R.K. ANDADARI, *Local clusters in global value chains, a case study of wood furniture clusters in Central Java (Indonesia).*
422. V. KARTSEVA, *Designing Controls for Network Organizations: A Value-Based Approach.*
423. J. ARTS, *Essays on New Product Adoption and Diffusion.*
424. A. BABUS, *Essays on Networks: Theory and Applications.*
425. M. VAN DER VOORT, *Modelling Credit Derivatives.*
426. G. GARITA, *Financial Market Liberalization and Economic Growth.*
427. E. BEKKERS, *Essays on Firm Heterogeneity and Quality in International Trade.*
428. H. LEAHU, *Measure-Valued Differentiation for Finite Products of Measures: Theory and Applications.*
429. G. BALTUSSEN, *New Insights into Behavioral Finance.*
430. W. VERMEULEN, *Essays on Housing Supply, Land Use Regulation and Regional Labour Markets.*
431. I.S. BUHAI, *Essays on Labour Markets: Worker-Firm Dynamics, Occupational Segregation and Workplace Conditions.*
432. C. ZHOU, *On Extreme Value Statistics.*
433. M. VAN DER WEL, *Riskfree Rate Dynamics: Information, Trading, and State Space Modeling.*
434. S.M.W. PHLIPPEN, *Come Close and Co-Create: Proximities in pharmaceutical innovation networks.*
435. A.V.P.B. MONTEIRO, *The Dynamics of Corporate Credit Risk: An Intensity-based Econometric Analysis.*
436. S.T. TRAUTMANN, *Uncertainty in Individual and Social Decisions: Theory and Experiments.*
437. R. LORD, *Efficient pricing algorithms for exotic derivatives.*
438. R.P. WOLTHOFF, *Essays on Simultaneous Search Equilibrium.*
439. Y.-Y. TSENG, *Valuation of travel time reliability in passenger transport.*
440. M.C. NON, *Essays on Consumer Search and Interlocking Directorates.*
441. M. DE HAAN, *Family Background and Children's Schooling Outcomes.*
442. T. ZAVADIL, *Dynamic Econometric Analysis of Insurance Markets with Imperfect Information.*
443. I.A. MAZZA, *Essays on endogenous economic policy.*

This thesis centers around two optimization problems that are of general interest: the inventory control of a typical perishable product (blood platelet pools) and the dynamic control of traffic lights.
Both problems can be formulated as multi-dimensional Markov Decision Problems (MDPs), which appear to be too large to compute an optimal solution. Nevertheless, by exploiting the problem structure, approximate solutions can be obtained by Stochastic Dynamic Programming (SDP). Simulation plays a crucial role in obtaining and evaluating these approximate solutions.

In Part I an SDP–Simulation approach is developed that alllows to reduce the fraction of blood platelet pools that outdate, from more than 15% in current practice to only 1% or even less. A user friendly version of the software that was developed during this PhD project currently runs at one of the Dutch blood banks.

In Part II the dynamic control of traffic lights is optimized, such that the average waiting time per car is minimal. New approximate solutions are developed that perform very well for signalized intersections in isolation as well as for networks of intersections.

This thesis is not only of interest for its applications but also for its insights in how to solve multi-dimensional MDPs.

René Haijema (1976) obtained a Master of Science in Operations Research, with distinction (cum laude), at the University of Amsterdam (UvA). During this study he worked on transportation planning problems for the Dutch Ministry of Justice and for the Dutch Railways (NS-reizigers). After a stay for one semester at the University of Otago (New Zealand), René worked on a PhD project at the UvA, which he combined with a part-time position as a teacher. Currently, he is assistant professor at Wageningen University and Research center (WUR). His general research interest is optimization within the fields of transportation and logistics.

Thela Thesis
Academic Publishing Services