A Markovian approach to the mathematical control of NPD projects

A Markovian Approach To The Mathematical Control Of NPD Projects

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
Rector Magnificus, prof.dr. R.A. van Santen, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen
op maandag 17 november 2003 om 16.00 uur

door

Andreea Bogdana Dragut

geboren te Bucharest

Dit proefschrift is goedgekeurd door de promotoren:

prof.dr.ir. J.W.M. Bertrand
en
prof.dr. A.G. de Kok

Copromotor:
dr. G.-J. van Houtum

A Markovian Approach To The Mathematical Control Of NPD Projects

Andreea Bogdana Dragut

CIP-DATA LIBRARY TECHNISCHE UNIVERSITEIT EINDHOVEN
Dragut, Andreea Bogdana
A Markovian Approach To The Mathematical Control Of NPD Projects / by Andreea Bogdana Dragut. – Eindhoven : Technische Universiteit Eindhoven, 2003. – Proefschrift.
ISBN 90-386-1658-9
NUR 919
Keywords : NPD Projects / Hierarchical Control Framework / Mathematical Modelling / Markovian Control / Non-stationary Markov Decision Processes / Monotonic Optimal Policies / Sample Path Analysis
Printed by Universiteitsdrukkerij Technische Universiteit Eindhoven
Cover design by Paul Verspaget

We know that one times one is one,
but an unicorn times a pear
have no idea what it is.
We know that five minus four is one
but a cloud minus a sailboat
have no idea what it is.
We know that eight
divided by eight is one,
but a mountain divided by a goat
have no idea what it is.
We know that one plus one is two,
but me and you, oh,
we have no idea what it is.
.....
Only you and me
multiplied and divided
added and subtracted
remain the same...
Vanish from my mind!
Come back in my heart!

”Another kind of Mathematics”, in The Grandeur of Coldness (1972)

***N. Stanescu** (1933-1983)*

English translation by G. Mustea

Contents

List of Figures

List of Tables

Acknowledgements

Although only my name appears on the cover of this dissertation, this research would not have been completed without the help of many others. First, I would like to thank Prof.dr.ir. J.W.M. Bertrand and Prof.dr. W.H.M. Zijm, who proposed the study of the NPD control problems from a mathematical perspective. A large part of this dissertation is a joint work with Prof.dr.ir. J.W.M. Bertrand. The collaboration with him was essential in dealing with the modelling issues of the NPD projects, and extremely fruitful and meaningful for the entire dissertation. I could come to him any time, with all my questions, problems, and achievements, and his continuous support was very encouraging. I am also very indebted to Prof.dr. A.G. de Kok, for directing the general line of the analytical part of this research and helping me to keep an eye on the practical aspects of my analytical results.

I would like to express my gratitude to dr. A.A.N. Ridder, for his careful reading of this dissertation and for his very useful comments and advice which lead to various improvements of the mathematical part of my dissertation.

Moreover, I would like to thank dr. G.-J. van Houtum and Prof.dr. J. van der Wal for helpful comments on the manuscript of the dissertation. I thank all the participating members of my dissertation committee for their enthusiasm and involvement.

I also thank my former professors and mentors from the University of Bucharest and from the Romanian Academy of Science, who guided my first steps into the wonderful world of research in Applied Mathematics.

I definitely have to mention the lively and humorous atmosphere of our research group, in which I especially felt the warmth of the best officemate, Cristina Ivanescu, of the always joyful and supporting Mustafa Dogru, as well as of Leonard Fortuin, Will Bertrand, Judith Spitter, and Jan Fransoo. I of course need to thank all the F-corridor dwellers for sympathizing with the rather Latin character of happiness freely blossoming in the tropical warmth of our office.

However, many months in Netherlands were extremely hard for me, and I would not have managed to navigate through the channels of life and bike over the windy dams, had I not discovered a refreshing lot of new friends. I am very grateful (in alphabetical order!) to Manuela Buda, Calin Ciordas, Darek Chodyniecki, Mustafa Dogru, Cristina Ivanescu, Laura Maruster, Silvia Nedea, Christine Pelletier, Samia Rouibah, Simona Vlad, and Marton Zelina.

I also hope this joy of accomplishment somehow fills in the gaps left by my many hours in front of a piece of paper, a book or a computer, yet met with so much warmth, love, trust and understanding by my family. My work would not have been possible and would not have had any sense without their wholeheartedly given support.

Part I

Modelling

Chapter 1

Introduction and Problem Statement

1.1 Motivation of the research

In the project planning and control literature, the breakthrough that led to the Program Evaluation and Review Technique in 1958 has been followed by many additions and revisions. In the early PERT system descriptions, the PERT network was updated on a regular basis. This allowed a formal process of status updating of all tasks scheduled to be started, or completed during the prior period, of producing new time estimates for future tasks, as well as the re-design process. According to (DOD/NASA, 1962), the current plan revision in the PERT/COST System updated the precedence relationships and/or the content (deletion and/or addition of tasks to the network) of the network. Later, in the project planning and control research, the project concepts were narrowed down, and the dominant concepts were:

- *precedence*: a fixed partial order on a fixed set of tasks for the entire project duration;

- *time-cost task trade-off*: task duration may be shortened, at a certain cost;

- *task indivisibility:* a task is a unity with start and finish times.

This explains why the various approaches to incorporate uncertainty in the project planning and control techniques ((Elmaghraby, 1995); (Herroelen et al., 1998); (Krishnan and Ulrich, 2001); (Tavares, 2002)) viewed a task as a unity and addressed the uncertainty issue in the duration of tasks, while considering the product to be fully defined at the start of the project. Explicit trade-offs in the product definition process in terms of design tasks to be performed did not appear, not even in the generalized activity networks approach ((Elmaghraby, 1995); (Dawson and Dawson, 1995)), which assumed only an early partial product definition.

According to (Pich et al., 2002) these attempts of modelling the uncertainty in project management are referred to as an *instructionist* approach: policies are derived – either a priori or as the project is executed – that completely determine the tasks executed in response to the decisions taken by management. They do fail when the initial model is not adequate. This can happen either due to the project ambiguity (i.e. a lack of awareness of the project team about certain states of the world or causal relationships) or due to the project

complexity (i.e. many different decisions and states of the world parameters interact, making it difficult to asses the effect of decisions taken). (see (Pich et al., 2002)).

Recently, in the New Product Development literature another form emerged for incorporating the uncertainty, based on ”flexibility”, which is the use of technologies and processes that accommodate multiple possible outcomes of the project. Thus, in a highly uncertain environment (see (Sobek et al., 1999)) it is recommended to use a ”set-based engineering” approach in which multiple solutions should be pursued ”in parallel”, choosing the best solution once their outcomes are observable (see also (Committee, 2000)). (Bhattacharya et al., 1998) formalize trade-offs underlying the new product definition process emphasizing the uncertainty caused by a highly dynamic market situation. Also, the empirical research of (Tatikonda and Rosenthal, 2000b) suggests that in many firms, NPD projects start with lax specifications of the new product requirements, which evolve during the project. This level of uncertainty may be far beyond what can be modelled by Beta probability density functions for the duration of design tasks (PERT) and what can be modelled by probabilities that design tasks will have to be redone (GERT) ((Dawson and Dawson, 1998); (Oorschot, 2001)).

In summary, this new stream of research concentrates on complete probability spaces with (subjective) probabilities – that is the project team knows the event is possible but they do not know whether it will happen. The implicit assumption of such an approach is that it is impossible to ”manage events that cannot be foreseen” (see (Wideman, 2000), (Williams, 1999)).This alternative way of modelling the projects uncertainty is called in (Pich et al., 2002) *selectionism*. It is an extension of the instructionist approach in the sense that the project management still relies on its ability to identify an optimal policy, modified over time as the project model evolves. In an NPD environment the selectionist approach might be viewed as planning multiple alternative product definitions, and retaining one of those with the best market payoff value. The intermediate results should be shared among these alternatives, which will thus all contribute to a successful new product.

In this research we focus on planning and control methods for a new type of projects called time-constrained NPD projects with high technological product or process uncertainty. They are referred to as experiential NPD projects (see (Eisenhardt and Tabrizi, 1995)). *Experiential product development* projects consists of *uncertain, ill-defined, and unstable* design tasks. At the start of the project, it is uncertain which design tasks are really necessary for realizing the product specifications and it is even uncertain which (or the extent to which) product specifications can be realized at a certain deadline. The set of design tasks is often reorganized during execution, and the product specifications of the product are gradually reconsidered, fact sustained by numerous researchers ((Dawson and Dawson, 1998), (Bourgeois and Eisenhard, 1988), (McDermott, 1999), (Turner and Cochrane, 1993); review in (Krishnan and Ulrich, 2001), (Oorschot, 2001)). Moreover, a recent survey by (Thomke and Reinerstein, 1998) showed that only 5% of product developing firms have complete product specifications before starting a design, and on the average only 58% of specifications are available before the design process begins. This type of projects has emerged over the last decade in industry, and empirical research shows that its planning and control are strongly influenced by its technological uncertainties. A model for the operational control of this type of process has not been formulated before.

Existing NPD models focus on market uncertainty and do not consider the technological uncertainty appearing inside the firm as the result of its own innovation process. In this research we propose a general control framework for managing NPD projects with a high technological uncertainty under tight time constraints, including the experiential NPD

projects. We explicitly allow the product definition to evolve after the beginning of the detailed design phase. This is achieved by considering that product specifications can be broken down via a specification tree into design tasks, and that design tasks can be performed at different performance levels. We use these levels as decision variables. Hence, the required product specifications can be updated during the project, and the new product is dynamically redefined until the deadline. Thus, a second contribution of our framework and of its subsequent computational models is to create an axiomatic system making explicit in measurable variables the changes induced by the operational uncertainties. In the literature no other approaches are found which integrate the technological uncertainty with the quality control under time constraints. Thus, we expound the trade-offs in the new product definition in terms of which design tasks at which performance level should be done before the deadline.

Another important aspect of the NPD projects that was previously neglected by the analytical models is the human aspect. NPD projects have as primary resource the engineers from the development team, not machines. Using psychological literature as well as recent empirical research on NPD projects, we model the behavior of engineers under time pressure, and their ability to perceive the concurrency and the relative urgency of design tasks.

Additionally, we have connected the value of product specifications to the expected market value of the new product, using models from marketing literature. The key modelling elements and relationships are based on recent empirical research from various fields as product innovation, quality function deployment, design activities definition, concurrent engineering, project structuring and management. The goal of such an approach is to facilitate the evaluation and acceptance of the computational results by managers. Also, the initial control framework forms a basis from which we can derive more realistic constraints for different computational models, allowing for a fair comparison of what the outcome of a computational model is, versus its hypothesis. The control framework is used in this research to formulate solvable mathematical problems, but it can also guide other formulations of NPD projects stochastic models, as similar frameworks have previously done for production processes in (Dempster et al., 1981), (Bensoussan et al., 1985) and (Hackman and Leachman, 1989), (Charalambous et al., 2000).

We formulated two mathematical problems specific for controlling time-constrained NPD projects with high technological product or process uncertainty. Their formulation and their analytical solutions are to the best of our knowledge new. The first problem was a multiple-choice knapsack problem. By formulating it as a discrete deterministic dynamic problem, we obtained a graph structure of the problem space. Based on established results from heuristic search algorithms, we proposed an A* type algorithm to efficiently search an optimal solution using the DP graph structure. The second problem was formulated as a discrete-time, finite horizon non-stationary Markov decision process. To enable a more efficient computation of optimal policies in Markov models of sequential decision processes, one is often interested in finding structured policies (monotonic, convex, etc.). To reduce the exponential growth with respect to the size of our decision problem, and to enable the derivation of numerical solutions, we introduced a new type of structured policies called weakly monotonic. Formulating the problem in a dynamic programming setting, it is shown that the optimal policy follows a weakly monotonic optimal control by establishing the supermodularity of the objective function. This is a new result, extending the monotonicity theory and partial ordering programming techniques to bounded subsets without holes of integer vector lattices.

1.2 Problem statement and research questions

The research presented in this dissertation is meant to provide a general framework to model the quality-time-cost trade-offs underlying the time-constrained NPD projects with high technological product or process uncertainty and to be a contribution to the area of developing and solving mathematical planning and control methods for these projects. In particular we are interested in modelling the interaction between the low level scheduling information and high level planning information.

In order to meet these goals the following questions are investigated.

1. What are the specific requirements needed for the planning and control of time-constrained NPD projects with high technological product or process uncertainty?

2. Which information should be exchanged between the low level of scheduling and the high level of planning?

3. What degree of detail is needed in the mathematical models of these levels?

4. To what extent can existing planning and control techniques be used for the control of time-constrained NPD projects with high technological product or process uncertainty? Which are the new techniques for these projects?

5. How can new/already existing planning and control techniques be used to predict the outcome of the time-constrained NPD projects with high technological product or process uncertainty?

6. What are the characteristics of the policies which control the quality achieved at the deadline by a time-constrained NPD project with high technological product or process uncertainty?

The methodology that is used for answering these research questions is described in the next section.

1.3 Research methodology

In this section are described the steps that are taken to perform this research. These steps are:

- integration in a mathematical control model of the key characteristics of time-constrained NPD projects with high technological product or process uncertainty, as found by earlier empirical research

- control model's goals analysis and validation of the approximation made for the degree of detail needed in the high planning level of the control model

- development of mathematical solution techniques

- analysis of solutions

The final goal of my research was to build and present solutions for an adequate model of time-constrained NPD projects with high technological product or process uncertainty. The models in OR are built using mathematical relationships, which correspond to relationships in the real world, like in our case, time and technological constraints. It it important to emphasize that a model is only an approximation of the real process being modelled and that different models can be created for the same process.

Before starting with the mathematical modelling, in order to get a broader view of the problem, we asked the following questions:

- What are the goals of our mathematical model?
- Which means are available to achieve these goals?
- Which precision is required in achieving the goals?

Our goal was to be able to maximize the new product quality (in terms of market value) under a high time constraint and a high uncertainty regarding the number and structure of the design tasks defining the new product. The means included in our case addition/deletion of design tasks, addition of new design activities, changing the design tasks targeted quality and implicitly their solving time, allocating design tasks to engineers, introducing safety margins. The precision of achieving at the deadline a new product which is at least fully functional is defined in probability terms.

These questions easily led to the conclusion that the general problem was too complex to be straightforward and completely solved by means of already existing mathematical tools. It was then a challenge to select appropriate computational sub-models which retain the most significant features of the different parts of the process under study. (Williams, 1978) gives a number of motives for building such computational models:

- The actual exercise of building a model often reveals relationships which were not apparent to many people. As a result, a greater understanding is achieved for the process being studied.
- Having built a model, it is usually possible to analyze it mathematically to help suggest courses of action which might otherwise not be apparent.
- Experimentation is possible with a model whereas it is often not possible or desirable to experiment with the process itself.

In the next section we describe the content of each chapter of this dissertation.

1.4 Outline of the dissertation

The remainder of this dissertation is organized into two parts.

Part I is dedicated to the modelling issues. In Chapter 2 the characteristics of time-constrained NPD projects with high technological product or process uncertainty are expressed in quantifiable measures and integrated into a general hierarchical control framework with multiple review periods. The key modelling elements are indicators of the duration of the NPD project as well as of the cost, the quality and the market value of the new product.

Chapter 3 presents the mathematical formulation of the hierarchical control framework, and discusses already available operations research methods and techniques that can be used for the project analysis and for solving the detailed and aggregate decision level problems.

Part II is dedicated to the analytical solutions of the problems constructed in Chapter 3. Two distinct problems are investigated.

In Chapter 4 an analytical solution is presented for the problem of allocating concurrent NPD design tasks to the engineers, at the detailed planning level. The other chapters are devoted to solving the aggregate decision problem in a multi-period setting. In Chapter 5 we first introduce simple heuristics for both the engineering and the detailed planning processes, and we construct a simple queueing model to estimate the solving time distribution of design tasks in NPD projects. Later in the same chapter, the use of those heuristics allows the computation of transition probabilities for a non-stationary Markovian decision process model of the aggregate decision problem in a multi-period setting.

Due to the curse of dimensionality, it was only theoretically possible to derive the optimal policies for the aggregate decision problem. Thus, in Chapter 7 and Chapter 8 we investigated the structural properties of the optimal policies in two particular cases of the general aggregate decision problem.

In Chapter 7 we focus on a concurrent NPD (i.e. without precedence constraints), consisting of concurrent design tasks and described by a discrete-time, finite horizon non-stationary Markov decision process. To enable a more efficient computation of optimal policies in our Markov model we derive new general conditions of obtaining weakly monotonic optimal policies. This leads to a new weakly-monotonic backwards induction algorithm, as well as to some robustness properties. General mathematical results that support this analysis can be found in Chapter 6.

In Chapter 8 we focus on an NPD project with precedence constraints, consisting of sequential design tasks only, described by a discrete-time, finite horizon non-stationary Markov decision process. We assume that any design task is available as soon as we finished with its direct predecessor. Using sample path analysis we reduced the initial multidimensional control problem to a unidimensional one in both state and action space. In the new control problem the optimal policy will decide only on how many review periods the team of engineers should work on each design task, being optimal to always choose as decision, while working on a design task, the maximal performance level.

Finally, in Chapter 9, conclusions are drawn and suggestions for further research are given. Using the characterization of the optimal policies in those particular cases, one can reduce the general aggregate decision problem from a multidimensional one to a unidimensional one, and derive afterwards heuristic optimal policies for it.

Chapter 2

A Hierarchical Control Framework (HCF)

2.1 Introduction

Using recent empirical studies, we formulate in this chapter a general framework of the hierarchical control processes needed for managing a new product development (NPD) project with a high technological uncertainty, under tight time constraints. Considering the project delivery time and resources as given, the project and its control are organized to solve the uncertainty in the new product specifications through repeated internal adjustments and interactions with customers. Our framework integrates the uncertainty regarding both the market requirements and technological uncertainties. They lead to the addition/deletion of design tasks, and to a stochastic solving time of the design tasks.

Introducing concepts formalizing the quality-time trade-off in the product specifications, this chapter contributes to the area of NPD project control models, and to the development of management-related NPD project control concepts, both areas presenting research opportunities according to (Brown and Eisenhardt, 1995), and (Krishnan and Ulrich, 2001).

Our general framework manages NPD projects with a high technological uncertainty under tight time constraints, including the experiential NPD projects, by explicitly allowing the product definition to evolve after the beginning of the detailed design phase. A first contribution of our framework is to create an axiomatic system making explicit in measurable variables the variation induced by the operational uncertainties. In the literature no other approaches are found which integrate the technological uncertainty with the quality control under time constraints. Thus, expounding the trade-offs in the new product definition in terms of which design tasks and up to which extent should be done before the deadline.

The framework's key modelling elements and relationships are based on recent empirical research, which facilitates the evaluation and acceptance of the computational results by the management practitioners. So a second contribution of the framework is to form a basis from which we can derive more realistic constraints for different computational models, allowing for a fair comparison of what the outcome of a computational model is, versus its hypothesis. The framework is used in the next chapters to formulate solvable mathematical problems, and it can also guide other formulations of NPD projects stochastic models, as

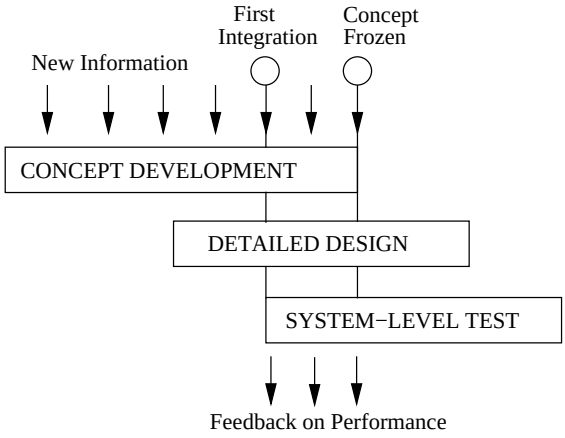

Figure 2.1. Phases of a flexible NPD process

similar frameworks have previously done for production processes in (Dempster et al., 1981), (Bensoussan et al., 1985) and (Hackman and Leachman, 1989), (Charalambous et al., 2000).

Section 2.2 describes a hierarchical control model with multiple review periods, which evaluates the NPD project progress and integrates the uncertainty regarding the product specifications, the design tasks, and the time needed for realizing the project. The model is based on accepted scientific knowledge on product innovation, quality function deployment, design activities definition, concurrent engineering, project structuring and management. In section 2.3 we define the model elements that play a role in the control framework. They will be used in the mathematical modelling of the decision processes in the hierarchical control structure of the framework. This mathematical model will be presented in Chapter 3. The section 2.4 concludes this chapter, by discussing the implications of developing a general framework for managing NPD projects with a high technological uncertainty under tight time constraints.

2.2 A multiple review periods hierarchical New Product Development control structure

A complete NPD project can be divided in a sequence of phases (Ulrich and Eppinger, 2000). For the design process, we consider the definition of (Doumeingts et al., 1996): it translates customer/market requirements/specifications into a product definition and a manufacturing process definition. Similarly to (McCormack et al., 2001), we distinguish three *overlapping* phases: the *system design/concept development* (performing a first work breakdown from customer needs into product specifications, and from product specifications into design tasks), the *detailed design phase* **(**consisting of solving the design tasks**)**, and the *system level test* **(**integrating the solved design tasks result into a complete system and tested**)** (see Figure 2.1).

The first and the last phase are too problem-dependent to be integrated in a general mathematical framework of the operational process. Therefore, the aim of this research is

to construct a dynamic model for the management of the time and resource aspects of transforming the customer needs into detailed product specifications during the detailed design phase, taking into account the overlap with the other two phases.

During the concept development part, the initial system specifications are set. They describe in precise, measurable terms what the product has to do. With them start both the development of a first work breakdown structure (WBS) and the partitioning of each system specification into a consistent set of specifications for all the intended units, assemblies and modules of the new product (i.e. a specification tree).

As in (Aslasken and Belcher, 1992) and (Shtub et al., 1994) we consider the WBS as being a product-oriented tree, which consists of a number of levels, starting with the complete product, and progressing downwards through as many levels as are necessary to obtain design elements that can be assigned to and performed by one of the engineers. Once this is done, it is possible to arrange all its design elements in a network resembling a directed tree with a single root node. The elements on the lowest level are always completely described by: a complete task statement (i.e. what work has to be accomplished); an identification of the necessary prerequisites to start it; a detailed description of what the output or result of the work should be and in what form is to be presented (Aslasken and Belcher, 1992). Elements on higher levels may or may not have this property. The lowest level design elements are called *design tasks*. To secure accountability through the design elements created by the WBS partitioning, a first specification tree will also be created. Thus, we assume a *one to one correspondence between a design task and a product specification from the module level of detail* (see Figure 2.2).

Later in time, more design tasks may emerge; design reviews are needed for mainly three reasons. First, no matter how uncertain the project is, design tasks have to be defined before its start, otherwise the WBS structure is not feasible (see (Shtub et al., 1994)). So, in real life, the management of the project will be forced to specify some design elements by decomposing them into design tasks, without being sure that the work content of those design tasks reflects exactly the achievement of that design element. Thus, only for small periods of time during the detailed design phase, the relationships between design tasks as well as their number can be viewed as stable. Second, the product specifications and their refinements are established before knowing all the constraints that either the technology or the market places on what can or should be achieved at the design tasks level. Third, unplanned design tasks may emerge as a result of the feedback from the system level tests.

Our NPD project control model performs decision/scheduling/execution cycles, each time taking into account the new surroundings it is facing. At the beginning of each new cycle, the state of the system is reviewed and updated by observing the technological knowledge accumulated at the engineering level, and by incorporating new information about customer needs. Thus, the uncertainty in the new product definition is decreased in time and this model structure allows the controller to adapt its decisions to changing conditions.

The planning and control problem at the beginning of each new cycle is hierarchically approached for two reasons. First, by decomposition of the overall planning problem into several sub-problems, the complexity of the planning problem is reduced. Second, the effects of uncertainty regarding the structure and solving times of design tasks are split over the levels. Thus, hierarchical planning leads to a consistent and controllable planning problem. The decisions made at a higher planning level provide targets and restrictions to the lower level decision making.

Our approach to NPD projects is supported by the recent research in organization

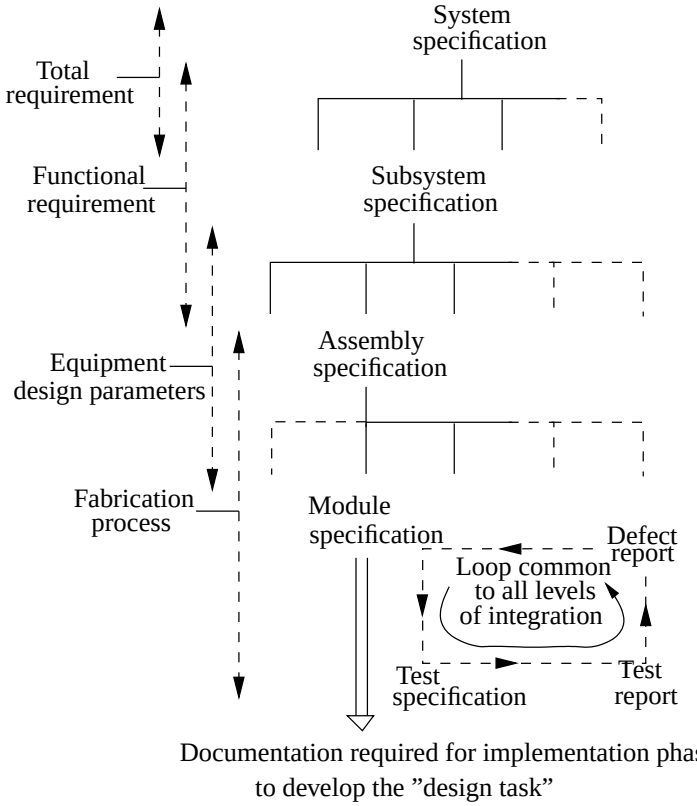

Figure 2.2. Specification tree after the flow-down of product specifications, (Aslasken and Belcher, 1992)

design of (Haque et al., 2000). Their paper models and analyzes the NPD project and organization in terms of tasks, teams, roles and communication links, relating the project hierarchy to the organization hierarchy. We assume in this paper similar project and resource levels, but for the internal structure of these levels we also use the research results of (Oorschot, 2001), which enable us to mathematically formulate the control problem of each of those levels. Thus, at the highest level, the project manager evaluates the overall NPD project progress and integrates the market and technological uncertainties. He decides then the structure of the network of design tasks to be performed by the NPD team in a short-time planning horizon. At the level of the NPD team, the design tasks are scheduled to the engineers, such that each engineer receives a sequence of design tasks from the network to be solved (any two such sequences are disjoint). In Figure 2.3 such a general NPD project and resource structure is presented; each of the involved notions and concepts are thereafter detailed and enriched.

In Section 2.3 we address in detail the quality, time and resource characteristics of NPD projects from a conceptual, and modelling point of view. As in (Ulrich and Eppinger, 2000) we use the term *product specifications* for the key product design variables. We introduce new concepts for the design tasks internal structure, as well as measures of their relative importance for realizing the product specifications. By a frequent NPD project progress eval-

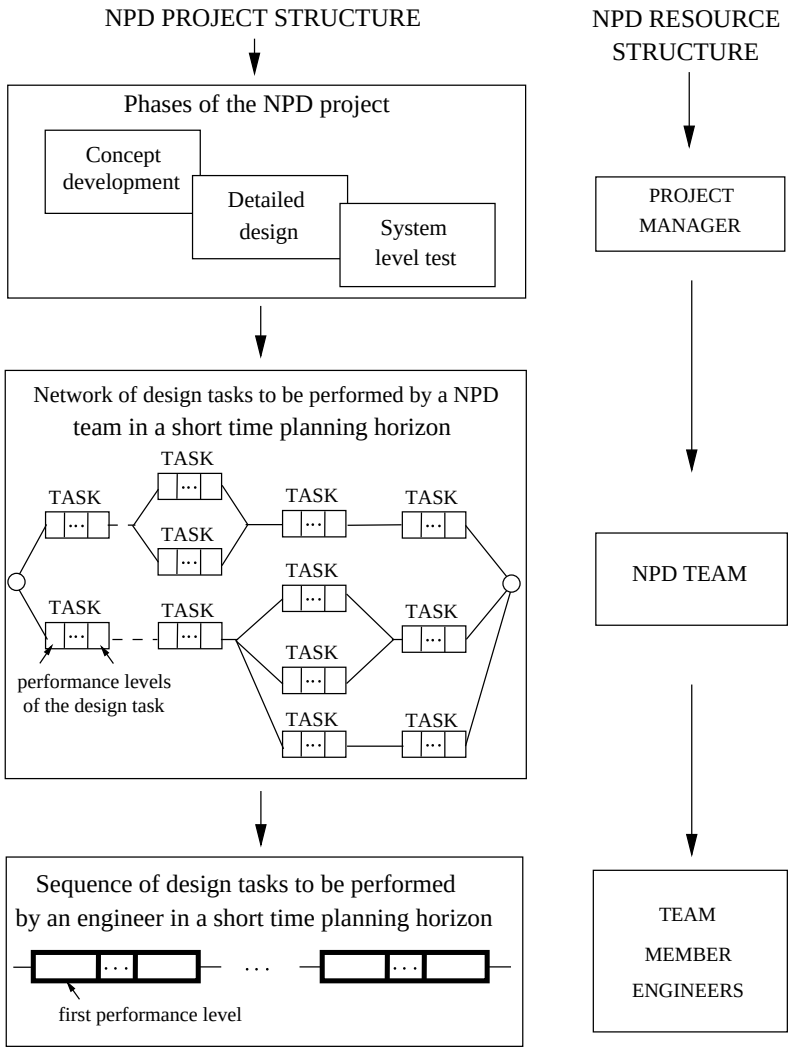

Figure 2.3. NPD project and resource structure

uation, these concepts allow either an optimal adjustment of product specifications, given the resources remaining until project deadline, or a project abandonment. The last situation occurs when either the project exceeds its budget or cannot longer meet the minimal requirements with a sufficiently high probability regarding the product specifications.

The proposed control model of the NPD project is a discrete time one. The project is reviewed at equidistant points in time until the deadline, T. The hierarchical structure proposed for each review period corresponds to a decision/scheduling/execution cycle, and consists of three levels: *aggregate decision level*, *detailed planning level* (rescheduling decision, scheduling), and *execution level*.

2.3 Model elements

Our NPD control structure has multiple review periods. Thus, the essential modelling assumptions are related to how we model the current knowledge about the NPD project at the beginning of each review period. To end up with a practical operational control framework, it is necessary to make the transition from a qualitative analysis dealing with concepts and structures, to a quantitative analysis in terms of measurable variables. We select a state variable set based on the significance of each variable in explaining some aspect of the product's status during its development, and to their aggregate ability of conveying the product status at each point during its development. This set of variables contains indicators of the duration of the NPD project as well as of the cost, the quality and the market value of the new product.

In this section, we first give the exhaustive list of the key elements and assumptions embedded in the structure of the model. For each of them we present the connections with existing literature as well as indications of how their values can be computed. Thereafter, we briefly describe the control levels in terms of the control elements introduced.

The NPD project model elements are grouped into:

2.3.1 General constraints:

- *a fixed development budget for the NPD project*

- *a fixed deadline for the NPD project:* ((Eisenhardt and Tabrizi, 1995); (Reppening, 2000); (Oorschot, 2001))

- *a set of current customer needs with their corresponding normalized importance weights (i.e. the weights sum up to one)*

The importance weights can be found using the Analytical Hierarchy Process or a similar procedure ((Kusiak, 1995)).

-a set of current product specifications with their:

- *ideal and minimal target values for their corresponding metrics* ((Askin and Dawson, 2000); (Ulrich and Eppinger, 2000))
- *relative importance rating*

As in (Ulrich and Eppinger, 2000) we consider that each specification consists of a *metric* (i.e. "Lateral stiffness at brake pivots") and a *value* (i.e. "more than $650\mathrm{kN/m}$") which can be a number, range, or inequality. The process of establishing the current product specifications includes the following (see (Ulrich and Eppinger, 2000)): prepare the list of metrics and their relative importance, using needs-metrics matrix if necessary; collect the competitive benchmarking information; set ideal and minimal (i.e. marginally acceptable) target values for each metric; refine the specifications, making the trade-offs with the technological and cost constraints; flow down the specification to the lowest level of the WBS structure. The initial metrics should be complete, practical, in general dependent variables. After their flow-down via the specification tree we set the specifications of the lowest level of the tree as the current product specifications.

(Krishnan and Ulrich, 2001) acknowledge in their review that it is useful to represent a new product by both customer needs and product specifications. These attributes are an abstraction of a product, and attribute-based methods are limited in their ability to represent the customer satisfaction or market share, but it is the way recognized in the NPD literature to build what is called the "core product concept." According to the same survey, it is an accepted knowledge that given the representation of a product into attributes one can determine weights for the customer needs, and then target values for the product specifications can be obtained. (Forman and Gass, 2001) give an overview of areas in which the Analytical Hierarchy Process (the general method of obtaining the weights, the target values, and how to order the different design alternatives function of them) has been successfully applied.

The process of setting the metrics ideal target values is generally a subjective and heuristic one. However, mathematical models are also available (Askin and Dawson, 2000).

The importance rating of a metric is derived from the weights of the customer needs it reflects. (Ulrich and Eppinger, 2000) do not recommend a formal algorithm, but for the case of few important specifications, conjoint analysis can be a solution. If the level of detail of the product specifications supports the assumption of independent metrics their relative importance can be obtained via regression analysis ((Askin and Dawson, 2000); (Yoder and Mason, 1995)).

2.3.2 The set of system states:

The state of the system at the beginning of each review period is *a directed acyclic graph of design tasks with a source and a sink (i.e. a network)*. Similarly to (Sieger et al., 2000), we choose the set of states of the system without abandoning the proven benefits provided by network analysis in the management of projects.

Ideally, a decision making moment should occur whenever a design task starts, an unplanned design task emerges or a new activity arrives changing the known information about the project. To avoid the discretization of the project duration into very small units, one can decompose the project into stages (review in (Tavares, 2002)). We construct stages for our graph by associating a representation into independent sets to it: sets of unordered design tasks (no precedence relations between any two of them) and all having the same length of *the longest path from the sink node to them* (in the precedence graph) (see Figure 2.4).

The decomposition of the set of vertices of a directed acyclic graph into a $T-$partition (i.e a partition with T stages) is unique. Thus, the concept of a T-stage network naturally associates the t-th decision moment with the allocation of design tasks from the t-th set of the partition of nodes. Also, empty sets can be added to achieve equidistancy of the control points, or for being able to control more often than the number of independents sets. The partition of the set of nodes gives the sets of design tasks that can be allocated at the beginning of a review period. All the design tasks allocated in the same review period can be performed in parallel. However, design tasks allocated in previous periods review periods might not be finished, thus during a review period the detailed planning problem has to take into account a general digraph of precedence relationships.

For controlling *coupled (interdependent)* tasks, the Design Structure Matrix (DSM) was developed as an alternative for formal project-scheduling representations ((Eppinger et al., 1994); (Krishnan and Ulrich, 2001)). If the DSM can be organized into a lower triangular form, the coupling is eliminated. Otherwise, if one would collapse the diagonal blocks for reducing the DSM matrix to a precedence network, the essential information on the de-

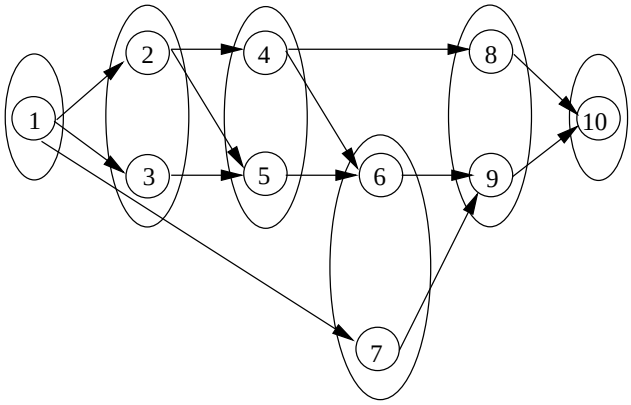

Figure 2.4. Stages of the directed acyclic graph describing the state of the system at time instant t_0, built using the longest path from the sink

sign iteration within blocks would be lost. However, DSM-s require complete information on the number of design tasks and their relationships, requirements not satisfied in the case of a NPD with a high technological uncertainty. Our state representation tries to capture the coupling information, while not giving up to the modelling of the technological uncertainty. In the beginning of the NPD project we start with a DSM representation, we collapse each remained block on its diagonal into one design task, and thus we model the task in detail, up to all its constituting activities. Thus, we construct a network of design tasks, which is however stable for only one review period. At its end, the structure of each design task is updated, and so we keep track of most of the changes that occurred in the former blocks, including a possible re-sequencing of the block elements.

2.3.3 The performance, cost and market-payoff structure:

The performance and valuation control concepts are to the best of our knowledge new. They follow from the assumption of the one-to-one mapping between the current product specifications and the design tasks, justified earlier (see Figure 2.2).

Each design task has

- a number of *increasing performance levels* giving the quality of its execution. They are induced by a scaling in between the minimal and the ideal target values for the corresponding current product specification metric. Each performance level consists of a *list of planned activities* (to be sequentially performed (Aslasken and Belcher, 1992)) with solving times random variables that are independent identically exponentially distributed (see for empirical evidence (Best, 1995); (Reed, 1988)). To attain a performance level, we assume that the engineer has to sequentially execute the design task at all previous performance levels, which implies different stochastic durations for the solving time, depending on the level initially specified (see Figure 2.5). The split of each level into activi-

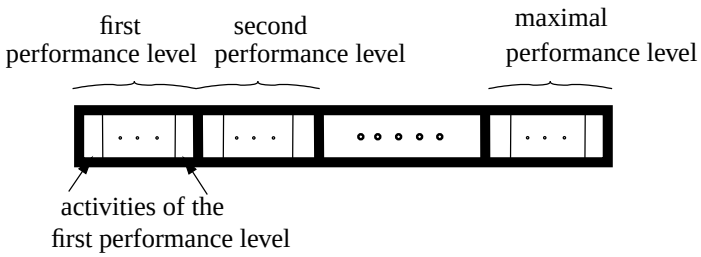

Figure 2.5. Internal structure of a design task

ties gives a uniform measure of the difficulty implied by its realization. For each design task a *minimal performance level* has to be achieved.

In their research on how a novice and an expert solve physics and mechanics problems ((Reed, 1988), pp.326-329) and (Larkin and Raif, 1979) found that for both categories the elapsed time between the solving of successive problems was random. Even if the novice problem solving process showed more randomness than the expert's one, for both cases a negative exponential law can be considered as governing their problem solving process. Moreover, according to (Loehle, 1994) innovation can be seen as a cumulative series of problem solving activities, starting with an a flash of insight ("eureka") which is mainly characterized by its suddenness. In this paper we assume that an average engineer working on an innovative design activity requires an exponentially distributed time to solve a problem.

In systems engineering, (see (Aslasken and Belcher, 1992) pp. 45) it is a common assumption to consider the internal structure of a design task to be sequential/linear. The mathematical advantages of combining this assumption with the assumption of the identical exponentially distributed design activities are important. A sequence of exponential design activities leads to an Erlang distributed solving time for the entire design task, which is exactly the distribution that the recent research of (Innam, 1999) considers appropriate for manufacturing systems with unreliable machines. Based on flexibility and goodness-of-fit using moment matching for the data sets, this distribution fits well. The research of (Oorschot, 2001) pp.54-56 also finds a similar type of skewed, delayed, long-tailed distribution for the solving time of design tasks in NPD projects.

- *a cost function* gives the incremental change in cost associated with performing one more planned activity, of one of its performance levels. This function models all non-engineer capacity related costs.
- *a time dependent design task contribution function*. At the beginning of each review period, using both the specification tree corresponding to our NPD project and its associated Quality Function Deployment (QFD) waterfall chart, one can obtain the *values* $\Theta(n, \delta, t)$ *giving the current maximal contribution of each design task* n*, in achieving each customer need* δ. Afterwards, for each design task its current *contribution function in achieving a customer need* δ is obtained by *scaling its corresponding current maximal contribution value, for its performance levels* (i.e. if L_{max} is the maximal number of levels for the design task n, its contribution function might be $f(n, l, t) := \Theta(n, \delta, t)\frac{l}{L_{max}}$). The scaling is not necessarily linear.

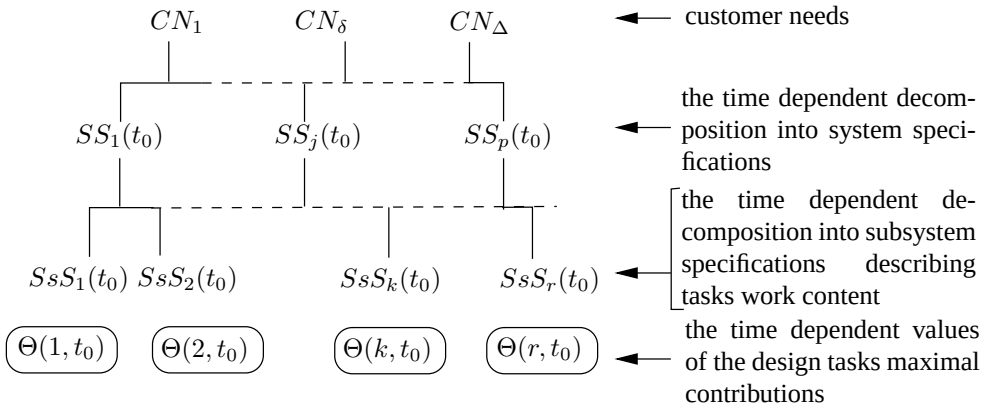

Figure 2.6. A particular specification tree

At the beginning of a review period t_0, we obtain the design tasks maximal contribution values as follows. Without loss of generality we consider the specification tree in Figure 2.6

Given the existence of a specification tree for our new product, one can always obtain the associated Quality Function Deployment charts linking system specifications to subsystem specifications, and subsystem specifications to sub-subsystem specifications, and so on down to the specification level required to identify the design tasks. According to a well-known method called AHP (Analytical Hierarchy Process), we do the following:

1. given $\delta = 1, ..., \Delta$ objectives, determine their respective weight coefficients ω_δ.
2. for each objective δ, compare the $j = 1, ..., p$ alternatives and determine their weight coefficients $\alpha_{\delta j}$.
3. determine the final alternative j_0 and its weight coefficients $A_\delta = \alpha_{1j_0}\omega_1 + ... + \alpha_{\Delta j j_0}\omega_\Delta$ with respect to all the objectives.

The AHP (see (Forman and Selly, 1999), (Forman and Gass, 2001)) is a resolution of choice problems in a multi-criterion environment based on pairwise comparisons of objectives and alternatives.

Applying the steps 1. and 2. for the customer needs and the ideal targets of the product specifications metrics, we get a set of coefficients which reflect how the ideal realization of each product specification metric j affects each customer need. The same procedure applies for product specifications to subsystem specifications, and so on, obtaining the Quality Function Deployment (QFD) waterfall chart shown in Figure 2.7.

The $w_\delta(t_0)$, $\delta \in \{1, ..., \Delta\}$ represent the normalized weights (i.e. $\sum_{\delta=1}^{\Delta} w_\delta(t_0) = 1$) corresponding to the current customer needs. The first QFD chart contains the normalized quantifiers, $\alpha_{ij}(t_0)$, $\delta \in \{1, ..., \Delta\}$, $j \in \{1, ..., p\}$, for the contribution of the maximal target value for the metric corresponding to the system specification j, in achieving the customer need δ. The second one contains the normalized quantifiers, $\beta_{jk}(t_0)$, $j \in \{1, ..., p\}, k \in$

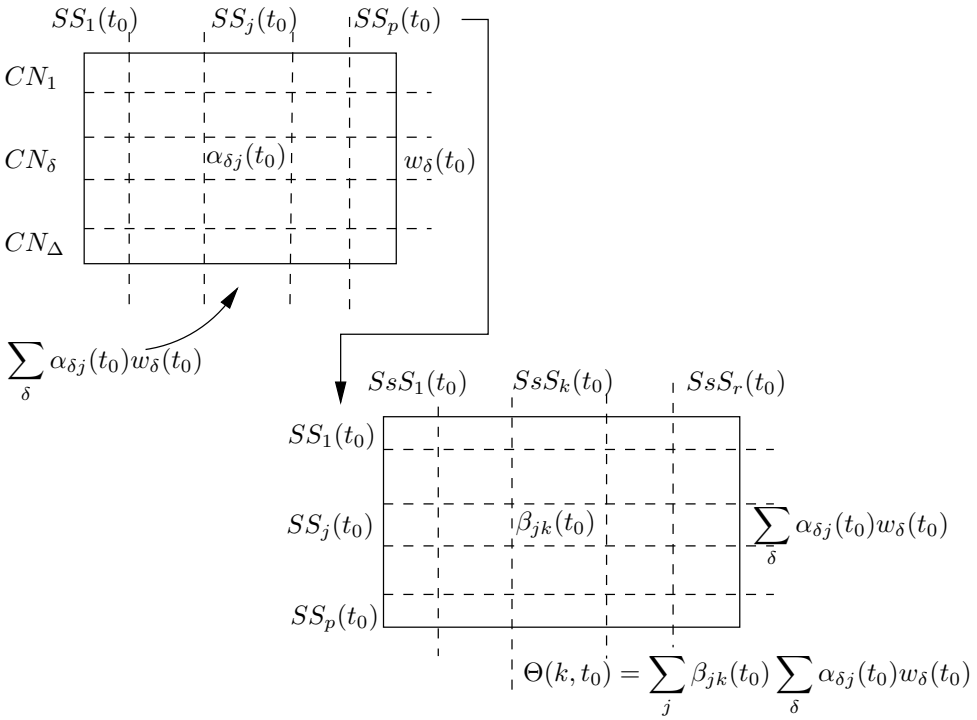

Figure 2.7. QFD waterfall for the above specification tree

$\{1, ..., r\}$, for the contribution of the maximal target value for the metric corresponding to the subsystem specification k, in achieving the maximal target value for the metric corresponding to the system specification j (thus one level down). Now, using both charts, we can compute a normalized quantifier for the contribution of the maximal target value for the metric corresponding to the subsystem specification k (i.e. design task k), in achieving the customer need δ is given by $\sum_{j=1}^{p} \beta_{jk}(t_0)\, \alpha_{\delta j}(t_0)$. The maximal overall contribution of the design task k at t_0 (to all custom needs) is:

$$\sum_{j=1}^{p}\sum_{\delta=1}^{\Delta} \beta_{jk}(t_0)\, \alpha_{\delta j}(t_0)\, w_\delta(t_0) = \sum_{\delta=1}^{\Delta}\sum_{j=1}^{p} \beta_{jk}(t_0)\, \alpha_{\delta j}(t_0)\, w_\delta(t_0) = \\ = \sum_{\delta=1}^{\Delta} \left(\sum_{j=1}^{p} \beta_{jk}(t_0)\, \alpha_{\delta j}(t_0) \right) w_\delta(t_0).$$

It cumulates the contributions of the maximal target value for its corresponding metric in achieving the maximal target values for the metrics corresponding to the system specifications.

Also, a more detailed QFD chart relates directly the customer needs to design tasks specifications, and consequently to the realization of each design task up to its maximal

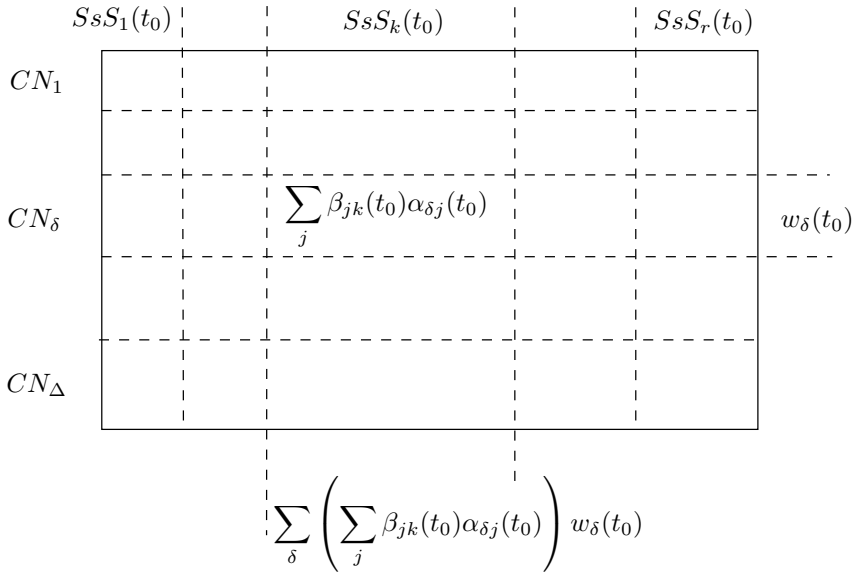

Figure 2.8. Customer needs/design tasks QFD chart

performance level (see Figure 2.8) via $\sum_{j=1}^{p} \beta_{jk}(t_0)\,\alpha_{\delta j}(t_0)\,, \delta \in \{1, ..., \Delta\}$ quantifiers.

The project has a market payoff structure:

The design task contribution functions are useful for maximizing the fulfillment of the customer needs subject to technological and cost constraints. For each customer need, we construct a *time dependent cumulative contribution function* of the performance levels of the design tasks belonging to it in the flow-down specification process. We obtain it by *adding the current design tasks contribution functions* to that customer need.

There are different levels of customer's satisfaction for the achieved new product, function of the distance between the cumulated and the ideal value for each of the customer's needs. We measure the expected market value for each cumulated value of a customer need achieved at the deadline through what we call a *market payoff function.* For the new products which have to fulfill several customer needs, their market value is described with *cumulative market payoff functions*. In literature we encounter different $S-$shaped curves for the one-customer-need and/or cumulative market payoff functions: general (Yoshimura, 1996; Huchzermeier and Loch, 2001), linear (Askin and Dawson, 2000), etc.

For detailed planning level optimization purposes, we may derive design task market payoff functions. In the linear case, these functions coincide with the design task contribution functions (see Section 3.4 and Appendix).

2.3.4 The technological uncertainties:

Too much abstraction can cause a lack of information, but too much detail makes the model solution intractable. So, as (Choi and Lee, 2001), (Reibman, 1990) and (Buzacott and Shanthikumar, 1980) suggest, we focus on the key features of the system. Since the

purpose of this framework is to be a basis for further computational models, our choices are based also on the fact that most of the work on performance modelling relies on the operational research techniques of queueing theory.

- *a time dependent arrival rate of new activities:* during a review period, new activities arrive to the design tasks in progress. According to cognitive psychology, they appear as a result of the incapacity of knowing beforehand all activities needed for performing a design tasks, so we model them to have preemptive resume priority over the planned activities (see (Best, 1995) pp.439). In later stages, the time dependent arrival rate decreases. In order to ensure the mathematical tractability of such a technological uncertainty we assume in the next chapters that the arrival process is Poisson.

- *a time dependent arrival rate of unplanned design tasks:* the integration of the knowledge created by the interaction in between the three considered NPD phases into a coherent product definition may add/delete design tasks from the project structure (Tatikonda and Rosenthal, 2000b; Tatikonda and Rosenthal, 2000a; Oorschot, 2001). Their deletion is modelled by allowing the controller to set their target performance level to zero, if the current minimal performance level is zero. For reasons of mathematical tractability, their addition is modelled by assuming general Markovian review period-dependent arrival processes of unplanned design tasks. Thus, each arrival process consists of design tasks concurrent either with design tasks to be allocated to the team at the beginning of the current review period, or with those allocated in previous NPD project review periods. We recall from Subsection 2.3.2 that to each review period there corresponds a stage in the network of precedence relationships. For the same reasons of mathematical tractability, each such newly arrived task concurrent with tasks of the stage t is to be performed only after all tasks of the stage $t-1$ are finished and before any task of the stage $t+1$ or newly arrived and concurrent with the ones in stage $t+1$ start to be solved.

  As seen in real life (see (Oorschot, 2001)), their arrival rate decreases in time. In order to use results already available in the class of queueing models, the unplanned design tasks arrived during one review period are assumed to have a common performance level structure, and an identical value function. Thus, for each period, instead of having an arrival of different types of design tasks, with different value functions, we take into account only an "average" type of task for that review period, assuming statistical identical design tasks. They are associated to customer's need transformation into product specifications at the beginning of the next review period, and then their contribution functions are calculated similarly as for the initial design tasks.

2.3.5 The state updating information (given at the end of each review period):

- for each already allocated design task, the performance level already achieved, and, per level, the number of remaining planned activities;

- for the sequences of newly arrived design tasks (statistically identical): their cardinality, their common number of activities per performance level, their common value function, and their precedence relationships as described above.

Given the previous definitions of the model elements, let us look at the control model we propose. The project horizon is split into a number of review periods. The scheme of the control structure is given in Figures 2.9 and 2.10. It has a total number T of review periods. Review period t starts at time t and ends at time $t + 1$.

At the start of each review period the state of the project is reviewed in order to incorporate the new information about the customer needs (from the market), and about the progress the engineers made in working on design tasks. In response to the new information, the control framework considers the process of solving design tasks by the engineers, the allocation of design tasks to engineers, and the updating of the performance levels of design tasks. The control framework furthermore assumes that decisions are taken at the aggregate decision level in order to maximize at the deadline the expected market value of the new product, given the resources remaining until project deadline, or a project abandonment. The last situation occurs when either the project exceeds its budget or cannot longer meet with a sufficiently high probability the minimal requirements regarding the product specifications. In the case of the continuation of the NPD, with a probability greater than a given safety margin, this new product will be delivered at the deadline, to the market.

At the aggregate decision level the performance levels of the design tasks are set, while at the detailed planning level a non-preemptive schedule is, generally, obtained for a relatively short planning horizon, e.g. two review periods. After updating the design task network structure, the project management may decide to decrease/increase the performance levels of some of the already scheduled design tasks, but not finished yet, due to the addition/deletion of design tasks and the limited capacity available versus their stochastic solving time. Then the design tasks and/or the number of their planned activities is changed. Since a feasible schedule is obtained by means of stochastic ordering, re-scheduling may occur.

2.4 Conclusions

The defined model elements are rich enough to incorporate the available knowledge from the relevant fields such as new product management and systems engineering, and still allow at each control level for the mathematical analysis of the process. In Chapter 3 we present the mathematical formulation of the hierarchical control framework, and discuss new or already available operations research methods and techniques that can be used for the project analysis, and for solving the detailed and aggregate decision level problems.

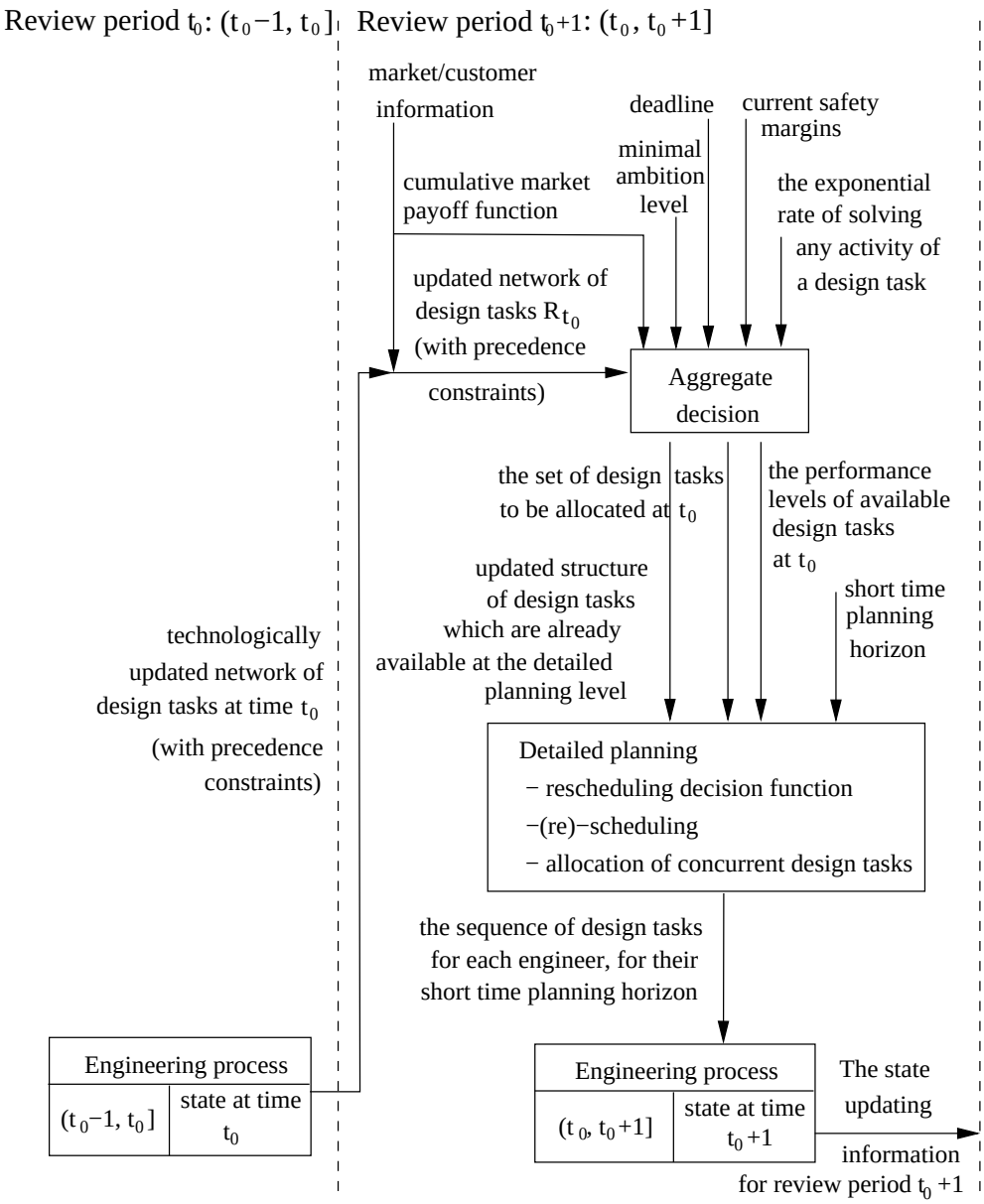

Figure 2.9. NPD control model

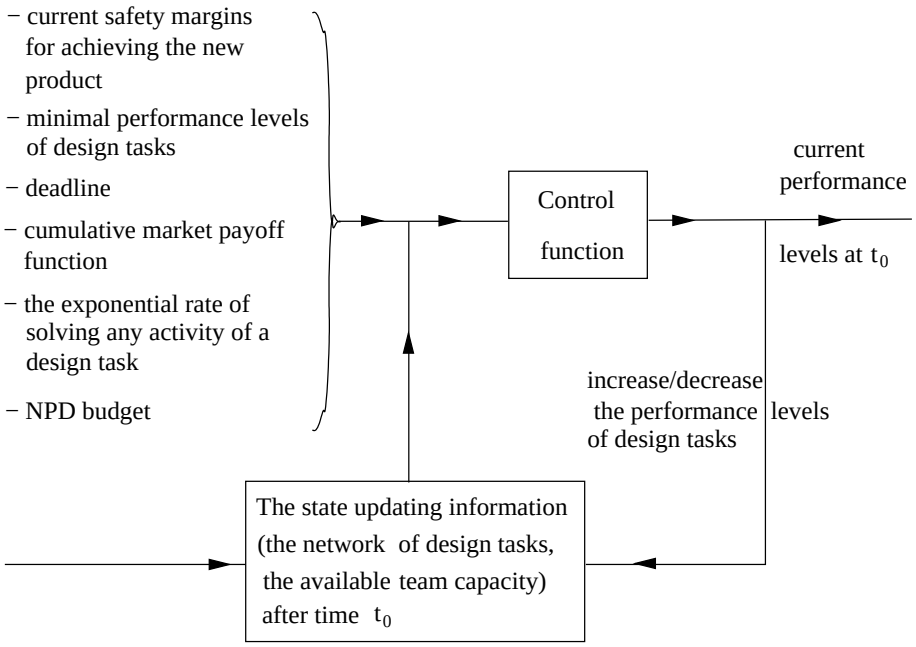

Figure 2.10. Aggregate decision at the beginning of review period $t_0 : (t_0, t_0 + 1]$

Chapter 3

Mathematical Modelling of the HCF

3.1 Introduction

In Chapter 2, Section 2.3 we first introduced a list of concepts, variables and relationships that enable the mathematical modelling of the process of performing the design tasks, of the dynamics in the design tasks that constitute the project, and of the relationship between expected market performance of a product and the performance levels of the design tasks of the project. These concepts, variables and relationships take into account relevant knowledge in published literature or new product management, product development, system engineering, project planning and control, production scheduling and human performance management. Afterwards, we introduced a hierarchical control framework (HCF) for the control of New Product Development (NPD) projects under a hard time constraint. Recall that the HCF has an aggregate decision level, a detailed planning level and an engineering level. The former deals with an overall evaluation of the NPD project. Recall also that time is divided in review periods. In this chapter, we first construct these control levels to cyclically operate for each review period individually (i.e. aggregate, detailed, engineering, then feedback and again aggregate, etc.). These constructions need a rigorous mathematical formulation: it makes things unambiguous and precise, and it enables us to see which parts have already been solved in the literature. In this chapter we thus also discuss the relationships of our constructions with well-known mathematical project models. This chapter therefore contributes as well to the area of mathematical models for the organization of work in an NPD, as to the development of management-related control concepts in the NPD projects, both areas that present research opportunities according to (Brown and Eisenhardt, 1995).

Moreover, in the next chapters we build upon this rigorous formulation in extending the model: since the aggregate decision level oversights the NPD project, it would clearly be more efficient if we were able to solve the decision process for multiple review periods, thus enabling the decision maker to foresee the outcome of the NPD project at the deadline. To this purpose we need carefully built heuristic models for the lower control levels; in Chapter 5 we thus present all the necessary models expressing the whole process seen from the aggregate decision level as a non-stationary Markov decision process, for which optimal policies can

be studied and computed.

This chapter is organized as follows. Section 3.3 describes the aggregate decision process of periodically setting the design tasks performance levels, given the update of the project status in view of the time and capacity remaining until the project deadline. Section 3.4 deals with the detailed planning process consisting of a re-scheduling decision function, its associated re-scheduling problem, and the allocation of design tasks to the engineers subject to workload constraints. In section 3.5 we present the engineering process that transforms the project state during each review period, by solving design tasks allocated to the engineers.

The aggregate decision process, as well as the detailed planning level allocation problem are specific for controlling NPD projects under a hard time constraint, and their formulation it is, to the best of our knowledge, new. The entire framework can be used to determine in probabilistic terms the expected NPD project outcome, as it is discussed in Section 3.6. It is not practical to separately solve for each review period the problems defined at the hierarchical structure levels, but the review periods can be easy linked using their approximate solutions. The chapter is concluded in Section 3.7.

3.2 Notation

The global variables of our NPD project model are:

T : the total number of review periods (review periods are numbered from 0 to $T-1$);

M : the total number of engineers;

$\overline{N}$: the initial number of design tasks;

N : an upper bound for the maximum number of design tasks during the whole project;

$L_{\max}(n)$: the number of performance levels of the initial design task n; $n = 1, ..., \overline{N}$;

Δ : the total number of customer needs considered;

h : the short time detailed planning horizon (i.e. a multiple number of review periods);

$c\,(n)$: the cost of performing one activity of the design task n; $n = 1, ..., N$

μ : the rate of the exponential distribution of an activity solving time.

Since N is an upper bound, we set to zero all the parameters depending on a virtual $n \in \{\overline{N}+1, \ldots, N\}$.

3.3 The aggregate decision process

In this section we construct the control level to operate for each review period individually. The aggregate decision problem formulation deals with both the technological uncertainty, and market requirement variability ((Huchzermeier and Loch, 2001); (Bhattacharya et al., 1998)).

A T-stage network of design tasks reflects the precedence relations among design tasks at the beginning of each review period (see Chapter 2 Subsection 2.3.2). The concept

of T-stage network naturally associates the $t-$th decision moment with the allocation of the $t-$th set of the partition of design tasks.

At the beginning of each review period t, $(t, t+1]$, the aggregate decision maker integrates in the NPD project the design tasks newly arrived during review period $t-1$.

We assume a general Markovian arrival process of statistically identical unplanned design tasks during review period $t-1$ (see Chapter 2 Subsection 2.3.4). These unplanned design tasks are concurrent with design tasks allocated/to be allocated during the i-th review period, with $i \in \{0, ..., t\}$. Being statistically identical means they have a common performance level structure, and an identical market payoff structure. Moreover, each such newly arrived task concurrent with tasks of the stage i is to be performed only after all tasks of the stage $i-1$ are finished and before any task of the stage $i+1$ or newly arrived and concurrent with the ones in stage $i+1$ start to be solved. Afterwards, the aggregate decision process itself takes place. We can briefly present it as follows.
Given at moment t

- the global variables of the NPD project (see Subsection 3.2)
- an updated network $R_t := (J(R_t), \mathcal{A}(R_t))$ of design tasks with their precedence relations
- for each design task n (planned or newly arrived) its maximum number of performance levels $L_{\max}(n,t)$, its minimum required performance level $l_{\min}(n,t)$ and its currently achieved performance level $l(n,t)$
- for each performance level l of each design task n, the number $N_a(n,t,l)$ of sequential activities planned to solve it, assuming all the previous levels already solved
- the current remaining budget $B(t)$ and the safety margins: $\alpha(t)$, to respect of the project deadline, and $\beta(t)$ not to exceed the maximal solving capacity of the team of engineers

Determine

- for each design task its target performance level, so as to maximize the expected market payoff

Subject to the following constraints:

1. the target performance level of each design task n is greater than $\min(l_{\min}(n,t), l(n,t))$, and smaller than $L_{\max}(n,t)$
2. the completion time of the NPD project defined by the design tasks targeted performance level is smaller than the remaining time until the deadline with a probability at least the current safety margin $\alpha(t)$.
3. the remaining total workload does not exceed the team remaining maximal solving capacity with a probability at least equal to the required $\beta(t)$ margin.
4. the total remaining cost for performing the design tasks up to their target performance level does not exceed the remaining budget $B(t)$.

Decisions on allocating design tasks to resources (engineers) are not considered at the aggregate decision level, but at the detailed planning one (see Section 3.4). Now, in order to rigorously formalize the setting, we need the following.

Input parameters (at the beginning of review period t):

$\alpha(t)$: the required current safety margin for the probability of completing the project before the deadline; $\alpha(t) \in (0,1)$

$\beta(t)$: the required current safety margin for the probability of exceeding the maximal team solving capacity; $\beta(t) \in (0,1)$

$B(t)$: the current remaining NPD project budget;

$R_t := (J(R_t), \mathcal{A}(R_t))$: the newly updated $T-$partite directed acyclic graph of unfinished design tasks precedence relations, where $J(R_t) := \Lambda_{t-1}^0 \cup \Lambda_{t-1}^1 \cup ... \cup \Lambda_{t-1}^{T-1} \cup \Omega_0(t-1) \cup \ldots \cup \Omega_t(t-1)$; $\Lambda_t^{t_0} = \Lambda_{t-1}^{t_0} \cup \Omega_{t_0}(t-1)$ forms the current design tasks set to be allocated at t_0, and $\Omega_i(t-1)$ is the set of newly arrived design tasks (during review period $t-1$) concurrent with design tasks allocated/to be allocated during the $i-$th review period; $\forall i = 0, ..., t$.

$L_{\max}(n,t)$: the current maximal number of performance levels of the design task n; $L_{\max}(n,t) = L_{\max}(n)$ for $n = 1, ..., \overline{N}$ and $L_{\max}(n,t) = 0$ if $n \notin J(R_t)$ (i.e. there is place reserved for design tasks not planned or not arrived yet up to the upper bound N but we set to zero the maximal performance level depending on such a virtual design task n);

$l_{\min}(\cdot,t) : \{1, ...N\} \to \mathbb{N}$: the minimal performance design task level function, where $l_{\min}(n,t) = 0$ if $n \notin J(R_t)$;

$l(\cdot,t) : 1, ..., N \to \mathbb{N} \cup \{-1\}$: the achieved performance design task level function, where $0 \leq l(n,t) \leq L_{\max}(n,t)$, for $n \in J(R_t)$ and by convention we define $l(n,t) = -1$ for $n \notin J(R_t)$;

$N_a(n,t,l)$: the number of sequential activities planned for solving the design task n, at the performance level l, assuming the previous levels already solved. All activities are assumed to have an exponentially distributed solving time with the same mean time μ (see Chapter 2), independent of the engineer which will perform them; $\forall\ n = 1, ..., N, \forall l = 1, ..., L$, where $N_a(n,t,l) = 0$ if $n \notin J(R_t)$;

Notation (at the beginning of review period t):

$N(t-1)$: the random variable giving the number of design tasks arrived since the NPD project beginning until the end of review period $t-1$, $[t-1, t)$;

$S_n(t,l)$: the solving time of the performance level l of the design task n, assuming the previous levels already solved. ; $n = 1, ..., N$. They are independent random variables Erlang-$(N_a(n,t,i), \mu)$;

$C(t, l(\cdot,t), R_t)$: the completion time of the network of design tasks, R_t, if $l(\cdot,t)$ gives the design tasks performance levels;

$M\left[t, N, l(n,t)_{1 \leq n \leq N}\right]$: the cumulated market payoff function, where $l(\cdot,t)$ gives the design tasks performance levels;

$\hat{l}(\cdot,t) : \{1, ..., N\} \to \{0, ..., L\}$: the target design task performance level function, at the beginning of review period t

Now, the aggregate decision problem can be mathematically formulated as follows:

Given

- the input parameters defined above

Determine the target design task performance level function $\hat{l}(\cdot,t)$ for each $n \in \{1,\ldots,N\}$ so as to maximize the cumulated market payoff function $M\left[t,N,\hat{l}(n,t)_{1\leq n\leq N}\right]$.
Subject to

$$L_{\max}(n,t) \geq \hat{l}(n,t) \geq \min\left(l_{\min}(n,t), l(n,t)\right),\ n=1,...,N \tag{3.1}$$

the completion time constraint

$$\Pr\left\{C\left(t,\hat{l}(\cdot,t),R_t\right) \leq (T-t)\right\} \geq \alpha(t) \tag{3.2}$$

the workload constraint

$$\Pr\left\{\sum_{\substack{n\in J(R_t)\\ \text{or}\\ \overline{N}+E[N(t-1)]+1\leq\ n\leq\overline{N}+E[N(T-1)],\text{ for } t>0}}\ \sum_{i=l(n,t)+1}^{\hat{l}(n,t)} S_n(t,i) \leq M\cdot(T-t)\right\} \geq \beta(t) \tag{3.3}$$

the budget constraint

$$\left[\sum_{\substack{n\in J(R_t)\\ \text{or}\\ \overline{N}+E[N(t-1)]+1\leq\ n\leq\overline{N}+E[N(T-1)],\text{ for } t>0}}\ \sum_{i=l(n,t)+1}^{\hat{l}(n,t)} N_a(n,t,i)\cdot c(n)\right] \leq B(t) \tag{3.4}$$

In (3.2), the analytical evaluation of a general directed acyclic graph completion time distribution is a $NP-$complete problem, but (Colajanni et al., 2000) gives a polynomial time algorithm for determining a tight upper bound.

In (3.3), the workload is computed by adding both the remaining solving times of the unfinished design tasks from R_t and the solving times of the average number of unplanned design tasks expected to arrive in the future review periods until the deadline of the project.

In (3.4), the cost is computed by summing for each remaining activity of each design task. The cost per activity may differ from one design task to another, refining the previous constraint.

This aggregate project planning formulation studies the project risk in terms of the probability of obtaining a total duration, a total quality, and a total cost, achieving dynamically the product definition. During the aggregate decision making, the decision maker has to assign numerical values to $\alpha(t)$, $\beta(t)$. Those choices depend on the risk that the controller is willing to take. The adoption of small values means that more time will be spent on the design tasks already allocated, since more uncertainty is allowed for the outcome of the NPD project. Adopting large values means precisely the opposite. A selection with a large $\alpha(t)$

and a small $\beta(t)$ means that the controller focuses more on the possibility of finishing the design tasks situated on most "stochastic critical paths ", than on the capacity issue.

For rigorously presenting several types of market payoff functions encountered in the literature we need more input parameters:

Market Payoff Related Input Parameters:

Δ : the total number of customer needs considered;

$w_\delta(t)$: the customer need normalized weight δ**;** $\forall\, \delta = 1, ..., \Delta$;

$\Theta(n, t, \delta)$: the normalized maximal contribution of the design task n in fulfilling the customer need δ; $\Theta(n, t, \delta) \in (0, 1]$; $\forall\, n = 1, ..., N, \forall\, \delta = 1, ..., \Delta$;

$\overline{S}_\delta(t, l(\cdot, t))$: the normalized $S-$function family giving the market payoff as function of the distance between the cumulated design tasks contribution per customer need δ, $\sum\limits_{n=1,..,N} \Theta(n, t, \delta) \cdot \frac{l(n,t)}{L_{\max}(n,t)}$, and the customer need ideal value given by the maximal target performance levels $L_{\max}(n, t)$ for each design task n; $\overline{S}_\delta((t, l(\cdot, t))) \in (0, 1), \forall \delta = 1, ..., \Delta$.

The market payoff function describes the expected market value of a new product which has to fulfill several customer needs. There are different levels of customer's satisfaction for the achieved new product, function of the distance between the cumulated and the ideal value for each of the customer's needs. An $S-$curve type of market payoff function gives an expected market value for each cumulated value of a customer need achieved at the deadline. As mentioned in Chapter 2 Section 2.3, different types of $S-$curve models and analytical cumulative market payoff functions are encountered in literature. The simplest one is $\sum\limits_{\delta=1}^{\Delta} w_\delta(t) \left[\sum\limits_{n=1,...,N} \Theta(n, t, \delta) \cdot \frac{l(n,t)}{L_{\max}(n,t)} \right]$ – the linear weighted additive one of (Askin and Dawson, 2000), while the most general function is the multiplicative one of (Yoshimura, 1996): $\prod\limits_{\delta=1}^{\Delta} \left[\overline{S}_\delta(t, l(\cdot, t))\right]^{w_\delta(t)}$.

3.4 The detailed planning process

The upper level of aggregate decision only sets the target performance levels for each task. Here we construct a sequence of design tasks for each engineer, from the older sequences and from the output of the aggregate decision process which took place at the beginning of the current review period. This data consists of: left-over design tasks (i.e. unallocated during the previous detailed planning process), unfinished design tasks already in the schedule, newly arrived design tasks — concurrent either with the ones unfinished or with the left over ones —, planned design tasks to be allocated at the beginning of the current review period, and newly arrived design tasks — concurrent with the ones to be allocated now. We shall distinguish between the planned design tasks for the current period together with the newly arrived tasks concurrent with them and, on the other hand, all the "old" design tasks (unallocated or unfinished) together with the newly arrived design tasks concurrent with design tasks from older review periods. The reason is that all these "older" design tasks (from the latter set) need to be finished as quickly as possible, while for the "current" ones (from the former set) the value function derived from market payoff maximization criteria can safely be applied. We thus have the following three step general procedure, at the beginning of any review period t, $(t, t + 1]$.

Step 1 — Re-scheduling decision function
Given

- an updated subnetwork of design tasks from the aggregate decision level (i.e. until the $t+1$ stage of the network),
- for each engineer m, the old sequence of design tasks from the previous detailed planning process,
- for each design task, its updated number of planned stochastic activities,
- the mean of the exponential distribution for activity solving times

Find all the tasks from these old sequences not respecting anymore the precedence relationships (i.e. the earliest start time of a design task should be greater or equal than the completion times of all its predecessors), remove them from the sequences and mark them as unscheduled in the network of tasks

Step 2 — (Re-)Scheduling of design tasks
Given

- the updated network from step 1
- for each engineer m, the updated old sequence $\sigma(m)^{\setminus}$ of design tasks from step 1,
- for each design task its updated number of planned stochastic activities,
- the mean of the exponential distribution for activity solving times

Schedule all the "old" design tasks (the unallocated ones, unfinished ones and the newly arrived concurrent with them) so as to minimize the expected makespan
Subject to the following constraints

- the precedence relationships from the updated network and from the updated old schedule (received from step 1)
- each design task is scheduled to exactly one engineer
- for each engineer, the "old" design tasks are scheduled at the end of the sequence from step 1.

Step 3 — Allocation of concurrent design task
Given

- the number of engineers with their optimal work-pressure level
- the closeness parameter δ, the mean of the exponential distribution for activity times
- the short time planning horizon h (used only to compute the engineers work pressure, see Subsection 3.4.3).
- for each engineer m, the updated old sequence $\sigma(m)^{\setminus\setminus}$ from step 2,
- the "current" design tasks (the planned ones for the current review period and the newly arrived concurrent with them) with their value function and number of planned stochastic activities

Allocate these "current" design tasks so as to maximize the value function of the allocated design tasks, and *set* for the allocated design tasks their due-date to $t + h$ (t being the current review period).
Subject to following constraints

- each design task is either allocated to exactly one engineer or is not allocated at all
- resources (engineers) have to be used close to their corresponding work pressure levels
- for each engineer, the current design tasks are allocated at the end of the sequence from step 2.

The design tasks left over after the allocation procedure will be further available at the detailed planning level. No due date will be set for them. This way of doing detailed planning reflects the particular purpose of using due dates in NPD projects. Since the precedence relationships between tasks are quite loose, the due dates are not mostly given to be met, but to ensure the efficiency of the involved engineers, as we further explain in Subsection 3.4.3, based on organizational psychology literature. So, while the deadline of the whole NPD project is a hard constraint, design tasks due dates are not.

An important remark concerning the optimality criteria of the allocation problem is that the detailed planning level controller does not know the market payoff function, nor the customer needs. For optimization purposes we then derive *design task value functions*, as an indication of design task realizations influence on the market payoff function. They are obtained by taking into account both the *maximal contribution of each design task* n, *in achieving each customer need* δ, and the type of *cumulative market payoff function* (see with the notation below the Appendix).

In order to formally present the setting, we first precisely define the input parameters, and thereafter we formally describe each of above mentioned steps in greater detail.

Input parameters (at the beginning of review period t):

$\alpha(m)$: the optimal work pressure level for the engineer m, for all $m = 1, ..., M$.

ε : the closeness parameter, i.e. the allowed variation with respect to the optimal work pressure level of each engineer.

$\sigma_m^{\setminus}(t)$: the $m-$th engineer scheduled design tasks sequence from review periods prior to t, $m = 1, ..., M$;

$\Omega_i(t-1)$: the set of newly arrived design tasks (during review period $t-1$) concurrent with design tasks allocated/to be allocated during the $i-$th review period, $i = 0, ..., t$;

$Z(t) := \bigcup_{m=1,...,M} \sigma_m^{\setminus}(t)$: the set of design tasks that were already scheduled in previous review periods;

$Y(t) := \Omega_t(t-1) \cup \Lambda_t^t$:the set of concurrent (planned or newly arrived) design tasks to be allocated to the engineers at the beginning of review period t;

$G_t = (J(t) \cup Y(t), \mathcal{A}(G_t))$: the directed, acyclic graph of precedence relations among design tasks at the detailed planning level, where $J(t) := Z(t) \cup \left(\bigcup_{i=0}^{t-1} \Omega_i(t-1) \right)$;

$\hat{l}(n,t)$: the $n-$th design task performance level, as established at the aggregate decision level, $n \in J(t) \cup Y(t)$;

$N_a(n,t,l)$: the $n-$th design task's number of sequentially planned activities, if its performance level is l, assuming the previous levels already solved; $l = 1,...,\hat{l}(n,t)$, $n \in J(t) \cup Y(t)$; A design task that is in progress will start with the next activity not performed yet.

$V(n,t)$: the $n-$th design task value; $\forall n \in J(t) \cup Y(t)$;

Notation (at the beginning of review period t):

$S_n(t)$: the random variable denoting the $n-$th design task solving time; $\forall n \in J(t) \cup Y(t)$;

$\Omega(t-1) := \bigcup_{i=0,...,t-1} \Omega_i(t-1)$: the set of newly arrived design tasks (during review period $t-1$) concurrent with previously allocated ones;

$k|\sigma$: the real index (in the numbering of design task from 1 to N) of the $k-$th design task from the sequence σ, $\forall k = 1,...,|\sigma|$

$Pred(n)$: the set of direct predecessors of the design task n in the graph G_t, $n \in J(t)$

For an engineer, to solve a design tasks n, scheduled to him, means that he has to sequentially solve the list of activities planned for it at the beginning of the review period t, for each level up to $\hat{l}(n,t)$. Thus, we consider the solving time of any design task n as being a random variable distributed Erlang-$\left(\sum_{i=1}^{\hat{l}(n,t)} N_a(n,t,i), \mu\right)$.

3.4.1 Re-scheduling decision function

At the beginning of each review period t, $(t, t+1]$, the re-scheduling decision function decides whether for the updated design tasks in $Z(t) \cup \Omega(t-1)$ we already have a partial schedule, satisfying the precedence relations among the design tasks according to a stochastic ordering.

Definition 1 *Given two random variables X_1, X_2 with distribution functions F_{X_1}, F_{X_2} we say that X_1 is stochastically smaller than X_2 (denoted by $X_1 \leq_{stoch} X_2$) if $F_{X_1}(z) \leq F_{X_2}(z), \forall z \geq 0$. The stochastic ordering is a partial order relationship among random variables and distribution functions which is closed under multiplication and convolution.*

We define the earliest starting time of a design task as $E_{k|\sigma_m^{\setminus}(t)}(t) := \sum_{i=1}^{k-1} \left[S_{i|\sigma_m^{\setminus}(t)}(t)\right]$ and its completion time as $C_{k|\sigma_m^{\setminus}(t)}(t) := \sum_{i=1}^{k} \left[S_{i|\sigma_m^{\setminus}(t)}(t)\right]$, for any $k \in Z(t)$.

So if there exists m_0 and k_0 such that $E_{k|\sigma_m^{\setminus}(t)}(t) <_{stoch} \max_{stoch}_{j \in \left\{Pred\ \ k|\sigma_m^{\setminus}(t)\right\}} C_j(t)$ (i.e. the earliest starting time of the k_0 is stochastically smaller than the completion time of one of its predecessors) then the design task k_0 and all its successors are added to $J(t) \setminus Z(t)$ and removed from the sequences assigned for the engineers, and implicitly from $Z(t)$.

3.4.2 (Re-)Scheduling of design tasks

The project controller of a design team will schedule to the engineers the design tasks from the set $J(t) \setminus Z(t)$ updated by the re-scheduling decision function. This is done

mainly to minimize the expected makespan, because we want to avoid the delays that may occur due to the precedence constraints relating $J(t) \setminus Z(t)$ to the set of concurrent planned design tasks to be allocated next to the engineers.

As a result of solving the scheduling problem, we add for each engineer m an optimal sequence $\sigma_m^{\setminus\setminus}(t)$ of design tasks, after its previous existing sequence $\sigma_m^{\setminus}(t)$ of the design tasks on progress from previous control periods. For the design tasks in the sequence $\sigma_m^{\setminus}(t)$ the precedence relationships respect was already checked by the rescheduling decision function (see Subsection 3.4.1). Thus, assuming that the engineer cannot work at more than one design task at the same time and that no preemption is allowed, our (re-)scheduling problem can be formulated as a stochastic identical parallel machine scheduling problem, with non-unit jobs, and arbitrary precedence relationships. Its minimization criterion is the expected maximum completion time. The precedence relationships are given by the directed, acyclic subgraph of G_t spanned by $J(t) \setminus Z(t)$.

Such a problem was solved analytically only for tree-like precedence constraints (see for a review (Weiss, 1995)), while the research of (Foulds et al., 1991) and (Neumann and Zimmerman, 1998)) gives polynomial heuristics even for more general types of precedence constraints (i.e. stochastic precedence relationships).

Denoting by $\sigma_m^{\setminus\setminus}(t)$ the sequence of design tasks from $J(t) \setminus Z(t)$ scheduled for the engineer m in the beginning of review period t ($m = 1, ...M$), the detailed planning level (re-)scheduling problem can be formulated as follows:
Minimize the makespan:

$$\min_{\sigma_1^{\setminus\setminus}(t),...,\sigma_M^{\setminus\setminus}(t)} \left\{ \max_{m=1,...,M} E\left[C_{|\sigma_m^{\setminus\setminus}(t)|\ \sigma_m^{\setminus\setminus}(t)}(t) \right] \right\} \tag{3.5}$$

Subject to $\forall k \in 1, ..., \left|\sigma_m^{\setminus\setminus}(t)\right|,\ \forall m = 1, \ldots, M$
precedence relationship respect (i.e. a task only starts after all its predecessors all solved)

$$E_{k|\sigma_m^{\setminus\setminus}(t)}(t) \geq \max_{stoch} \left\{ C_j(t),\ j \in \textit{Pred}\left(k|\sigma_m^{\setminus\setminus}(t)\right) \right\} \tag{3.6}$$

a task is scheduled to an unique engineer

$$\forall n \in J(t) \setminus Z(t), \exists! m \in 1, ..., M \text{ s. t. } n \in \sigma_m^{\setminus\setminus}(t) \tag{3.7}$$

where the completion and respectively the earliest starting time of a design task are given by:

$$C_{k|\sigma_m^{\setminus\setminus}(t)}(t) := \sum_{i \in \sigma_m^{\setminus}(t)} \left[S_{i|\sigma_m^{\setminus}(t)}(t) \right] + \sum_{i=1}^{k} \left[S_{i|\sigma_m^{\setminus\setminus}(t)}(t) \right] \tag{3.8}$$

$$E_{k|\sigma_m^{\setminus\setminus}(t)}(t) := \sum_{i \in \sigma_m^{\setminus}(t)} \left[S_{i|\sigma_m^{\setminus}(t)}(t) \right] + \sum_{i=1}^{k-1} \left[S_{i|\sigma_m^{\setminus\setminus}(t)}(t) \right] \tag{3.9}$$

3.4.3 Allocation of concurrent design tasks

After the (re-)scheduling, the design team project controller allocates the set $Y(t)$ of concurrent (planned or newly arrived) design tasks to engineers.

In order to investigate the problem more precisely, we need to model the dependency between engineers' productivity and their perceived work pressure. In (Bowers et al., 1997) a first estimation of the work pressure is given by comparing, for a given engineer, the estimated duration for completing the allocated design tasks and the available time until their due date. For this reason, we introduce the notion of a common short planning horizon h for all the engineers. We recall that for all the allocated design tasks it was set a common due date $t + h$. Thus, we use the probability $\overline{p}_h(A, t)$ of finishing all design tasks of the set $A \in \mathcal{P}(Y(t))$ in time h. Therefore, we achieve efficiency for the engineers involved in the process by requiring for each engineer m that $|\overline{p}_h(A_m, t) - \alpha(m)| < \varepsilon$, where $\alpha(m)$ is her/his optimal work pressure level. In words, this requires that for a short time horizon h, the probability of finishing the design tasks allocated to m is close to an optimal subjective value. This ensures both not overloading the engineers and not giving them too little work to do. Organizational psychology (see for a review (Wickens and Hollands, 1999)) shows that this dependency between the productivity and the work pressure is curvilinear (concave), and this is the reason of the absolute value bound. The impact of work pressure on engineers productivity has been confirmed by empirical research in (Oorschot, 2001).

The allocation of concurrent design tasks to engineers, as sketched in the beginning of Section 3.4 (i.e. Step 3), is a multiple choice knapsack problem specific for NPD projects under hard time constraint and was solved in (Dragut, 2002) (see Chapter 4). The stochasticity and the fact that to each engineer we can assign more than one design task, allow neither a mixed-integer formulation of the problem, nor a simpler formulation of the partial solutions to be eliminated, as required by more efficient algorithms ((Ibaraki et al., 1978); (Dyer et al., 1995)).

Remark 2 *The solving time $S_n(t)$ of an arbitrary design task $n \in A \in \mathcal{P}(Y(t))$ is a sum of i.i.d. exponential random variables with mean μ. Since the mean μ is the same for the activities of all tasks, the solving time of the entire set A of design tasks is Erlang$(|A|, \mu)$ distributed. Thus the inequality $|\overline{p}_h(A_m, t) - \alpha(m)| < \varepsilon$ leads to a minimal and a maximal number of activities that can be performed by an engineer during the short-time planning horizon h. This is a supplementary reason to keep the design task's control at activity level.*

Under the assumption that we have enough design tasks to be allocated to the engineers, the allocation problem can be written as:

$$\max_{\pi_{M+1}=(A_1,A_2,\ldots,A_{M+1})\in\Pi_{M+1}} \left(\sum_{m=1,\ldots,M} \sum_{n\in A} V(n,t) \right) \qquad (A0)$$

where Π_{M+1} is the set of all $(M+1)$-partitions $\pi_{M+1} = (A_1, A_2, ..., A_{M+1})$ of the set of design tasks Y which, for any engineer i with $i \in \{1, \ldots, M\}$, satisfy the optimal work pressure level condition $|\overline{p}_h(A_i) - \alpha(i)| < \delta$, and gather the left-over design tasks in the last component A_{M+1} of π_{M+1}.

Starting from dynamic programming techniques, in Chapter 4 we propose a solution to this allocation problem. First we formulate this problem as a discrete deterministic dynamic-programming problem. This formulation ensures the existence of an optimal solution, and creates a graph structure of the problem space. Afterwards, we avoid the exhaustive search of the problem space by constructing a heuristic evaluation function of A* type for a best-search algorithm. This function is based on aggregate information on the design task

set. We discuss the efficiency of our algorithm, and we prove that it finds an optimal feasible allocation, provided that there exists a feasible allocation.

If after solving the aggregate level problem, there are not enough design tasks to be allocated to engineers, the engineers may receive design tasks from other projects to ensure their efficiency. From this project point of view they will have zero-value, and hence will be the last to be allocated.

3.5 Engineering process

During the review period t, $(t, t+1]$, the engineers work concurrently, each on its corresponding sequence of design tasks determined at the beginning of review period t in the detailed planning. Each design task has been allocated to only one engineer.

Input parameters (at the beginning of review period t):

$\sigma_m(t)$: the sequence of design tasks allocated to the engineer m during the detailed planning; $m = 1, ..., M$;

$\Upsilon(t) := \bigcup_{m=1}^{M} \{k | \, k \in \sigma_m(t)\}$: the set of all design tasks allocated to the team of engineers;

$N_a(n, t, l)$: the number of activities planned for solving the design task n, at the performance level l, assuming the previous levels already solved; $l = 1, ..., L$, $n \in \Upsilon(t)$;

$\hat{l}(n, t)$: the level at which the design task n must be performed, as established at the aggregate decision level, $n \in \Upsilon(t)$;

$\lambda(t)$: the review-period dependent Poisson arrival of unplanned activities for all the design tasks allocated to the engineers ($\lambda(t)/\mu < M$).

In real life, each engineer of the team of M engineers has to perform the tasks allocated to her/him by an NPD project control process, according to a certain priority order. As discussed in Section 3.4, this order depends on the precedence relationships structure of the NPD project, on the optimization criteria considered at the detailed planning level, and on the type of market payoff function considered in the aggregate decision process. The sequence of design tasks allocated to an engineer may contain design tasks that can be performed in parallel and their order in a sequence reflects only the scheduler optimality criteria. However, unlike machines, human beings are able to perceive the concurrency of design tasks. It is reasonable to assume that during the execution process the engineers will not work all the time on the allocated design tasks in the sequential order established by the scheduler.

During the solving process of the design tasks, several other disturbances may occur, resulting in new activities for the design tasks in progress. Also, addition/deletion of design tasks in the NPD project may occur (see Chapter 2, Section 2.3). They appear as a result of the incapacity to foresee at the outset all activities needed to complete the design task. We model them to have preemptive resume priority over the planned activities.

All these uncertainties influence the execution of the schedule. Therefore, for the next period, at the aggregate decision level the decision maker takes into consideration the engineering level status at the end of the current review period. This fact is consistent with real life situations where, on a weekly basis, each engineer measures how much time was spent on solving each design task, what activities were solved, what activities were added.

Thus, at the beginning of review period $t+1$, the output variables from the engineering process are:

$N_a(n, t+1, l)$: the number of activities of design task n, if design task n is performed at level l, assuming the previous levels already solved; $l = 1, ..., L$, $n \in \Upsilon(t)$;

$l(\cdot, t+1) : \{1, ..N\} \rightarrow \{0, ..., L\}$: the achieved performance design task level function;

3.6 Expected NPD project outcome

Before starting a new review period, the aggregate decision maker must update the previous network of design tasks according to the technological changes occurred (deletion, addition) of (unplanned) design tasks, arrival of new activities for the design tasks on which the engineers have worked). In this chapter in Section 3.3 we have formulated the aggregate decision process for one review period. Based on the previous mathematical descriptions, one can link the review periods using approximate solutions for the detailed planning and engineering process level problems, and assuming review period-dependent Poisson arrival processes of unplanned design tasks with a common contribution function and identical performance level structure.

In Chapter 5, after discussing how simple priority rules may be used to schedule detailed planning level design tasks, we propose a simple queueing model to describe the working behavior of an engineer/team of engineers in a NPD environment. We consider that at the beginning of each stage t the linkage problem gives a queueing system with M parallel servers, and a common queue of $\tilde{N}_a(t) = \sum_{n \in Y(t) \cup Z(t)} \sum_{l=l(n,t)+1}^{l(n,t+1)} N_a(n, t, l)$ planned activities with $Exp(\mu)$ distributed processing times. The solving process is disturbed by a $\lambda(t)$-Poisson arrival of unplanned activities ($\lambda(t)/\mu < M$), having preemptive resume priority over the planned activities. By simplifying both the engineering, and the detailed planning processes, this model is further enriched and used to compute the transition probabilities of a non-stationary Markovian decision process model of the multi-period aggregate decision problem (see Chapter 5). Thus, the aggregate decision process is extended from one-review period to the multiple review period horizon. The Markov model can be further used to predict the expected NPD project outcome in terms of market payoff, and to derive optimal policies for achieving it.

3.7 Conclusions

Managing the new product definition is a complex managerial task. Based on recent research, this chapter proposes a mathematical formulation of the NPD project management, which includes two new problems specific for the NPD projects under hard time constraint.

We model the real-life project, and formalize the quality-time-cost trade-offs underlying the NPD project mainly from the technological uncertainty point of view. This chapter's goal is to close the gap between the mathematical models world and management world, providing a basis from which one can derive the constraints for computational models. Thus, it facilitates the evaluation and acceptance of the computational results by managers.

Part II

Analytical Solutions

Chapter 4

NPD Design Tasks Allocation

4.1 Introduction

Allocation problems appear in many decision making situations like the allocation of tasks to resources, of workers to jobs, of salespeople to regions, or of requirements to suppliers. The problem definition and, in particular, the constraints depend on the purposes of allocation ((Atan and Pandit, 1996), (Diks and Kok, 1998), (Romeijn and Morales, 2000)). The allocation problems involving people have recently received increased attention of researchers, who developed optimal or heuristic allocation techniques for various real-life situations ((Abboud et al., 1998), (Bossert, 1998), (Reeves and Reid, 1999), (Haluk, 2000)). The difficulty of these problems resides in both their combinatorial nature and in the diversity of the real-life factors that have to be included in the model.

Based on the model choices from Chapter 3, Section 2.3, and Section 3.4 our allocation problem can be stated as follows:
Given

- a finite number of design tasks, with their value function, and number of planned stochastic activities i.i.d. $Exp(\mu)$
- a finite number of engineers, with their optimal work pressure level
- the closeness parameter δ, the mean of the exponential distribution for activity times μ and the short time planning horizon h

Determine the allocation of design tasks to engineers

- *in order to maximize the value function of the allocated design tasks*

Subject to the following constraints:

- each design task is either allocated to exactly one engineer or is not allocated at all, and resources (engineers) have to be used close to their corresponding optimal work pressure levels.

The remainder of this chapter is structured as follows. In Section 4.2 we state in a mathematical way the problem and prove that it can be formulated as a discrete deterministic dynamic-programming problem. This formulation ensures the existence of an optimal solution; through a graph structure of the problem space, it also allows the finding of an optimal solution without an exhaustive search of the entire problem space. This graph structure can actually be searched with much less computational effort with heuristic search algorithms. Next, in Section 4.3, we construct such a heuristic evaluation function of A* type (see Definition 7), using aggregate information (from the input data) on the design tasks set. Based on established results from heuristic search algorithms, we propose an A* type algorithm for solving the problem, with two possible implementations: standard best-first search (open and closed priority lists, see for example (Russell and Norvig, 1995), and RBFS implementation (Korf, 1993). We also prove that owing to the properties of the proposed heuristic, our algorithm is guaranteed to find an optimal cost allocation of the design tasks to engineers. For the implementations, we have a complexity trade-off: the RBFS implementation runs in linear space in the branching factor and the depth of the tree, and in linear time in the number of generated nodes, at the expense of revisiting some nodes. The standard best-first search implementation on the other hand is proven here to expand only a minimal number of nodes (with respect to any other algorithm from its class), yet the spatial complexity may be exponential: the branching factor elevated to the depth.

Experimental evidence of the tests we have performed shows however that the RBFS solution revisits very few more nodes (having thus a small supplemental running time with respect to the standard best-first), and the total number of visited nodes is very small compared to the cardinality of the search space (second kind Stirling numbers). A detailed discussion of these aspects concludes the chapter.

4.2 Problem formulation

In order to investigate the type of allocation problem more precisely, we recall from Chapter 3, Section 3.4 that cognitive psychology studies (see for a review (Oorschot, 2001)) show there is a curvilinear dependency between engineers' productivity and the time pressure perceived by them. In (Bowers et al., 1997) a first estimation of the time pressure is given by comparing, for a given engineer, *the estimated duration for completing the allocated design tasks* and *the available time until the deadline*.

As described in Chapter 3, Section 2.3, our design task solving times are stochastic, and there is a total available time h for all engineers (also called *short time planning horizon*, and given as input); thus we use the probability $\overline{p}_h(A)$ of finishing all design tasks of a set A in time h. We also assumed in Chapter 3, Section 2.3 the existence (i.e. input data) for each engineer m of an optimal value $\alpha(m)$ for this probability, called *optimal work pressure level*. Therefore, we achieve efficiency for the engineers involved in the process by requiring for each engineer m that $|\overline{p}_h(A_m) - \alpha(m)| < \delta$. In words, this requires that for a short time horizon h, the probability of finishing the design tasks allocated to m is close to an optimal subjective value (the closeness δ being also an input). This ensures both not overloading the engineers and not giving them too little work to do.

The notation used in this chapter is a simplification of the notation from Chapter 3, Section 3.4. The references to the target performance level or to the time moment were removed from the notation. The current target performance levels are established during the

aggregate decision level, so they cannot be influenced here. The allocation problem occurs at the beginning of each review period, and we assumed that an approaching deadline of the entire NPD project will not influence it.

Input parameters:

h : the short time planning horizon;

Y : the set of design tasks that have to be allocated; $K := |Y|$;

M : the number of engineers;

$\alpha(m)$: the optimal workload level for the engineer m, for all $m = 1, ..., M$;

δ : the closeness parameter, i.e. the allowed variation with respect to the optimal workload level of each engineer;

μ : the rate of the exponential solving time for each activity (of any design task);

$V(n)$: the value function of the design task n; $V(n)$ is a real positive number, for all $n \in Y$;

$N_a(n)$: the total number of activities of the design task n; for all $n \in Y$;

Definition 3 *Given $Y \neq \emptyset$, $|Y| \geq m$, we say that $\pi_m = (A_1, A_2, ..., A_m)$ is an $m-$partition of the set Y if $\bigcup_{i=1}^{m} A_i = X$, $A_i \cap A_j = \emptyset$ and $A_i \neq \emptyset$ for all $i \neq j$, $i, j = 1, ..., m$.*

Notation:

S_n: the solving time of the design task n.

$\sigma : \{1, ..., M\} \to \mathbb{R}$, where $\sigma(i) = \sum_{j=i+1}^{M} (\alpha(j) + \delta)$, for $i \in \{1, ..., M-1\}$, with $\sigma(M) = +\infty$.

$v(A) := \sum_{n \in A} V(n)$: the cumulated value of the design task set $A \in P(Y)$; $u(\emptyset) = 0$;

$\overline{p}_h(A) := \Pr\{\text{Solving time}(A) < h\} = \Pr\left\{\sum_{n \in A} S_n < h\right\}$: the probability of solving the design tasks in time; $A \in P(Y)$;

Π_{M+1}: the set of all $M+1$-partitions π_{M+1} of the set of design tasks Y which, for any engineer i with $i \in \{1, \ldots, M\}$, satisfy the optimal work pressure level condition $|\overline{p}_h(A_i) - \alpha(i)| < \delta$, and gather the left-over design tasks in the last component A_{M+1} of π_{M+1}.

The solving time $S_n(t)$ of an arbitrary design task $n \in A \in \mathcal{P}(Y(t))$ is a sum of i.i.d. exponential random variables with mean μ. Since the mean μ is the same for the activities of all tasks, the solving time of the entire set A of design tasks is Erlang$(|A|, \mu)$ distributed. Thus the inequality $|\overline{p}_h(A_m, t) - \alpha(m)| < \varepsilon$ leads to a minimal and a maximal number of activities that can be performed by an engineer during the short-time planning horizon h.

Under the assumption that we have enough design tasks to be allocated to the engineers, the optimization problem is defined as follows:

$$\max_{\pi_{M+1}=(A_1,A_2,...,A_{M+1}) \in \Pi_{M+1}} \left(\sum_{m=1,...,M} v(A_m) \right) \qquad (A0)$$

In what follows, we reformulate the problem $(A0)$ from a maximization to minimization problem by subtracting the objective function from a strict upper bound of it. This

allows us to associate to the new problem, called A, a dynamic programming graph with all the arc costs being positive numbers. Such a construction further enables us to find an A^*-type heuristic since such an heuristic has to have positive values (see Definition 7).

Proposition 4 *With the above notation, the problem* $(A0)$ *is equivalent to the minimization problem:*

$$min_{\pi_{M+1}\in\Pi_{M+1}}\overline{C}\left(\pi_{M+1}\right) \qquad (A)$$

where $\overline{C}\left(\pi_{M+1}\right)=\theta-\sum\limits_{m=1,...,M}v\left(A_m\right)$, *with* $\theta=\theta_0\cdot K$ *for* $K=|Y|$, *and with* $\theta_0=(1+\sup_{k\in Y}V\left(k\right))$.

Proof. The function $\overline{C}$ is correctly defined on Π_{M+1}, because θ_0 exists (finite number of tasks, each with a finite value function). The equivalence of the objective functions holds because we have that

$$\widetilde{\pi}_{M+1}=\left(\widetilde{A}_1,...,\widetilde{A}_{M+1}\right) \text{ is an optimal solution of } (A0)\Leftrightarrow$$

$$\theta-\sum_{m=1}^{M}v\left(\widetilde{A}_m\right)\leq\theta-\sum_{m=1}^{M}v\left(A_m\right) \text{ for any } \pi_{M+1}\in\Pi_{M+1}\Leftrightarrow$$

$$v\left(Y\right)-v\left(\widetilde{A}_{M+1}\right)\geq v\left(Y\right)-v\left(A_{M+1}\right)\Leftrightarrow$$

$$v\left(\widetilde{A}_{M+1}\right)\leq v\left(A_{M+1}\right)\Leftrightarrow$$

$$\sum_{m=1}^{M}v\left(\widetilde{A}_m\right)=v\left(Y\right)-v\left(\widetilde{A}_{M+1}\right)\geq v\left(Y\right)-v\left(A_{M+1}\right)=\sum_{m=1}^{M}v\left(A_m\right)\Leftrightarrow$$

$$\widetilde{\pi}_{M+1} \text{ is an optimal solution of } (A).$$

■

4.2.1 Dynamic programming formulation of the reduced problem (A)

The DP approach for discrete optimization problems (such as planning, scheduling, knapsack, etc.) has already been studied in the literature. The dynamic programming (DP) formulation views the problem space as a directed graph with weighted arcs. This in turn avoids an exhaustive search of the initial space, which, for many discrete optimization problems of practical interests, is huge. The basic DP methods can be improved through various node selecting and pruning techniques, or through mixing them with other search methods. We can mention (Dyer et al., 1995), (Martello et al., 2000), (Klamroth and Wiecek, 2000). Also, many heuristic search methods can be used on this graph constructed by the DP formulation, to search it much more efficiently (as (Grama and Kumar, 1995), for example). This is the reason we first focus on the DP formulation of our problem, building a search space used in the subsequent section, where we solve the problem through best-first search algorithms.

In some sense, the design task allocation problem we focus on is also linked to multiple choice knapsack problems (studied in some of the above cited references). However, for our problem, the stochasticity and the fact that to each engineer we can assign more than one design task changes the setting quite substantially. The first feature of the problem does not allow a mixed-integer formulation of the problem, while the second one changes

dramatically the structure of partial solutions to be eliminated. Under the condition that to each engineer we can assign one and only one design task, the concepts of IP or DP infeasibility and dominance allow the development of more efficient algorithms as in (Dyer et al., 1995), and (Ibaraki et al., 1978).

Many authors have proposed different formulations of DP ((Kumar and Kanal, 1988), (Ghalil and Park, 1992), (Eppstein et al., 1988), (Yao, 1980)). Following the (Grama and Kumar, 1995) classification, the dependencies between subproblems in DP formulations separate them in serial and non serial ones. The serial DP formulations are such that the solution of a subproblem is constructed only from solutions of subproblems immediately preceding the considered ones, and this is the type of our formulation as well.

Proposition 5 *The (A) problem can be formulated as a finite-horizon discrete deterministic dynamic problem.*

Proof. Formulating, from an additive cost function, an optimization problem as a discrete deterministic dynamic problem with $(M+1)-$stages is equivalent to constructing an oriented graph (Y,T) called sequential or $(M+1)-$stage graph and to defining the corresponding costs C.

Let $*$ be a special symbol, denoting the finding of a solution. We define

$$X_0 = \{\emptyset\}, \quad B_0 = \emptyset$$

$$X_i = \left\{ (B_{i-1}, A_i) : \begin{array}{c} B_{i-1} \in \mathcal{P}(Y) \\ A_i \in \mathcal{P}(Y \setminus B_{i-1}) \\ |\overline{p}_h(A_i) - \alpha(i)| < \delta \\ \overline{p}_{(M-i)h}(Y \setminus (B_{i-1} \cup A_i)) < \sigma(i) \end{array} \right\} \text{ for } i \text{ s.t. } 1 \leq i \leq M-1$$

$$X_M = \left\{ (B_{M-1}, A_M) : \begin{array}{c} B_{M-1} \in \mathcal{P}(Y) \\ A_M \in \mathcal{P}(Y \setminus B_{M-1}) \\ |\overline{p}_h(A_M) - \alpha(M)| < \delta \end{array} \right\}$$

$$X_{M+1} = \{*\}$$

Thus, each set X_i represents possible allocations for the engineer i, when B_{i-1} contains the design tasks already allocated to the previous engineers (from 1 to $i-1$). The last condition for the X_i (with $i < M$) ensures there are enough design tasks left in $Y \setminus (B_{i-1} \cup A_i)$, so that a $\delta-$close optimal work pressure can be achieved also for the subsequent engineers (from $i+1$ to M). We recall that by $\sigma(i)$ we have denoted the sum $\sum_{j=i+1}^{M} (\alpha(j) + \delta)$ This is so because what we want is actually

$$\forall j \in \{i+1, ..., M\}, \quad |\overline{p}_h(A_j) - \alpha(j)| < \delta, \text{ where } A_j \in \mathcal{P}(Y \setminus (B_{i-1} \cup A_i)) \quad (4.1)$$

The absolute value inequality contains actually two inequalities, and the one bounding from above the probability of finishing in time h is the interesting one. Because the engineers need to experience some pressure in order to be efficient, which means that the probability of finishing should be not too large.

The other inequality of the absolute value inequality (4.1) ensures that the probability is not too low, i.e. the pressure is not too big. Yet, this could be dealt with by removing some tasks, so this is not a necessary condition for the existence of a complete solution. Thus, the inequality (4.1) implies

$$\forall j \in \{i+1, ..., M\}, \quad \overline{p}_h(A_j) < \alpha(j) + \delta, \text{ where } A_j \in \mathcal{P}(Y \setminus (B_{i-1} \cup A_i))$$

By adding them, and by noting the independence of the solving times of the tasks from the A_j sets (which are disjoint), we obtain a probability of either one of the A_j task sets to be finished before h:

$$\Pr\left\{\bigvee_{j=i+1}^{M}\text{Solving time}\left(A_j\right)<h\right\}<\sum_{j=i+1}^{M}\left(\alpha(j)+\delta\right)=\sigma\left(i\right).$$

We notice that

$$\Pr\left\{\text{Solving time}\left(\bigcup_{j=i+1}^{M}A_j\right)<\left(M-i\right)h\right\}\leq\Pr\left\{\bigvee_{j=i+1}^{M}\text{Solving time}\left(A_j\right)<h\right\}$$

(because anytime the events of the left member occur, the ones on the right occur also). Since $\bigcup_{j=i+1}^{M}A_j\subseteq Y\setminus\left(B_{i-1}\cup A_i\right)$ we also notice that

$$\overline{p}_{(M-i)h}\left(Y\setminus\left(B_{i-1}\cup A_i\right)\right)\leq\Pr\left\{\text{Solving time}\left(\bigcup_{j=i+1}^{M}A_j\right)<\left(M-i\right)h\right\},$$

thus proving the last condition as necessary.

Further we define for each decision step the decision set and the transition function.

$$D_1\left(\emptyset\right)=\left\{A_1:\ A_1\in\mathcal{P}\left(Y\right),\left|\overline{p}_h\left(A_1\right)-\alpha\left(1\right)\right|<\delta,\right\},\ t_1\left(A_1\right)=\left(\emptyset,A_1\right)$$

For $\left(B_{i-1},A_i\right)\in X_i$, $B_i=B_{i-1}\cup A_i,\ i=1,...,M-1$ we define:

$$D_{i+1}\left(B_{i-1},A_i\right)=\left\{A_{i+1}:\begin{array}{c}A_{i+1}\in\mathcal{P}\left(Y\setminus B_i\right)\\\left|\overline{p}_h\left(A_{i+1}\right)-\alpha\left(i+1\right)\right|<\delta\\\overline{p}_{(M-i-1)h}\left(Y\setminus\left(B_{i-1}\cup A_i\cup A_{i+1}\right)\right)<\sigma\left(i+1\right)\end{array}\right\}$$

$t_{i+1}\left(\left(B_{i-1},A_i\right),A_{i+1}\right)=\left(B_{i-1}\cup A_i,\ A_{i+1}\right)=\left(B_i,A_{i+1}\right)$ where the set $A_{i+1}\in D_{i+1}\left(B_{i-1},A_i\right)$

For $\left(B_{M-1},A_M\right)\in X_M$ we define

$$D_{M+1}\left(B_{M-1},A_M\right)=\left\{A_{M+1}:\begin{array}{c}A_{M+1}\in\mathcal{P}\left(Y\setminus B_M\right)\\\left|B_M\cup A_{M+1}\right|=K\end{array}\right\}$$

and for $\left(B_{M-1},A_M\right)\in X_M$, $A_{M+1}\in D_{M+1}\left(B_{M-1},A_M\right)$ let us define

$$t_{M+1}\left(\left(B_{M-1},A_M\right),A_{M+1}\right)=\left\{*\right\}$$

For all $\left(B_{i-1},A_i\right)\in X_i,\ 1\leq i\leq M$ let us define

$$T\left(B_{i-1},A_i\right)=\left\{\left(B_i,A_{i+1}\right)\ :\ \begin{array}{c}\left(B_i,A_{i+1}\right)=t_{i+1}\left(\left(B_{i-1},A_i\right),A_{i+1}\right)\\A_{i+1}\in D_{i+1}\left(B_{i-1},A_i\right)\end{array}\right\}$$

and

$$T\left(B_{M-1},A_M\right)=\left\{*\right\}$$

The nodes connected by arcs are:
$i)$ $\emptyset$ and $(\emptyset, A_1) \in X_1$,
$ii)$ (B_{i-1}, A_i) and (B_i, A_{i+1}) if $(B_{i-1}, A_i) \in X_i$ and if $(B_i, A_{i+1}) \in T(B_{i-1}, A_i)$, with $1 \leq i \leq M-1$,
$iii)$ $(B_{M-1}, A_M) \in X_M$ and $\{*\}$.
The corresponding costs are:
$i)$ $C(\emptyset, (\emptyset, A_1)) = \theta_0 \cdot |A_1| - v(A_1) > 0$
$ii)$ $C((B_{i-1}, A_i), (B_i, A_{i+1})) = \theta_0 \cdot |A_{i+1}| - v(A_{i+1}) > 0,\ 1 \leq i \leq M-1$;
$iii)$ $C((B_{M-1}, A_M), \{*\}) = \theta_0 \cdot (K - |B_{M-1} \cup A_M|) + 1 > 0$ ■

By construction, thus, we have the following.

Corollary 6 *The problem of finding the minimal cost path in the $(M+1)-$stage weighted graph (Y, T) is equivalent to the problem of obtaining an optimal allocation π_{M+1} of the set of design tasks Y to M engineers in the sense of problem (A).*

Proof. One can easily see that the objective function of problem (A) can be expressed as the sum of the costs per arcs, since

$$\overline{C}(\pi_{M+1}) = \sum_{i=1,\ldots,M} (\theta_0 \cdot |B_i \setminus B_{i-1}| - v(B_i \setminus B_{i-1})) + \theta_0 \cdot |Y \setminus B_M|$$

■

4.3 Heuristic search algorithms for solving the problem

4.3.1 State-space representation of a problem. Search graph, best-first search.

Once a specific problem is given, we can obtain its associated graph of the state-space representation type, where the nodes correspond to partial problem solution states and the arcs correspond to steps in a problem solving process. An initial state, corresponding to the given information in a problem instance, forms the root of the graph. The graph also defines a goal condition, which is the solution to a problem instance. The search on a graph of state space representation type characterizes problem solving as the process of finding a solution path from the initial state to a goal (Russell and Norvig, 1995). In view of this definition the $(M+1)$ stages dynamic programming oriented graph (Y, T) is already a state space representation of the problem (A), and the main result of the previous section was that the problem of finding the minimal cost path in this graph is equivalent to the one of obtaining an optimal allocation of the set of design tasks $X(t_0)$ to M engineers in the sense of the problem (A).

It is helpful to think of the search process as building up a search graph $S = (Y', T')$ that is an oriented subgraph of (Y, T) superimposed over it. Such a search graph is determined by a triple (I, O, G), where:

1) I is the set of initial states of the problem

2) O is the set of legal rules/operators that can be applied in order to generate the children of a node

3) G is the set of goal states of the problem.

Assuming that we have constructed an heuristic evaluation function, the main steps of a general heuristic graph best-first search method are:

1. Start with a vertex set called OPEN, containing just the initial states and an empty vertex set called CLOSED

2. Until a goal node is found, or there are no nodes left on OPEN do:

· Pick the node having the smallest value of the evaluation function among the ones in OPEN

· Generate its children, remove the node from OPEN and put it in the list of nodes called CLOSED

· For each child do:

(a) If it has not been generated before, evaluate its heuristic function, add it to OPEN, and record its parent

(b) If it has been generated before, change the parent if this new parent is better (from the evaluation function point of view) than the previous one. In that case, update the cost of getting to this node and to any children that this node may already have.

where

OPEN is the list of nodes that have been generated and have had the heuristic evaluation function applied to them but which have not yet been expanded (i.e. their children have not been generated yet) OPEN is actually a priority queue in which the elements with the highest priority are those with the smallest value of the heuristic function.

CLOSED is the list of nodes that have already been expanded. We need to keep these nodes in memory if we want to search a graph rather than a tree, since whenever a new node is generated, we need to check whether it has been generated before.

The stopping rule of the algorithm contains a condition for the moment in which we reach a goal state and also a condition for the problems which do not have feasible solutions. It is obvious that the existence of a solution depends on the set of input data.

4.3.2 Construction of a monotonic A* heuristic evaluation function

We consider only (I, O, G) triples with only one initial state: $|I| = 1$. Let us denote by $g^*, h^* : Y \rightarrow R_+$ the functions defined such that for all $n \in Y$, $g^*(n)$ is the cost of the shortest path from the start node to node n and for all $n \in Y$, $h^*(n)$ is the actual cost of the shortest path from n to a goal. Thus, the function $f^* : Y \rightarrow R_+$ defined, for all $n \in Y$, by $f^*(n) = g^*(n) + h^*(n)$ is in any node n the actual cost of the optimal path from a start node to a goal node that passes through node n. And, let $g, h : Y \rightarrow R_+$ be two functions such that for all $n \in Y$, $g(n)$ equals the cost of the current path to the node n, obtained by summing the costs of the arcs from the initial state to n, and for all $n \in Y$, $h(n)$ is an estimate of the actual cost from n to a goal state.

Definition 7 *A best-first search algorithm using $f = g + h$ as previously described, as an evaluation function for ordering nodes in a general heuristic graph search method is called an algorithm A, and f is called an A heuristic evaluation function. An A algorithm where $h(n) \leq h^*(n)$ is called A* and f is called an A* heuristic evaluation function (see (Nilsson, 1982)).*

Definition 8 *A best-first search algorithm is said to be admissible or optimal if for any state-space representation graph having a finite cost path to a goal state the algorithm finds an optimal path. (see (Nilsson, 1982))*

We also have, by immediately combining several results from (Nilsson, 1982) the following:

Proposition 9 *All A* algorithms are admissible, provided there is some strictly positive constant ε such that the cost on each arc of the state-space representation graph is at least ε, and each node in the graph has a finite number of children (if any).*

Lemma 10 *Consider the function $\overline{D} : \mathcal{P}(Y)\times\mathcal{P}(Y) \to \mathbb{R}_+$ defined by $\overline{D}(A,B) = v(B\backslash A)$, then we have the following properties:*

1. $\overline{D}(A,B) \geq 0$ *for all* $A, B \in \mathcal{P}(Y)$.
2. $\overline{D}(A,C) = \overline{D}(A,B) + \overline{D}(B,C)$ *for all* $A \subset B \subset C$ *with* $A, B, C \in \mathcal{P}(Y)$.
3. $\overline{C}(\pi_{M+1}) = \sum\limits_{i=1,\ldots,M} \left(\theta_0 \cdot |B_i \setminus B_{i-1}| - \overline{D}(B_{i-1}, B_i)\right) + \theta_0 \cdot |Y \setminus B_M|$

Proof. The first two properties hold due to the linearity and positivity of the cumulated value function u, while the last is straightforward from the definition of $\overline{D}$ and $\overline{C}$. ■

Proposition 11 *There exists an evaluation function $f = g + h$ defining an admissible A* algorithm for the (A) problem, provided that for the general search graph method we consider the triple $(\widetilde{I}, \widetilde{O}, \widetilde{G})$ where:*

$a)\ \widetilde{I} := \{\emptyset\}$

$b)\ \widetilde{O} := \left\{ \begin{array}{ll} \emptyset \text{ has as child } (\emptyset, A_1) & \text{if } A_1 \in D_1(\emptyset) \\ (B_{i-1}, A_i) \text{ has as child } (B_i, A_{i+1}) & \text{if } B_i = B_{i-1} \cup A_i \text{ with } \\ & A_{i+1} \in D_{i+1}(B_{i-1}, A_i) \end{array} \right\}$

$c)\ \widetilde{G} := \{(B_M, A_{M+1}) \,|A_{M+1} = (Y \setminus (B_{M-1} \cup A_M))\}$

Proof. We consider as a state space representation of the problem (A) the $(M+1)$ stages dynamic programming oriented graph $G = (Y, T)$ constructed in proposition 5 with its corresponding costs.

For every node describing a state of the problem of the type $(B_{i-1}, A_i), i = \overline{1, M+1}$ we have according to the definition of an A* algorithm that: $f^*(B_{i-1}, A_i) = g^*(B_{i-1}, A_i) + h^*(B_{i-1}, A_i)$, where $g^*(B_{i-1}, A_i)$ is the cost associated to an optimal path from $\emptyset$ to (B_{i-1}, A_i) and $h^*(B_{i-1}, A_i)$ is the cost associated with an optimal path from (B_{i-1}, A_i) to a final node.

We can construct approximations for the above defined evaluation functions:

$$
\begin{aligned}
f(B_{i-1}, A_i) &= g(B_{i-1}, A_i) + h(B_{i-1}, A_i) \\
g(B_{i-1}, A_i) &= \sum_{j=1}^{i} \left[\theta_0 \cdot |B_j \backslash B_{j-1}| - \overline{D}(B_{j-1}, B_j)\right] \\
h(B_{i-1}, A_i) &= \varepsilon_0 + \theta_0 \cdot |Y \backslash B_i| - \overline{D}(B_i, Y), \forall i = 1, ..., M-1 \\
h(B_{M-1}, A_M) &= \varepsilon_0 + \theta_0 \cdot |Y \setminus (B_{M-1} \cup A_M)| \\
h(B_M, A_{M+1}) &= 0
\end{aligned}
$$

Since there exists θ_0 such that $h(B_{i-1}, A_i) \geq 0, \forall i = 1, ..., M+1$, if we can prove that for any node $(B_{i-1}, A_i), i = 1, ..., M+1$ the relation $h(B_{i-1}, A_i) \leq h^*(B_{i-1}, A_i)$ holds, then the algorithm constructed with the above defined evaluation function is an A* algorithm, according to the definition.

Since our oriented graph G is finite and all the costs associated with the arcs of G are strictly positive, we have that there exists an $\varepsilon_0 > 0$ such that the cost on each arc of the state-space representation graph is at least ε_0. So, if our algorithm is of A* type then by using proposition 9 we have that our algorithm is an admissible one.

From the definitions of costs, we see that we can take $\varepsilon_0 = 1$, because the minimum cost, equal to 1, is reached when $|A_{i+1}| = 1$ and its element (the design task) has maximal value function. If from (B_{i-1}, A_i) we cannot reach a goal solution then $h(B_{i-1}, A_i) < h^*(B_{i-1}, A_i) = \infty$. Otherwise, let $((B_{i-1}, A_i), (B_i, A_{i+1}), ..., (B_M, A_{M+1}))$ be the best path (allocation) from the cost point of view which can be obtained by generating all the children of (B_{i-1}, A_i) according to the rules from $\widetilde{O}$. We have the following two possibilities

$a)$ $i < M$. Then using lemma 10 we have that

$$\begin{aligned} h(B_{i-1}, A_i) &= 1 + \theta_0 \cdot |Y \backslash B_i| - \overline{D}(B_i, Y) = \\ &= 1 + \theta_0 \cdot |Y \backslash B_i| - \sum_{j=i+1}^{M+1} \overline{D}(B_{j-1}, B_j) \leq \\ &\leq 1 + \theta_0 \cdot \sum_{j=i+1}^{M+1} |A_j| - \sum_{j=i+1}^{M} \overline{D}(B_{j-1}, B_j) \\ &= h^*(B_{i-1}, A_i) \end{aligned}$$

$b)$ $i = M$. In this case we have $h(B_{M-1}, A_M) = h^*(B_{M-1}, A_M)$.

$c)$ $i = M + 1$. In this case we also have $h(B_M, A_{M+1}) = 0 = h^*(B_M, A_{M+1})$.

■

Corollary 12 *For the (A) problem there exists an evaluation function defining an A* type algorithm which is guaranteed to find an optimal cost allocation of the design tasks to engineers.*

A* algorithms do not require $g(n) = g^*(n)$, therefore admissible heuristics may initially reach non-goal, non-final states along a suboptimal path, as long as the algorithm finds an optimal path to all states on the path to a goal. One way of describing the monotone property is that the search is everywhere locally consistent (i.e. they consistently find the minimal path to each state they encounter in the search) with the heuristic function used. The difference between the heuristic measure for a state and any one of its descendants is bound by the actual cost of going between the state and its descendent. This is to say that the heuristic function is everywhere admissible, reaching each state along the shortest path from its ancestors.

Definition 13 *A heuristic function h is said to satisfy the monotone restriction/consistency assumption if:*

1. for all n_i and n_j, where n_j is a descendant of n_i (i.e. n_j is obtained during the process of expanding the children of n_i) holds $h(n_i) - h(n_j) \leq cost(n_i, n_j) :=$ the actual cost of a minimal path from state n_i to state n_j.

2. $h(goal) = 0$.(see (Nilsson, 1982))

Proposition 14 *If for the function h the monotone restriction is satisfied, then A* has already found an optimal path to any node it selects for expansion. That is, if A* selects node n for expansion, and if the monotone restriction is satisfied, $g(n) = g^*(n)$. (see (Nilsson, 1982))*

Whenever a search algorithm using a monotonic heuristic rediscovers a state, it is no longer necessary to check if the path to it is shorter than the previous one, because it will surely not be! This allows any state that is rediscovered in the space to be dropped immediately without updating the path information retained on OPEN or CLOSED. Hence the computational complexity of the algorithm is reduced substantially.

Proposition 15 *Applied to the problem (A), the function h, defined in the proposition 11 satisfies the monotone restriction.*

Proof. Let (B_{i-1}, A_i) and (B_{i+m-1}, A_{i+m}) with $m \geq 1, i \leq M+1$ be two nodes connected by an arbitrary path $P_{i,i+m} = (B_{i-1}, A_i), (B_i, A_{i+1}), .., (B_{i+m-1}, A_{i+m})$. Then, by definition of the function h and by lemma 10 we have that:

$$\begin{aligned} h(B_{i-1}, A_i) - h(B_{i+m-1}, A_{i+m}) &= \theta_0 \cdot [|Y \backslash B_i| - |Y \backslash B_{i+m}|] - \overline{D}(B_i, Y) + \\ &+ \overline{D}(B_{i+m}, Y) \\ &= \theta_0 \cdot |B_{i+m} \backslash B_i| - \sum_{j=i+1}^{i+m} \overline{D}(B_{j-1}, B_j). \end{aligned}$$

By applying once more the same lemma 10, we have

$$\begin{aligned} h(B_{i-1}, A_i) - h(B_{i+m-1}, A_{i+m}) \leq \min_{P_{i,i+m}} \left\{ \theta_0 \cdot |B_{i+m} \backslash B_i| - \sum_{j=i+1}^{i+m} \overline{D}(B_{j-1}, B_j) \right\} = \\ = \text{cost}((B_{i-1}, A_i), (B_{i+m-1}, A_{i+m})) \end{aligned}$$

For $i + m - 1 = M$ the relation holds trivially since $h(B_M, A_{M+1}) = 0$ and $h(B_{M-m}, A_{M-m+1}) \leq h^*(B_{M-m}, A_{M-m+1})$ =cost$((B_{M-m}, A_{M-m+1}),$ goal state$)$. ■

4.3.3 Implementation and experimental results

The problem with the general best-first search algorithm is its spatial complexity. The general algorithm has to store in memory all the frontier nodes (of the already explored subgraph out from the total search space). This can be exponential: in our case, one may have to store $O(b^M)$ nodes in a worst case, where b is the average branching factor, that is the average number of children of a node. There are several variants of algorithms which simulate the general best-first search, that is, they explore the state space also in best-first order, yet with a different definition and management of the open and closed lists. These variants have been designed in order to get around the spatial complexity difficulty. We can mention iterative deepening (ID), node retraction, and recursive best-first search (RBFS) (see (Korf, 1995) for a discussion). In order to gain on the side of spatial complexity the most, we chose to implement an A*-RBFS variant for our A* algorithm. This variant combines the ideas and advantages of IDA* and node retraction with an A* evaluation function (see (Korf,

1995)), and it reduces the space complexity of the general best-first search from exponential to linear. This is done at a cost of only a constant factor in time complexity, as experimental evidence shows in (Korf, 1993), where the RBFS algorithm has first been presented.

In general, given an A*-evaluation function (as the one we have just constructed), the RBFS variant can be shown to always terminate, and to find an optimal solution, if there exists one (see (Korf, 1993)). Also, the same author shows that with a monotonic A*-evaluation function, RBFS generates fewer nodes than IDA*, up to tie-breaking among nodes whose cost equal the solution cost.

The RBFS algorithm always expands nodes in best-first order, storing in our case $O(bM)$ nodes at most. This algorithm goes as follows (see (Korf, 1993))

```
RBFS(node: N, bound: B)
if N.f > B then return N.f
if goal(N)=YES then EXIT
T:=children(N)
if length(T)=0 then return infinity
if length(T)=1 then a:=infinity
for each i from 1 to length(T) do
   if T[i].f < N.F then T[i].F := max(N.F,T[i].f)
   else T[i].F := T[i].f
sort(T) /* increasing order of T[i].F */
while T[1].F ≤ B and T[1].F < infinity do
   if length(T)>1 then a:=T[2].F
   T[1].F := RBFS(T[1],min(B,a))
   sort(T)
endwhile
return T[1].F
```

We see that besides the evaluation function f, for each node there exists another evaluation function, namely F, which changes during the exploration. The initialization for the root sets the F value to f, and the bound B to infinity. The RBFS algorithm may visit nodes more than once, thus visiting a greater number of nodes overall speaking, because of the updating of this function F. Yet the advantage over standard best-first is that this it may have to store $O(b^M)$ nodes in a worst case, while RBFS stores only $O(bM)$.

Experimental evidence tends to show moreover that this rate of revisiting nodes is very small, and this is in accord with the extensive experiments performed in (see (Korf, 1993)), comparing (when the available memory permitted it) RBFS with standard best-first search. Our experiments have been done for various correlations of duration (number of activities) with value function for the tasks, and for a few tens of tasks and a few engineers.

1. weak correlation: one cluster and a few outliers
2. weak correlation: three clusters
3. strong correlation: uniform distribution over one or two clusters

The results are collected in Table 4.1, giving, for 20 to 50 design tasks, and 5 engineers, the cardinality of the whole search space, and the average cardinality of the set of effectively expanded nodes by the algorithm, for the three types of correlation. The rate of revisiting nodes is under 0.1%. This is why we only give the set cardinality. For each entry

No. tasks	Cardinality of the whole search space (Stirling number)	Expanded nodes		
		Weak correlation		Strong correlation uniform, 1 or 2 cls
		outliers	3 cls	
20	749206090500	271	290	7821
25	2436684974110751	296	301	64231
30	7713000216608565075	322	412	2619156
35	24204004899040755811870	2142	21146	19645801
40	75740854251732106906082250	3600	472023	128452100
45	23679599799792256039278924265O1	8352	1532010	392084301
50	74009586436825301627l188139587625	15170	11512501	859780221

Table 4.1. Expanded nodes versus cardinality of whole search space for RBFS implementation

in the table, the averages have been computed independently for 25 different data sets. On the average our numbers should be accurate only within an order of magnitude, since the data was independently random generated for each test, and the space of all possible input data has a very large size compared to the number of actual data sets we tested on.

As the number of design tasks increases (we have done experiments up to 50 tasks), in the weak correlation case the numbers stay very small, while in the strong correlation, as one would expect, the execution time and number of expanded nodes is increasing. Moreover, from a practical point of view this allocation problem becomes more important when the design tasks to be allocated are quite different (i.e. there is a weak correlation in between the number of activities per design task).

4.4 Conclusions

The combinatorial nature of this allocation problem stems from the fact that the number of alternative ways of grouping K objects into M groups is given by a Stirling number of the second kind $S_K^{(M)}$, where $S_K^{(M)} = \frac{1}{M!}\sum_{m=0}^{M}\left[(-1)^{M-m}\begin{pmatrix} M \\ m \end{pmatrix} m^K\right]$ (see Table 4.1 for values). So, the solving method described in this paper starts with a dynamic programming model but it uses heuristic search algorithms, primarily due to the lack of analytical solutions with a tractable computational complexity. However, the general algorithm constructed in this paper has the property that if another algorithm of its type expands fewer nodes than it, then that other algorithm runs the risk of missing the optimal solution.

Proposition 16 *Among optimal algorithms (i.e. algorithms that find the highest quality solution when there are several different solutions) of its type – algorithms that extend search paths from the root – A* is optimally efficient for any given monotonic heuristic f. That is, no other optimal algorithm is guaranteed to expand fewer nodes than A*. (see (Dechter and Pearl, 1985))*

Corollary 17 *For the (A) problem we have constructed a general A* algorithm which is guaranteed to find an optimal cost allocation of the design tasks to engineers and to expand*

for any given monotonic heuristic f the minimal number of nodes from its associated dynamic programming graph.

We have chosen to implement this general algorithm through a variant: A* recursive best-first search (RBFS). This variant also expands nodes in best-first order, yet with a theoretic overhead of revisiting and reexpanding some nodes. However, the main advantage is the linear space complexity of $O(bM)$ nodes stored, versus the exponential space complexity of $O(b^M)$ nodes in the general algorithm. Moreover, experimental evidence shows a very low rate of revisited nodes, under 0.1%, and is in agreement with theoretical arguments, which show that in a particular abstract model, RBFS is asymptotically optimally efficient (see (Korf, 1993)).

Experimental evidence from our tests also shows a very low rate of number of nodes effectively visited, compared to the total number of nodes in the search space, given by the Stirling number of the second kind.

Chapter 5

Multi-period Aggregate Decision — A Markov Approach

5.1 Introduction

In Chapter 3 Section 3.3 we have discussed the aggregate decision problem for one review period. With the purpose of predicting the outcome of the NPD project at its deadline, we have to study this problem for multiple review periods. In order to be able to do that, we need a mechanism to predict what happens at the lower levels. Such a mechanism will help us have a formula to compute the transition probabilities of a discrete-time, finite-horizon, non-stationary Markov decision problem, which is what we propose to solve the multi-period aggregate decision problem. This Markovian decision process that we present in this chapter is an extremely general framework that supports the dynamic achievement of the new product definition. It takes into account not only a highly dynamic market situation, but also a high technological uncertainty that affects the content of the project design tasks.

This chapter is organized as follows. In Section 5.2 we introduce simple heuristics for both the engineering process, and the detailed planning process, and we construct and validate a simple queueing model to estimate the solving time distribution of design tasks in NPD projects. In Section 5.3 we reduce the influence of the market uncertainty on the NPD project, focusing on its technological uncertainty. Therefore, we take into account only the market payoff function values available at the deadline T of the NPD project. These modelling choices allow the computation of transition probabilities for a non-stationary Markovian decision process model of the aggregate decision problem. In the end of Section 5.3, we show that for this model there exists an optimal market payoff value as well as optimal policies to achieve it. In Section 5.4 we restrict the general Markovian decision problem to two particular cases. The first case considered is the one of an NPD project without precedence constraints. The second one is the case of an NPD project consisting of a sequence of design tasks. In the following chapters we obtaine for both cases structured optimal policies. By combining the derived insights we provided guidelines for heuristic policies in the general case situation.

5.2 An estimation model for the solving time distribution of a set of NPD design tasks with precedence relationships

This section concerns the detailed planning level and the engineering level; here we present the estimation model used as a tool to predict what happens at these levels, for use in our multi-period aggregate decision control. We now focus on the detailed planning level and engineering level.

Considering the potential risks of new product development projects (NPD), the characteristics of their design tasks are critical for an effective management. From an operations management perspective it is important to realize what operational characteristics of design tasks cause projects to be late. Earlier attempts to study and model the variability of the NPD design tasks are due to (Oorschot, 2001), and (Tatikonda and Rosenthal, 2000a). They identify the technological novelty, the magnitude of the design tasks, the interactions between the design tasks in the NPD project, and the balancing between projects among the most important causes of the unpredictability of the design tasks solving times in NPD projects.

The main contribution of this section is the derivation of a simple mathematical model for both the engineering and detailed planning level processes from Chapter 3. This model allows the estimation of solving time distribution function of a set of design tasks with precedence relationships from an NPD project. The model is based on operational characteristics of NPD projects that evolved from theoretical and empirical research on these projects. Moreover, we try to asses the validity of our model on real-life data.

5.2.1 Model description and analysis

We focus on both the detailed planning and engineering processes that take place during an arbitrary review period of the NPD project control model from Chapter 3 Section 3.4. We recall that at the beginning of each review period we have a subnetwork of (planned or newly arrived) design tasks being subject to precedence constraints inherited from the project network. As a consequence of the assumptions from Chapter 2, Section 2.3, at the detailed planning level each design task n can be viewed as a *list of planned activities*, $N_a(n)$ (to be sequentially performed). The split of each task into activities gives a uniform measure of the difficulty implied by its realization. For the unfinished design tasks, unplanned activities arrive according to a Poisson process of rate λ. They appear as a result of the incapacity to foresee at the outset all activities needed to complete the design tasks, so we model them to have preemptive resume priority over the planned activities. The solving time of the activities are random variables independent identically exponentially distributed $Exp(1/\mu)$.

The notation used in this chapter is a simplification of the notation from Chapter 3 Section 3.4, and Section 3.5. The references to the target performance level, or to the time moment were removed from the notation. The current target performance levels are established during the aggregate decision level, so they cannot be influenced here. Also, the detailed planning, and engineering process control problems occur at the beginning of each review period, and we assumed that an approaching deadline of the entire NPD project will not influence them.

Input parameters:

M : the number of engineers;

μ : the rate of the exponential solving time for each activity (of any design task);

λ: the rate of the Poisson arrival of unplanned activities for all the design tasks allocated to the engineers at the beginning of the current review period;

η : the rate of the exponential solving time for the comparisons in between each two design tasks during the detailed level reprioritization process;

$G = (J(G), \mathcal{A}(G))$: the directed, acyclic graph of precedence relations among the design tasks;

$V(n)$: the value function of the design task n; $V(n) \in \mathbb{R}_+$, for all $n \in J(G)$;

$N_a\,(n)$: the total number of activities of the design task n; for all $n \in J(G)Y$;

Even if all the design tasks allocated in the previous review periods are finished, the previous chapters have shown that the detailed planning process is a complicated one. Nevertheless, the knowledge about its results is crucial for the aggregate decision, and the process of acquiring this knowledge should be very simple in order to allow the decision maker to evaluate the various possible choices.

What, we propose is to assume that the team of engineers will work with all its available capacity on any of the design tasks. Such an heuristic simplifies the detailed planning problem by allowing more than one engineer to work on a design task. The order in which the design tasks are performed may be approximated. In the case of the general cumulative market payoff functions from (Askin and Dawson, 2000), and (Yoshimura, 1996) we can order the design tasks to the team of engineers according to their value (see Appendix and Chapter 3 Section 3.4). Other heuristic orderings suitable in the case of a new product which has to fulfill only one customer need might be heuristics for nonpreemptive scheduling problems with identical processors (i.e. in our case the engineers) and precedence constraints. There, the concept of list schedule turned out to be useful (see for a list (Neumann and Zimmerman, 1998)). An unscheduled design task is said to be ready if all its predecessors, if any, have been solved. A list schedule is a permutation σ of the set of all design tasks with the following meaning: any time that a processor becomes idle, the solving of that ready design task with the minimum index in the permutation σ is begun by the idle processor.

The sequence of design tasks allocated to the team may contain more than one design task that can be performed in parallel and their order in a sequence reflects only the optimality criteria of the scheduler from the detailed planning level. Unlike machines, human beings are able to perceive the concurrency and the relative urgency of design tasks. Therefore, it is reasonable to assume that during the planning period the engineers will not work all the time on the sequenced design tasks in the order established by the scheduler. The decision of an engineer of choosing or not a specific design task to work on is not really something that can be exactly modelled, due to the large variety of variables implied and due to the lack of detailed good quality empirical data to support theories. However, we can model this in probabilistic terms, by assuming that at each time instant the team can decide, with a small probability $1 - p$, to temporarily re-prioritize the order of the design tasks that can be performed in parallel, by successively comparing the first activities to be done from each of the other allocated tasks. Comparing design tasks requires capacity, which can be modelled as an activity with exponential distribution with mean $1/\eta$. Thus, at any time instant the team will either work with the probability p on the planned activities of the design tasks in the order given by the scheduler, or, with a probability $1 - p$, will try to re-prioritize the design tasks instead of trying to solve them.

Proposition 18 *Let Ξ be the random variable giving the solving time of the allocated design tasks, and i the number of concurrent design tasks allocated to the engineers. Then for $i > 2$, Ξ's cumulative distribution function equals $F_Y(h) = \Pr\{\Xi \leq h\} = p \cdot \Pr\{C_{\max}(k) \leq h\} + (1-p) \cdot \Pr\{Y_{Erlang(i-1)} \leq h\}$, where $C_{\max}(k)$ is the makespan of a queueing system of M parallel servers with a common queue having a series of $k = \sum_{n \in J(G) \cup Y} N_a(n)$ of planned activities, and $(1-p)$ is the probability that during the solving process the design tasks order is reconsidered. The planned activities have a common processing time, exponentially distributed with mean $\frac{1}{\mu}$, and their solving process is disturbed by a $\lambda-$Poisson arrival of unplanned activities. The arrival process of unplanned activities stops when all the planned activities are finished. Thus, the cumulative distribution function of Y can be computed numerically.*

Proof. All the planned activities are assumed to be independent identically distributed, and once the design tasks are ordered, we have as well an ordering among all the planned activities included in all the design tasks. Thus, the switches due to the re-prioritizing decisions may change only the indices of the activities to be done. So, the solving time of the planned activities already ordered is independent of the trials done in order to re-prioritize the design tasks. However, this process creates extra work and will affect the completion time distribution of the entire set of design tasks. Thus, $\Pr\{\Xi \leq h\} := p \cdot \Pr\{C_{\max}(k) \leq h\} + (1-p) \cdot \Pr\{Y_{Erlang(i-1)} \leq h\}$. Now we have to prove that the Laplace-transform of the distribution function of $C_{max}(k)$ can be obtained, and inverted to complete our proof. Let BP be the busy period in a $\mathrm{M/M/1}$ queue with arrival rate λ and service rate $M\mu$. Then we have, for $k = M + q \geq M$

$$C_{\max}(M+q) = BP + C_{\max}(M+q-1)$$

and for $k < M$,

$$C_{\max}(k) = Z(\Theta(k)) + \begin{cases} C_{\max}(k+1) & \text{with probability } \frac{\lambda}{\Theta(k)} \\ C_{\max}(k-1) & \text{with probability } \frac{k\mu}{\Theta(k)} \end{cases}$$

$$C_{\max}(0) = 0$$

where $\Theta(k) = \lambda + k\mu$ and Z is an exponential random variable. Let us consider the Laplace transform of $C_{\max}(k)$, and denote, for $s \geq 0$,

$$\widehat{f}_k(s) = \Phi(k)(s) = E(\mathrm{e}^{-sC_{\max}(k)})$$

After computing it, the Laplace transform $\widehat{f}_k(s)$ can be inverted according to the Euler Inverse Laplace Transform method ((Abate and Whitt, 1995)). We have two cases, according to the value of k with respect to M.

Case 1: $k = M+q \geq M : \Phi(k)(s) = \beta(s)\Phi(k-1)(s) = \cdots = \beta(s)^{k-M+1}\Phi(M-1)(s)$, where $\beta(s)$ is the Laplace-Stieltjes transform of the busy period BP. From (Kleinrock, 1975), we have that

$$\beta(s) = \frac{\lambda + M\mu + s - \sqrt{(\lambda + M\mu + s)^2 - 4\lambda M\mu}}{2\lambda}$$

So we shall only need to know how to compute $\Phi(M-1)(s)$. For that, we need the other case.

Case 2: $k < M$

$$\begin{aligned}\Phi(k)(s) &= \frac{\Theta(k)}{\Theta(k)+s}\left(\frac{\lambda}{\Theta(k)}\Phi(k+1)(s) + \frac{k\mu}{\Theta(k)}\Phi(k-1)(s)\right)\\ &= \frac{\lambda}{\Theta(k)+s}\Phi(k+1)(s) + \frac{k\mu}{\Theta(k)+s}\Phi(k-1)(s)\\ \Phi(0)(s) &= 1\end{aligned}$$

Now, on the border, we have the following:

$$\begin{aligned}\Phi(M-1)(s) &= \frac{\lambda}{\Theta(M-1)+s}\Phi(M)(s) + \frac{(M-1)\mu}{\Theta(M-1)+s}\Phi(M-2)(s)\\ \Phi(M)(s) &= \beta(s)\Phi(M-1)(s)\\ \Longrightarrow \Phi(M-1)(s) &= \frac{\lambda\beta(s)}{\Theta(M-1)+s}\Phi(M-1)(s) + \frac{(M-1)\mu}{\Theta(M-1)+s}\Phi(M-2)(s)\\ \Longrightarrow \Phi(M-1)(s) &= \frac{1}{1-\frac{\lambda\beta(s)}{\Theta(M-1)+s}}\cdot\frac{(M-1)\mu}{\Theta(M-1)+s}\Phi(M-2)(s)\\ \Longrightarrow \Phi(M-1)(s) &= \frac{(M-1)\mu}{\Theta(M-1)+s-\lambda\beta(s)}\Phi(M-2)(s)\end{aligned}$$

We want to have a relation linking the values of $\Phi(k)(s)$ either with respect to $\Phi(\cdot)(s)$ of greater values than k, or of smaller values, not both at the same time. Fix s and denote by $\Phi_k = \Phi(k)(s)$, by $\alpha(k) := \frac{\lambda}{\Theta(k)+s}$ and by $\delta(k) := \frac{k\mu}{\Theta(k)+s}$. Then we can rewrite the recurrence of case $k < M$ as follows

$$\Phi_k = \alpha(k)\cdot\Phi_{k+1} + \delta(k)\cdot\Phi_{k-1}$$

with the fact that $\alpha(M-2) = \frac{\Theta(M-1)+s-\lambda\beta(s)}{(M-1)\mu}$ and $\delta(M-2) = 0$. Because all Φ_k are positive, but not zero (because they are the result of an integral from a strictly positive function), we can also define $\gamma(k+1) = \frac{\Phi_{k+1}}{\Phi_k}$. Then we can rewrite the last equation as follows

$$\Phi_k = \frac{\delta(k)}{1-\alpha(k)\cdot\gamma(k+1)}f_{k-1} = \gamma(k)\cdot\Phi_{k-1} = \Phi_0\cdot\prod_{j=k}^{1}\gamma(j) = \prod_{j=k}^{1}\gamma(j)$$

and for each γ we have the recurrence relation

$$\gamma(k) = \frac{\delta(k)}{1-\alpha(k)\cdot\gamma(k+1)}, \text{ with } \gamma(M-1) = \frac{(M-1)\mu}{\Theta(M-1)+s-\lambda\beta(s)}$$

This shows how to compute each of the $\Phi_k = \Phi(k)(s)$, for an arbitrary $s \geq 0$. Thus, we completely solved the case $k < M$. ■

The shape of the distribution function obtained via this model is confirmed by the data collected in the experimental research of (Innam, 1999) (time to repair distribution functions in manufacturing systems), which suggest a long fat-tailed, skewed, maybe multi-modal distribution function. In Subsection 5.2.2 we test with a goodness-of-fit test that the frequency diagrams of the eight real-life data sets of (Oorschot, 2001) are consistent with our model cumulative distribution function for the particular case of one engineer. The last data frequency diagrams showed some similarity with gamma-type probability density functions as well, but this hypothesis was rejected by a goodness of fit test at $\alpha = 0.01$ in (Oorschot et al., 2002).

5.2.2 Goodness-of-fit test results

We have tested the null hypothesis that the frequency diagrams of the eight real-life data sets from (Oorschot, 2001) represent the model cumulative distribution function for the lead times of the NPD design tasks. The eight sets of data were collected from ten engineers of a firm that develops, produces and services advanced micro-lithography systems. Each engineer worked on two or three design tasks allocated to him. Before starting to work on the design tasks, the engineers made estimates from one to eight weeks for the design tasks lead times. Thus, the first data set contained the design tasks that were supposed to be finished in one week, the second contained the ones with estimated lead time two weeks, and so on. We used only the initial engineers conjectured characteristics of the design tasks in setting the parameters of the model. Since we did not have any information regarding the average time of an comparing any two design tasks, we tested the data considering $\eta = \mu$.

However, two important issues have to be discussed in relationship with the model testing. The first issue was that the data was collected separately per design task, and not per engineer. We did not know which set of tasks was performed by each engineer. This lead us to consider in our model that for each engineer the resequencing of its allocated design tasks took place during the solving of each of its design tasks with the same probability p. Thus, if an engineer had $i > 2$ concurrent design tasks allocated to him/her, the cumulative distribution function of one of them, say n, was of the form: $F_Y(h) = \Pr\{\Xi \leq h\} = p \cdot \Pr\{C(n) \leq h\} + (1-p) \cdot \Pr\{Y_{Erlang(i-1)} \leq h\}$, where $(1-p)$ is the probability that during the solving process the design tasks order is reconsidered, and $C(n)$ is the completion time of queueing system with one server with a queue having a series of only $N_a(n)$ of planned activities (i.e. those planned for that specific design task n). The planned activities have a common processing time, exponentially distributed with mean $\frac{1}{\mu}$, and their solving process is disturbed by a $\lambda-$Poisson arrival of unplanned activities. The arrival process of unplanned activities stops when all the planned activities are finished.

The second issue was that the data collected referred to the design tasks lead times, not to their solving times. If for the short lead times we could consider the lead time derived directly from the solving time, a different situation held for the longer lead times (i.e. more than three weeks). There were two main reasons for a design task to have a longer lead time, but the data was not collected separately for each. Either the design task lead time was indeed proportional with the amount of planned activities, or the engineer to whom was allocated had other design tasks to be performed earlier. Thus, we checked our model for both conjectured cases. For the first case we used a multiplication of the stochastic planned activities proportional with the number of weeks given by the due date. For the second case we delayed the distribution given by the model proportional with the number of weeks given by the lead time. Based on the results of the Kolmogorov-Smirnov goodness of fit test, the model showed in both cases no statistically significant difference with respect to data sets considered. The Kolmogorov-Smirnov test finds the greatest discrepancy between the empirical and expected theoretical cumulative distribution functions, which is called the "D-statistic". We compared this against the critical D-statistic for that sample size. The results are shown in Table 5.1.

In general, we cannot reject the null hypothesis that the distribution is of the expected form according to the above model. The D-statistic was less than the critical one (for $\alpha = 0.1$) for most of the tests. In the remaining tests we could get an acceptance for a lower α, except for the fifth data set, for which the first conjectured hypothesis (i.e. without delay)

Conjectured Lead Time	Sample size	Model Parameters $(A(n), \mu, \lambda, p, i, \text{delay})$	D-statistic	K-S (0.01)	K-S (0.05)	K-S (0.1)
1 week	66	$(4, 4, 1, 0.85, 2, -)$	.1829	.1972	.1644	.1480
2 weeks	85	$(8, 4, 1, 0.85, 2, -)$	.1678	.1742	.1452	.1307
3 weeks	70	$(12, 4, 1, 0.85, 2, -)$	.1236	.1916	.1597	.1438
4 weeks	52	$(12, 4, 1, 0.85, 2, 1)$ $(16, 4, 1, 0.85, 2, -)$	.1519 .1397	.2217	.1848	.1663
5 weeks	43	$(12, 4, 1, 0.85, 2, 2)$ $(20, 4, 1, 0.85, 2, -)$	.1676 .2792	.2433	.2028	.1825
6 weeks	41	$(12, 4, 1, 0.85, 2, 3)$ $(24, 4, 1, 0.85, 2, -)$	.2020 .1524	.2490	.2076	.1868
7 weeks	35	$(12, 4, 1, 0.85, 3, 4)$ $(28, 4, 1, 0.85, 3, -)$	.1951 .1908	.2686	.2242	.2018
8 weeks	32	$(12, 4, 1, 0.85, 3, 5)$ $(32, 4, 1, 0.85, 3, -)$	.2531 .1996	.2809	.2342	.2108

Table 5.1. Kolmogorov-Smirnov goodness-of-fit test results

was rejected. We explain this slight variation in the acceptance rates, as well as the one rejection by the fact that probably the design tasks were having a longer lead time for both above mentioned reasons (the model $(16, 4, 1, 0.85, 2, 1)$ was accepted at a higher α than both separated cases). It is worthwhile to mention that if the conjectured lead time was one or two weeks we could also set $p = 1$, without rejecting the null hypothesis for $\alpha = 0.1$, which was not possible in all the other cases. Also, for the first two data sets which have a shorter initially estimated lead time, the application of our model in the second conjectured hypothesis (i.e. with delay) lead to a rejection. This may allow a model simplification for very short solving times. The psychological literature suggests that people tend to give more priority to urgent design tasks, presumably because time pressure rises and this motivates people to make progress.

5.3 Nonstationary Markovian multi-period aggregate control of NPD projects

This section focuses on the aggregate decision level, presenting the promised nonstationary Markov decision process. In order to do that, we make use of the results from Section 5.2 concerning the estimation model presented for the detailed planning level and engineering level. We consider that the short-time planning horizon h to be equal to the length of one review-period (i.e. h=1).

We propose a discrete-time control model, because it is too costly to continuously measure the progress in a NPD project. The project will be reviewed at equidistant points in time until the deadline, T.

At the beginning of each review period t, $(t, t + 1]$, the aggregate decision maker integrates in the NPD project the newly arrived unplanned design tasks. We consider that

all the new arrived design tasks during the review period $(t-1,t]$ are concurrent with the ones to be allocated at t, and each such task is to be performed only after all tasks of the stage $t-1$ are finished and before any task of the stage $t+1$. However, the computation of the transition probabilities can be easily extended to the case described in Chapter 2. There we assumed an arrival of unplanned design tasks (during review period $t-1$) concurrent with design tasks allocated/to be allocated during the $i-$th review period; $\forall i = 0,...,t$. For reasons of mathematical tractability, the general Markovian review period-dependent arrival process of unplanned design tasks from Chapter 2 and Chapter 3 Section 3.3 becomes a Poisson review period-dependent arrival process of rate $\varsigma(t)$. In later review periods the rate $\varsigma(t)$ decreases. We recall from Chapter 2 Subsection 2.3.4 that for each period $t-1$ and for each $i = 0,...,t$ the unplanned design tasks are assumed be statistically identical (i.e. with a common performance level structure, and an identical market payoff structure).

Afterwards, at the beginning of each review period the aggregate decision maker decides whether to continue or not the NPD project. The abandonment is the result of either an expected exceeded NPD budget, or of a low product performance, which does not enable the achievement of a fully functional product before the deadline. In case of continuation the controller modifies the design tasks performance levels in an interactive process aiming at a maximal market payoff at the deadline. Thus, the targets on design tasks realization for the detailed planning level are provided, under the several aggregate constraints of achieving the currently target performance levels, at the deadline, with certain probabilities. A $T-$partite directed acyclic graph of design tasks reflects the precedence relations among design tasks at the beginning of each review period (see Chapter 2 Subsection 2.3.2).

During each review period, we estimate the outcomes of the detailed planning and engineering process using the simple model built in Section 5.2. This model allows the computation of the transition probabilities of the Markov decision process we propose for the control of NPD projects. Thus, we assume that during an arbitrary review period the team of engineers will always perform the tasks allocated to it, according to a priority order. We also recall from Chapter 2, Section 2.3 that for each design task we have different levels of performance, giving the quality of its execution. Each performance level consists of a *list of planned activities* with solving times random variables independent identically exponentially distributed. For each design task a *minimal performance level* has to be achieved, in order to have a fully functional new product. New unplanned activities arrive at the design tasks in progress during the review period t according to a Poisson process of rate $\lambda(t)$. In later stages the rate $\lambda(t)$ decreases.

The notation used in this chapter was introduced in Chapter 3 Section 3.3, with the exception of the arrival rate of new activities. This rate is not an input parameter for the aggregate control in itself, but for the model we use for the engineering and detailed planning processes (see Section 5.2). In the Markov decision process formulation of the aggregate decision process we take into account the arrival of unplanned design activities while computing the transition probabilities of the process. Consequently, we will not update anymore the number of sequential activities planned for solving a design task n, at the performance level l. In their corresponding notation the time index will be withdrawn, as well as in the solving time of a performance level l of a design task n, and in the minimal performance level required.

Input parameters (global variables) **:**

T : the total number of review periods (review periods are numbered from 0 to $T-1$);

M : the total number of engineers;

$\overline{N}$: the initial number of design tasks;

N : an upper bound for the maximum number of design tasks during the whole project;

$L_{\max}(n)$: the maximal number of performance levels of the initial design task n; $n = 1, ..., \overline{N}$;

$l_{\min}(n)$: the minimal performance level at which the design task n must be performed in order to obtain a functional product; $n = 1, ..., N$

$N_a\,(n, l)$: the number of sequential activities planned for solving the design task n, at the performance level l, assuming the previous levels already solved; $\forall\, n = 1, ..., N, \forall l = 1, ..., L_{\max}(n)$;

μ : the rate of the exponential distribution of an activity solving time;

η : the rate of the exponential solving time for the comparisons in between each two design tasks during the detailed level reprioritization process;

$c\,(n)$: the cost of performing one activity of the design task n; $n = 1, ..., N$.

Since N is an upper bound, we set to zero all the parameters depending on a virtual $n \in \{\overline{N} + 1, \ldots, N\}$. If due to the arrival of unplanned design tasks more than N design tasks arrive then either they will be neglected, or the NPD project will be stopped.

Input parameters (at the beginning of review period t)**:**

$\alpha\,(t)$: the required current safety margin for the probability of completing the project before the deadline; $\alpha\,(t) \in (0, 1)$

$\beta\,(t)$: the required current safety margin for the probability of exceeding the maximal team solving capacity; $\beta\,(t) \in (0, 1)$

$B\,(t)$: the current remaining NPD project budget;

$\varsigma(t)$: the current rate of the Poisson review period-dependent arrival process of unplanned design tasks;

$\Omega(t-1)$: the set of newly arrived unplanned design tasks (during review period $t-1$) concurrent with the design tasks to be allocated during the tth review period;

$R_t := (J(R_t), \mathcal{A}(J(R_t)))$: the newly updated $T-$partite directed acyclic graph of unfinished design tasks precedence relations, where $J(R_t) = \Lambda_t^0 \cup \Lambda_t^1 \cup ... \cup \Lambda_t^{T-1}$ and $\Lambda_t^t = \Lambda_{t-1}^t \cup \Omega(t-1)$ is the current design tasks set allocated/to be allocated at t;

$N(t-1)$: the random variable giving the number of design tasks arrived since the NPD project beginning until the end of review period $t-1$, $[t-1, t)$;

$L_{\max}(n, t)$: the current maximal number of performance levels of the design task n; $L_{\max}(n, t) = L_{\max}(n)$ for $n = 1, ..., \overline{N}$ and $L_{\max}(n, t) = 0$ if $n \notin J(R_t)$ (i.e. there is place reserved for the design tasks not planned or not arrived yet up to the upper bound N but we set to zero the maximal performance level depending on a virtual design task n);

$l\,(\cdot, t) : \{1, ...N\} \to \mathbb{N}$: the achieved performance level of a design task function, where $0 \leq l\,(n, t) \leq L_{\max}(n, t)$, for $n \in J(R_t)$ and by convention we define $l\,(n, t) = -1$ for $n \notin J(R_t)$ (i.e. we mark the design tasks not arrived yet);

$\lambda(t)$: the review-period dependent Poisson arrival of unplanned activities for all the design tasks allocated to the engineers.

Notation (at the beginning of review period t):

$\digamma_t := \left\{n \geq \overline{N} \,|\overline{N} + E\left[N(t-1)\right] + 1 \leq \ n \leq \overline{N} + E\left[N(T-1)\right]\right\}$: the estimated set of all unplanned design tasks for NPD project;

$S_n(l)$: the solving time of the performance level l of the design task n, assuming the previous levels already solved; $n = 1, ..., N$. They are independent random variables Erlang-$(N_a(n,l), \mu)$.

$C(t, l(\cdot,t), R_t)$: the completion time of the network of design tasks, R_t, if $l(\cdot,t)$ gives the design tasks performance levels;

The decision time points: The decision points are equidistant and the horizon of the problem is finite. The decision point t corresponds to the beginning of review $t+1$. Say $t \in \{0, 1, ...T-1\}$.

The state space and the action space:

At time $t = 0$, at the beginning of the NPD project the engineers start with zero performance levels achieved for the $\overline{N}$ planned tasks, and preserve place for a number $N - \overline{N}$ of unplanned design tasks. Thus, for $t = 0$:
$X(0) = \{x_0\} = (0_{\mathbb{R}^{\overline{N}}}, -1_{\mathbb{R}^{N-\overline{N}}})$.

The state set $X\left(t\right)$ at moment t and the action set $A_t\left(x_t\right)$ in the state $x_t \in X\left(t\right)$ have probability constraints similar to conditions 3.2, 3.3 and 3.4 from the one-period aggregate decision problem presented in Section 3.3:

- the target performance level of each planned or newly arrived design task n is greater than $\min(l_{\min}(n,t), l(n,t))$, and smaller than $L_{\max}(n,t)$

$\Diamond$ *with the probability safety margins:*

- the completion time must be smaller than the remaining time until the deadline
- the remaining workload of the team of engineers should not exceed their maximal solving capacity
- the remaining budget must not be exceeded.

For the state space the probability constraints are computed using the minimal performance levels, $l_{\min}(\cdot,t)$, while for the action space are used the target performance levels, $l(\cdot,t) + a(\cdot,t)$. Why? Since the NPD project is stopped if in the achieved states, for the existing design tasks (i.e. $L_{\max}(n,t) > 0$) one cannot take anymore actions which lead at the deadline to a rewarded new product in the conditions required by the safety margins. The rewarded region is $R = \{(l_1, ..., l_N) | l_n \geq l_{\min}\left(n\right), \forall n = 1, ..., N\}$.

For $t \in \{1, \ldots, T\}$ the state $x_t \in X(t)$ describes how many performance levels $l(n,t)$ were solved for each design task n:

$$X(t) = \left\{ x_t \left| \begin{array}{c} x_t = (l(1,t), ..., l(n,t), ..., l(N,t)) \in \{\mathbb{N} \cup \{-1\}\}^N \\ \text{and } -1 \leq l\left(n,t\right) \leq L_{\max}(n,t),\ n \in \{1, ..., N\} \\ l(n,t) \geq 0, \forall n \in \{1, ..., N\} \text{ s.t. } L_{\max}(n,t) > 0 \\ \Pr\left\{C\left(t, l_{\min}\left(\cdot\right), R_t\right) \leq (T-t)\right\} \geq \alpha\left(t\right) \\ \Pr\left\{ \sum\limits_{n \in J(R_t) \cup \digamma_t} \ \sum\limits_{i=l(n,t)+1}^{l_{\min}(n)} S_n\left(i\right) \leq M \cdot (T-t) \right\} \geq \beta\left(t\right) \\ \left[\sum\limits_{n \in J(R_t) \cup \digamma_t} \ \sum\limits_{i=l(n,t)+1}^{l_{\min}(n)} N_a\left(n,i\right) \cdot c\left(n\right) \right] \leq B\left(t\right) \end{array} \right. \right\} \tag{5.1}$$

$$X(T)=\left\{x_T \;\middle|\; \begin{array}{c} x_T=(l(1,T),...,l(n,T),...,l(N,T))\in\{\mathbb{N}\cup\{-1\}\}^N \text{ and} \\ l_{\min}(n)\leq l\,(n,T)\leq L_{\max}(n,t),\forall n\in\{1,...,N\}\text{s.t. } l(n,T)\geq 0 \end{array}\right\}$$

For $t\in\{0,...T-1\}$, $x_t=(l(1,t),...,l(n,t),...,l(N,t))\in X(t)$ the action a_t in the state x_t decides how many other levels above x_t we want to perform. The performance level up to which the design task n may be solved after the action a_t was taken is $(l(n,t)+a_t(n))^+\overset{def}{:=}\max(l(n,t)+a_t(n),0)$.

$$A_t\,(x_t)=\left\{a_t \;\middle|\; \begin{array}{c} a_t=(a_t(1),...,a_t(n),...,a_t(N))\in\mathbb{N}^N \\ l(n,t)+a_t(n)\leq L_{\max}(n,t),\forall n\in\{1,...,N\}\text{ s.t. } l(n,t)\geq 0 \\ \Pr\left\{C\left(t,(l(n,t)+a_t(n))^+,R_t\right)\leq(T-t)\right\}\geq\alpha\,(t) \\ \Pr\left\{\sum\limits_{n\in J(R_t)\cup F_t}\;\sum\limits_{i=l(n,t)+1}^{(l(n,t)+a_t(n))^+}S_n\,(i)\leq M\cdot(T-t)\right\}\geq\beta\,(t) \\ \left[\sum\limits_{n\in J(R_t)\cup F_t}\;\sum\limits_{i=l(n,t)+1}^{(l(n,t)+a_t(n))^+}N_a\,(n,i)\cdot c\,(n)\right]\leq B\,(t) \end{array}\right\} \quad (5.2)$$

An important thing related to the state and action space descriptions is that they *are sensitive to the arrival of unplanned design tasks up to the upper bound* N. If at a decision point t an unplanned design task n just appeared it will be numbered in an increasing way from $\overline{N}+N(t-1)$ up to N, and $l(n,t)$ will be changed from -1 to zero. Thus, at the next decision point their corresponding actions might be greater than zero, and the team of the engineers might start working on any of the unplanned design tasks.

We notice as well that in the second probability constraint from (5.1) and (5.2) the total workload is computed by adding the remaining solving times of both the design tasks from R_t and of the expected unplanned design tasks, up to their minimal and respectively maximal decided performance level. The computation can be easily done under the assumption of review period-dependent Poisson arrival processes of statistically identical design tasks. Thus, the sum of the independent remaining solving times of the design tasks leads to an Erlang distributed random variable.

The immediate **rewards:** $\forall\; x_t\in X(t)$, $x_{t+1}\in X(t+1)$, and $a\in A_t\,(x_t)$, the immediate reward is

$$\rho_t\,(x_t,a,x_{t+1})=0,\forall t=0,...,T-1$$

and the final reward is

$$\rho_T(x_T)=\left\{\begin{array}{cc} 0, & \text{if } \exists\; 1\leq n\leq N,\quad l\,(n,T)<l_{\min}\,(n) \\ f\,(\max(x_T,0)) & \text{otherwise} \end{array}\right.,$$

where $f(\cdot):\mathbb{N}^N\to\mathbb{R}_+$ is a *nondecreasing function with respect to the partial order on* $\mathbb{N}^N$(i.e. we say that $x\leq\tilde{x}$, $x,\tilde{x}\in\mathbb{N}^N$, if $x\,(n)\leq\tilde{x}\,(n)$, $\forall 1\leq n\leq N$). This type of reward function is very general, describing the market value of a new product which has to fulfill several customer needs including the analytical cumulative market payoff functions of (Yoshimura, 1996); (Huchzermeier and Loch, 2001), and (Askin and Dawson, 2000).

The transition probabilities: In order to obtain a Markov decision process formulation of the aggregate decision problem, the transitions should depend only on the decision time point, the observed state and the chosen action and not on the whole history of the process. The probability that the next state is x_{t+1}, given that the state at the beginning of stage t is x_t and that the action $a_t\in A_t(x_t)$ is chosen, will be the nonstationary probability $p_t(x_t,a_t,x_{t+1})$.

Due to the partial order of precedence relations induced by the $T-$partite acyclic graph of design tasks, $p_t(x_t, a_t, x_{t+1}) = 0$ if one of the following holds:

1. $\exists n_1, n_2 \in \{1, ..N\}$ s.t. n_1 is a predecessor of n_2 in the partial order, $l\left(n_2, t+1\right) \neq 0$ and $l\left(n_1, t+1\right) \neq l\left(n_1, t\right) + a_t(n_1)$;

2. $\exists n \in \{1, ...N\}$ s.t. $l\left(n, t+1\right) \neq 0$ and $n \notin \Lambda_t^0 \cup \ldots \cup \Lambda_t^t \cup \Omega(t-1)$.

Otherwise, the transition probabilities can be calculated, by estimating during each review period the results of team of engineers with the simple model built in Section 5.2.

We now wish to formally give the transition probabilities of the Markov decision process. There are mainly two things which can happen during a transition from one state to another:

- some design tasks are performed at some performance levels above their previous ones and at most equal to the ones target, set by the action.

- new design tasks can appear due to technical uncertainties

We recall that each state is a vector storing the current performed levels (zero or positive), and the value -1 for design tasks not in the system (i.e. unplanned, but theoretically allowable to appear). Now, these two independent processes will respectively modify on the one hand the zero or greater than zero components, and on the other hand, the -1 components, making them become zero, i.e. new freshly arrived design tasks and not processed yet at all. During each review period $[t, t+1)$ we take into account at most a number K_t of new arrived design tasks. Moreover, as these new unplanned design tasks arrive, they "receive" indices in increasing order, so actually for any state, all its say n_0 components strictly greater than -1 are at the beginning, from the first one to the n_0-th one, the other ones to the end being equal to -1. The outcome of the first process is estimated using the estimation model from Section 5.2.

Proposition 19 *Let us assume that during each review period $[t, t+1)$ we take into account at most a number K_t of unplanned design tasks which arrive with a probability greater than a given threshold $\vartheta(t) > 0$ (i.e. $Pr\{$at most K_t arrivals during review period $[t, t+1)\} \geq \vartheta(t)$) and $\sum_{t=0}^{T-1} K_t = N - \overline{N}$. Then the transition probabilities of the Markov decision process constructed above are given by*

$$p_t\left(x_t, a_t, x_{t+1}\right) =$$

$$= \begin{cases} p_t\left(\xi_{t+1} - x_t, i_{a_t}\right) \cdot \frac{[\varsigma(t)]^k}{k!} e^{-\varsigma(t)} & , x_t \leq x_{t+1} \leq x_t + a_t + z_t \text{ and} \\ & k \leq K_t \stackrel{def}{:=} card\{n | z_t(n) = 1\} \\ 0 & , \quad \text{otherwise} \end{cases}$$

where

$\xi_{t+1}(n) = \begin{cases} x_{t+1}(n) & , x_t(n) = l(n,t) \geq 0 \\ -1 & , \quad \text{otherwise} \end{cases}$, $\forall n \in \{1, \ldots, N\}$ *(i.e. x_{t+1} restricted to the progress made on the previous existing design tasks),*

i_{a_t} is the number of nonempty components in the action a_t, corresponding to concurrent design tasks (i.e. the number of concurrent design tasks given to the team of engineers), and

$$z_t(n) = \begin{cases} 0 & , x_t(n) = l(n,t) \geq 0 \\ 1 & , x_t(n) = -1, \forall n \in \{n_0+1, \ldots, n_0+K_t\}, \textit{ and } x_t(n_0) = l(n_0,t) \geq 0 \\ 0 & , \textit{ otherwise} \end{cases},$$

$\forall n \in \{1, \ldots, N\}$ *(i.e. counts the maximal number of unplanned design tasks that might be included in the project during the current review period). These transition probabilities can be fast numerically approximated by the Euler Inverse Laplace Transform method (Abate and Whitt, 1995).*

Proof. As we have said before, during each review period there are two independent processes that take place, and we will make use of the estimation model built in Section 5.2. The first process is the solving process of the design activities from the design tasks performance levels decided by the action a_t. The second process that takes place is the Poisson arrival process of unplanned design tasks to be added to the NPD project description. According to our assumption the number of new design tasks that are taken into account at the end of the review period $[t, t+1)$ is less than K_t, with $\sum_{k=0}^{K_t} \frac{[\varsigma(t)]^k}{k!} e^{-\varsigma(t)} \geq \vartheta(t)$ (i.e. the length of each review period is 1).

To estimate the solving process we use the simple model built in Section 5.2. Its basic assumptions are:

- the design tasks are sequenced, their order reflecting the optimality criteria of the detailed planning level scheduler,
- the team of engineers works with all its capacities as one processor on all the design tasks
- unlike machines, human beings are able to perceive the concurrency and the relative urgency of design tasks, and at any time instant the time will either work with the probability p on the planned activities of the design tasks in the order given by the scheduler, or, with a probability $1-p$ will try to re-prioritize the design tasks instead of working on them.

Let $\Xi(k, i_{a_t})$ be the random variable giving the solving time of the allocated design tasks. In the line of reasoning of Proposition 18, Section 5.2, for $i_{a_t} > 2$, $\Xi(k, i_{a_t})$'s cumulative distribution function equals $F_{\Xi(k,i_{a_t})}(h) = \Pr\{\Xi(k, i_{a_t}) \leq h\} = p \cdot \Pr\{C_{\max}(k) \leq h\} + (1-p) \cdot \Pr\{Y_{Erlang(i_{a_t}-1)} \leq h\}$, where $C_{\max}(k)$ is the makespan of queueing system of M parallel servers with a common queue having a series of $k = \sum_{n=1}^{N} \sum_{i=l(n,t)+1}^{l(n,t+1)} N_a(n,i)$ of planned activities, and $(1-p)$ is the probability that during the solving process the team of engineers instead of working on the design tasks reconsider their order. They re-prioritize the order of the design tasks that can be performed in parallel, by successively comparing the first activities to be done from each of the other allocated tasks. Comparing design tasks requires capacity, which can be modelled as an activity with exponential distribution with mean $1/\eta$. This defines a complex queueing system, where the planned activities have a common processing time, exponentially distributed with mean $\frac{1}{\mu}$, and their solving process is disturbed by a $\lambda(t)$-Poisson arrival of unplanned activities ($\lambda(t)/\mu < M$). The arrival process of unplanned activities stops when all the planned activities are finished.

Following the proof of Proposition 18, Section 5.2 the cumulative distribution function of Ξ can be computed numerically.

We set, as said before, $h = t - (t-1) = 1$.

We consider that the transition probabilities $p_t\left(x_t, a_t, \xi_{t+1}\right)$ should give the probability that the solving time of the design tasks up to the performance levels from x_{t+1} is contained in an interval of length ε around 1 (i.e 1 is the length of a review period). However, they do not equal exactly $\int_{1-\varepsilon/2}^{1+\varepsilon/2} f_{\Xi(k,i_{a_t})}(z)dz$ because when being in the state x_t, the same number of planned activities to be done can be obtained for different $x_{t+1} \in X(t+1)$, which we also assume to be equiprobable. Therefore in the previous integral, $f_{\Xi(k,i_{a_t})}(z)$ which actually is $p \cdot f_{C_{\max}(k)}(z) + (1-p) f_{Y_{Erlang(i_{a_t}-1)}}(z)$, has to be divided by the number of possible ways of arriving at these equiprobable x_{t+1}.

Thus we have that $p_t\left(x_t, a_t, x_{t+1}\right) \stackrel{def}{:=}$

$$\int_{1-\varepsilon/2}^{1+\varepsilon/2} \left[\frac{p \cdot f_{C_{\max}(k)}(z) + (1-p) f_{Y_{Erlang(i_{a_t}-1)}}(z)}{\varsigma(N_a\left(n,i\right); 1 \le n \le N, l\left(n,t\right)+1 \le i \le l(n,t+1))} \right] dz$$

$$= \frac{p \cdot \int_{1-\varepsilon/2}^{1+\varepsilon/2} f_{C_{\max}(k)}(z)dz + (1-p) \int_{1-\varepsilon/2}^{1+\varepsilon/2} f_{Y_{Erlang(i_{a_t}-1)}}(z)dz}{\varsigma(N_a\left(n,i\right); 1 \le n \le N, l\left(n,t\right)+1 \le i \le l(n,t+1))}$$

$$\simeq \varepsilon \left[\frac{p \cdot f_{C_{\max}(k)}(1) + (1-p) f_{Y_{Erlang(i_{a_t}-1)}}(1)}{\varsigma(N_a\left(n,i\right); 1 \le n \le N, l\left(n,t\right)+1 \le i \le l(n,t+1))} \right]$$

depends only on the following parameters: $\sum_{n=1}^{N} \sum_{i=l(n,t)+1}^{l(n,t+1)} N_a\left(n,i\right)$, μ, $\lambda(t)$, i_{a_t}, where ε is small and $\varsigma(N_a\left(n,i\right); 1 \le n \le N, l\left(n,t\right)+1 \le i \le l(n,t+1))$ is the number of partitions of $\sum_{n=1}^{N} \sum_{i=l(n,t)+1}^{l(n,t+1)} N_a\left(n,i\right)$ into multiples of the number of activities per level. If for each already available design task $1 \le n \le N$ we have that $N\left(n,i\right)$ is constant for any $1 \le i \le L_{\max}(n,t))$ then the correction factor can be easily computed. The term

$$\frac{\int_{1-\varepsilon/2}^{1+\varepsilon/2} f_{C_{\max}(k)}(z)dz}{\varsigma(N_a\left(n,i\right); 1 \le n \le N, l\left(n,t\right)+1 \le i \le l(n,t+1))}$$

depends only on the μ, $\lambda(t)$, and on the number of activities between the states x_t and x_{t+1}, while the second term depends on the i_{a_t} μ, $\lambda(t)$, and on the number of activities between the states x_t and x_{t+1}. Thus, one can write for μ, $\lambda(t)$ constant that

$$p_t\left(x_t, a_t, \xi_{t+1}\right) = p_t(\xi_{t+1} - x_t, i_{a_t}) = p \cdot P_t(\xi_{t+1} - x_t) + (1-p) \cdot \frac{P_t(i_{a_t})}{\varkappa(\xi_{t+1} - x_t)}$$

Now we have to prove that $\sum_{x_{t+1} \in X(t+1)} p_t\left(x_t, a_t, x_{t+1}\right) \le 1$ in order to show that we obtain indeed a Markov process.

First we remark that in the queueing system described in Proposition 18, Section 5.2 both the arrival of new activities and the re-prioritization process stop when the last of the k planned activity is solved. Thus, after a random finite number Q of solved activities the system stops, and

$$\Pr\{t < \Xi(K, i_{a_t}) \le t + h | K = k\} = \Pr\{t < S_Q < t + h | Q = q\}$$

where S_q is the time of the q'th event of the queueing system (the q'th activity solved). The queueing system under study is a version of an M/G/s queuing system, which has the probability of two or more events in $(t, t+h)$ equal to an $o(h)$. Why? Because the system without re-prioritization was an M/M/s system (i.e. the departure process of the solved activities is Poisson, and hence has this property), and the re-prioritization delays the solving process without adding any solved activity. Consequently we can write that

$$\begin{aligned}\Pr\{t < S_q < t+\varepsilon\} &= \Pr\{\pounds(t) = q-1, \text{one event in } (t, t+\varepsilon)\} + o(\varepsilon) \\ &= \Pr\{\pounds(t) = q-1\} \Pr\{\text{one event in } (t, t+\varepsilon)\} + o(\varepsilon)\end{aligned}$$

where $\pounds(t)$ is the total number of activities that have been solved up to time t. So $\sum_k \Pr\{t < \Xi(K, i_{a_t}) \leq t+\varepsilon | K = k\} = \sum_q \Pr\{\pounds(t) = Q-1 | Q = q\} \Pr\{\text{one event in } (t, t+\varepsilon)\} + o(\varepsilon)$. If we take $t = 1$ and we divide both sides of the last equation by ε, we obtain for small ε that $\sum_k \int_{1-\varepsilon/2}^{1+\varepsilon/2} f_{\Xi(k, i_{a_t})}(z) dz \simeq 1$.

The arrival process of unplanned design tasks is a Poisson process of rate $\varsigma(t)$. Then $\Pr(k \text{ arrived design tasks in} [t, t+1)) = \frac{[\varsigma(t)]^k}{k!} e^{-\varsigma(t)}$ and $\sum_{k=0}^{K_t} \frac{[\varsigma(t)]^k}{k!} e^{-\varsigma(t)} \leq 1$.

Thus, we obtain a stochastic process since by imposing for each pair (x_t, a_t) upper limits on the number of planned activities to be done the transition probabilities matrix is sub-stochastic, i.e.

$$\begin{aligned}&\sum_{x_t \leq x_{t+1} \leq x_t + a_t + z_t} p_t(x_t, a_t, x_{t+1}) \\ &= \sum_{k=0}^{K_t} \left[\sum_{x_t \leq \xi_{t+1} \leq x_t + a_t} p_t(x_t, a_t, \xi_{t+1}) \right] \cdot \Pr(k \text{ arrived design tasks in } [t, t+1)) \leq 1\end{aligned}$$

for any $x_t \in X(t)$, where $card\{n | z_t(n) = 1\} = K_t$. ■

In relation to the way of computing the transition probabilities two issues should be discussed. The first one is that to compute $p_t(x_t, a_t, x_{t+1})$ we do not even need to limit the number of unplanned design tasks arrival. However, having a finite number of engineers and desiring to finish with a probability higher than a given safety margin lead us to a finite solving capacity until the deadline. Thus, there should exist an upper bound N on the total number of design tasks included in the NPD project definition. Moreover, limiting during each review period the number of new design tasks to be added to the project definition will help us obtain structural results for the case of NPD without precedence constraints (see Chapter 7).

The second issue is that while computing in a real life situation the transitions probabilities $p_t(x_t, a_t, x_{t+1})$ of our Markov decision process we might not need the correction factor $\varsigma(N_a(n,i); 1 \leq n \leq N, l(n,t)+1 \leq i \leq l(n, t+1))$. Why? Because if we might have the same number of planned activities to be done for different x_t, x_{t+1}, we also might have some $k \in \{0, \ldots, \infty\}$ that cannot be written as a sum of the type $\sum_{n=1}^{N} \sum_{i=l(n,t)+1}^{l(n,t+1)} N_a(n,i)$, for some $x_{t+1} \in X(t+1)$. Moreover, due to this phenomenon

we might skip quite a number of small values of k, while repeating its higher values which give a much lower probability of being finished in a unit of time. In conclusion, for a higher number of activities per level, and a small μ, the need of a correction factor for the actual computation of the transition probabilities is purely theoretical.

Corollary 20 *If we assume that there is no unplanned design tasks arrival process, the transition probabilities of the Markov decision process constructed above are given by*

$$p_t\left(x_t, a_t, x_{t+1}\right) = p_t\left(x_{t+1} - x_t, i_{a_t}\right) =$$

$$= \begin{cases} p\left(\sum_{n=1}^{N} \sum_{i=l(n,t)+1}^{l(n,t+1)} N_a\left(n,i\right), \mu, \lambda(t), \eta, i_{a_t}\right) & , x_t \le x_{t+1} \le x_t + a_t \\ 0 & , \quad \textit{otherwise} \end{cases}$$

where i_{a_t} is the number of nonempty components in the action a_t, corresponding to concurrent design tasks (i.e. the number of concurrent design tasks given to the team of engineers) and can be fast numerically approximated by the Euler Inverse Laplace Transform method (Abate and Whitt, 1995).

Expected total reward criterion: The expected cost of a policy π, starting from in initial state $x_0 = 0_{\mathbb{R}^N}$ is $v_0^{\pi}(x_0) = E_{x_0}^{\pi}\left[\sum_{t=0}^{T-1} \rho_t\left(x_t, a\right) + \rho_T(x_T)\right]$, where the expected reward during the time interval $[t, t+1)$ is:

$$\rho_t\left(x_t, a\right) = \sum_{x(t+1)} \rho_t\left(x_t, a, x_{t+1}\right) p_t\left(x_t, a_t, x_{t+1}\right) = 0.$$

We are looking for the maximal expected total reward.

Remark 21 *The targets on design tasks realization given by the actions of the Markov decision process play also another role. We recall from Chapter 3, Section 3.4 that organizational psychology research (see for a review (Wickens and Hollands, 1999), (Oorschot, 2001)) shows that there is a curvilinear dependency of the engineers productivity on their workload. For this reason the team of engineers should receive not only a maximal requirement of work during one review period, but also a minimal one. Thus, one can restrict even more the action space description:*

$$K_{\min} := \left\{ a_t \in A_t(x_t) \middle| \Pr\left\{ \sum_{n \in J(R_t) \cup F_t} \sum_{l=l(n,t)+1}^{l(n,t)+a_t(n)} S_n(l) \ge M \cdot 1 \right\} \ge \rho > 0 \right\} \quad (5.3)$$

where $\rho \in (0, 1]$. However, then one should consider very carefully the choices made for the parameters ρ and $\beta(t)$, the risk being the one of obtaining an empty action set.

An even more restrictive and simpler way of introducing the minimal work restriction will be to consider only the actions $a_t \in A_t(x_t)$ such that

$$K_{\min} := \left\{ a_t \in A_t(x_t) \middle| \begin{array}{c} a_t \ge a_{\min}(t) \\ \Pr\left\{ \sum_{n \in J(R_t) \cup F_t} \sum_{l=l(n,t)+1}^{l(n,t)+a_{\min}(n,t)} S_n(l) \ge M \cdot 1 \right\} \ge \rho > 0 \end{array} \right\} \quad (5.4)$$

where $a_{\min}(\cdot,t) : \{1,...N\} \rightarrow \mathbb{N}^*$ *denotes the function giving the minimal requirement of work to be done during the review period* t*. This function is time dependent such that* $a_{\min}(n,t) > 0$ *for* $n = 1,...,\overline{N}$ *and* $a_{\min}(n,t) = 0$ *if* $n \notin J(R_t)$ *(i.e. we cannot decide to work on the design tasks not planned or not arrived yet).*

Remark 22 *The above constructed Markov decision process has a partially ordered state space, and action space, which can be viewed as bounded subset without holes of the lattice* $\{\mathbb{N} \cup \{-1\}\}^N$ *and respectively* $\mathbb{N}^N$ *(see Definition 33, from Chapter 6). The partial order considered on them is the one induced by the order on* $\mathbb{N}^N$*:* x*,*$y \in \mathbb{N}^N$ *and* $x \leq y$ *iff* $x_n \leq y_n$*,* $\forall n = 1,...,N$*. Since there is a total possible number of activities to be performed during one review period (depending on the values of* $\alpha(t), \beta(t), B(t)$*), there will during each review period a total maximal number of levels to be done for each design task. Thus, the action space will be upper bounded.*

Proposition 23 *Close to optimal and optimal policies for our Markov decision process may be obtained. by the backward induction algorithm.*

Proof. The horizon, state and action space are finite, and the reward function is linearly additive. The backward induction algorithm can be used to find the optimal policies and value functions (Puterman, 1994). Also a non-stationary version of the sequential backward approximation algorithm for finite horizon problems (Bertsekas and Tsitsiklis, 1996) can be used to find policies close to optimal. ■

However, no reasonable computational results can be derived directly from this Markov decision model, due to the curse of dimensionality. Also, we could not derive analytical results on the structure of the optimal policies. Thus, we decided to investigate analytically the structural properties of the two basic particular cases of this problem. These cases will be used in Chapter 9 to suggest possible ways of deriving heuristic solutions for the general problem.

5.4 Particular cases of the nonstationary Markovian control of NPD projects

The first case considered is the one of an NPD project consisting of a set of concurrent design tasks (i.e. without precedence constraints), while the second one is the case of an NPD project consisting of a sequence of design tasks. In order to simplify the problem description, we have chosen not to consider the budget constraint in the particular cases. This restriction was a refinement of the workload restriction, and we conjectured that its exclusion will not essentially affect the structure of the optimal policies.

For enabling the mathematical tractability of the two particular cases we have made two more general assumptions. The first one was that no arrival of unplanned design tasks takes place in the NPD project consisting of a sequence of design tasks. In the other NPD case, we only reduced our assumptions regarding the arrival process as well as the type of newly arrived design tasks to the conditions of Proposition 19 (i.e during each review period $[t, t+1)$ we take into account at most a number K_t of unplanned design tasks which arrive with a probability greater than a given threshold $\vartheta(t) > 0$ and $\sum_{t=0}^{T-1} K_t = N - \overline{N}$).

The second assumption was that there is an identical number of activities for all performance levels of the same design task (i.e. $N_a(n,l) := n_a(n)$, for any $n = 1, ..., N$, and $l = 1, ..., L_{\max}(n,t)$). Thus, assuming the previous levels already solved, the solving times of all the performance levels of the same design task will be i.i.d. random variables distributed Erlang-$(n_a(n), \mu)$.

5.4.1 NPD without precedence constraints

In this subsection we restrict the nonstationary Markov decision model from Section 5.3 to a NPD project without precedence constraints, consisting of N concurrent design tasks. All the design tasks are allocated to resources (engineers) from the first review period. New design tasks may arrive during each review period, and they will become available to the team of engineers at the beginning of the next review period. They are concurrent with the initial design tasks allocated to the team of engineers. Their arrival is given by a Poisson review period-dependent arrival process.

Due to the curvilinear dependency between the productivity and the workload of the team of engineers, the targets on design tasks realization have to satisfy a minimal requirement of work during one review period (see Remark 21, equation (5.4)). As mentioned in the beginning of this section, *no budget constraint will be considered i*n this Markov decision process, and *all performance levels of one design task will have the same number of activities. An important* supplementary *assumption concerning only this NPD case is related to the maximal performance levels of the design tasks.*

Supplementary Assumption: *For each design task n its maximal performance level $L_{\max}(n,t)$ is large enough so that even if the team of engineers will work with all capacities on it, $L_{\max}(n,t)$ cannot be achieved earlier than the deadline with the probability $\min_{t=0,...,T} \beta(t)$. However this holds only for planned or already arrived design tasks. Since N is an upper bound, we have to set to zero all the parameters depending on a virtual $n \in \{\overline{N}+1, \ldots, N\}$.*

This assumption requires no artificial preset bounding of the action spaces at each decision point. Moreover since there is a total possible number of activities to be performed during one review period (depending on the values of $\beta(t)$), there will a total possible number of levels to be done for each design task. Thus, the action space will have as upper bound a nondecreasing function.

This enlargement of the action space will prove to be useful in Chapter 7, since otherwise we might not be able to prove in all the cases the existence of structured optimal policies.

All the other assumptions and notation are given in Section 5.3.

The reduced state space and the action space: The state set $X(t)$ at moment t and the action set $A_t(x_t)$ in the state $x_t \in X(t)$ are defined by simplifying the ones from the general Markov decision process from Section 5.3, by keeping only the workload constraint among the main three ones:

- the target performance level of each planned or newly arrived design task n is greater than $\min(l_{\min}(n,t), l(n,t))$, and smaller than $L_{\max}(n,t)$
- the remaining workload of the team of engineers should not exceed their maximal solving capacity with the probability greater than safety margin $\beta(t)$.

Similar with the general Markov decision process from Section 5.3 for the state space the workload constraint is computed using the minimal performance levels, $l_{\min}(\cdot,t)$, while for the action space it is used the target performance levels, $(l(\cdot,t)+a(\cdot,t))^{+}$.

For $t=0$: $X(0)=\{x_0\}=(0_{\mathbb{R}^{\overline{N}}},-1_{\mathbb{N}^{N-\overline{N}}})$.

For $t\in\{1,\ldots,T\}$ the state $x_t\in X(t)$ describes how many performance levels $l(n,t)$ were solved for each design task n:

$$X(t)=\left\{x_t\left|\begin{array}{c}x_t=(l(1,t),...,l(n,t),...,l(N,t))\in\{\mathbb{N}\cup\{-1\}\}^N \text{ and}\\ -1\le l\,(n,t)\le L_{\max}(n,t),\ n\in\{1,...,N\}\\ l(n,t)\ge 0,\forall n\in\{1,...,N\}\text{ s.t. } L_{\max}(n,t)>0\\ \Pr\left\{\sum\limits_{n\in\{1,...,\overline{N}+N(t-1)\}\cup F_t}\ \sum\limits_{i=l(n,t)+1}^{l_{\min}(n)}S_n\le M\cdot(T-t)\right\}\ge\beta\,(t)\end{array}\right.\right\} \tag{5.5}$$

$$X(T)=\left\{x_T\left|\begin{array}{c}x_T=(l(1,T),...,l(n,T),...,l(N,T))\in\{\mathbb{N}\cup\{-1\}\}^N \text{ and}\\ l_{\min}(n)\le l\,(n,T)\le L_{\max}(n,t),\forall n\in\{1,...,N\}\text{s.t. } l(n,T)\ge 0\end{array}\right.\right\}$$

For $t\in\{0,...T-1\}$, $x_t=(l(1,t),...,l(n,t),...,l(N,t))\in X(t)$ the action a_t in the state x_t decides how many other levels above x_t we want to perform. The performance level up to which the design task n may be solved after the action a_t was taken is $(l\,(n,t)+a_t(n))^{+}$.

$$A_t\,(x_t)=\left\{a_t\left|\begin{array}{c}a_t=(a_t(1),...,a_t(n),...,a_t(N))\in\mathbb{N}^N\\ 0<a_{\min}(n,t)\le a_t(n),\forall n\in\{1,...,\overline{N}+N(t-1)\}\\ l(n,t)+a_t(n)\le L_{\max}(n,t),\forall n\in\{1,...,N\}\text{ s.t. } l(n,t)\ge 0\\ \Pr\left\{\sum\limits_{n\in\{1,...,\overline{N}+N(t-1)\}\cup F_t}\ \sum\limits_{i=1}^{a_t(n)}S_n\le M\cdot(T-t)\right\}\ge\beta\,(t)\end{array}\right.\right\} \tag{5.6}$$

We recall from Section 5.3 that the state and action space descriptions are sensitive to the arrival of unplanned design tasks, up to the upper bound N.

Also the minimal work requirement function is time dependent $a_{\min}(n,t)>0$ for $n=1,...,\overline{N}$ and $a_{\min}(n,t)=0$ if $n\notin\{1,...,\overline{N}+N(t-1)\}$ since we cannot decide to work on the design tasks not planned or not arrived yet.

In the probability constraint from (5.6), the total workload is computed by adding up the solving times of the total number of concurrent design tasks during the whole project. The restriction has a simplified form since $\sum\limits_{i=l(n,t)+1}^{(l(n,t)+a_t(n))^{+}}S_n(i)=\sum\limits_{i=1}^{a_t(n)}S_n$.

In the state and action space descriptions (5.5, (5.6) of this particular case we gave up to the completion time constraint, since the completion time $C\,(t,l\,(\cdot,t)+a_t(\cdot),R_t)$ is equal to $\max\limits_{n\in\{1,...,\overline{N}+N(t-1)\}\cup F_t}\ \sum\limits_{i=1}^{a_t(n)}S_n$, and thus it becomes similar to the workload inequality. To be coherent with the supplementary assumption made above for this particular NPD case, and avoid an in depth discussion on the reasonable assumptions on the relationships in between $\alpha(t)$, $\beta(t)$, and M we kept only the most natural constraint in this particular case: the workload inequality.

5.4.2 Sequential NPD

In this subsection we restrict the nonstationary Markov decision model from Section 5.3 to a NPD project consisting of a sequence of $N = \overline{N} \leq T$ design tasks. As in Section 5.3 we consider that the team of engineers will work together on each of the design tasks. We focus on a NPD project with precedence constraints, consisting of N sequential design tasks. We assume that the design tasks in the given sequence can be performed by the team of engineers one after the the other and they can start only at the beginning of a review period. The team may start working on first design task from the first review period. No arrival of unplanned design tasks takes place in the case of the sequential NPD project (i.e. $N = \overline{N}$). As mentioned in the beginning of the current section, no budget constraint will be considered in the Markov decision process. *An important* supplementary *assumption concerning only this NPD case is related to the form of the final reward.*

Supplementary Assumptions:

1) We do not consider anymore a time dependent safety margin for the probability of completing the project before the deadline $\alpha(t) = \alpha$, $t \in \{1, ...T-1\}$. However, our results can be generalized for a decreasing sequence of $\alpha(t)$, $t \in \{1, ...T-1\}$.

2) The final reward is of the more restrictive form

$$\rho_T(x_T) = \begin{cases} 0, & \text{if } \exists\ 1 \leq n_0 \leq N, \quad l(n_0, T) < l_{\min}(n_0) \\ \sum\limits_{n=1,..,N} l(n,T) \cdot V(n) & \text{otherwise} \end{cases}$$

giving a linear (weighted additive) cumulated market payoff function in the arguments $V(n)$ which represent the scalable value functions associated with the design tasks. Such a function is similar with the one defined by (Askin and Dawson, 2000).

If we consider a final reward of the form given in (Askin and Dawson, 2000) then

$$\max_{\substack{l(n,T),\\ n\in\{1,...,N\}}} \sum_{\delta=1}^{\Delta} w_\delta \left[\sum_{n=1,..,N} \Theta(n,\delta) \cdot \frac{l(n,T)}{L_{\max}(n)} \right] =$$

$$= \max_{\substack{l(n,T),\\ n\in\{1,...,N\}}} \sum_{n=1,..,N} \frac{l(n,T)}{L_{\max}(n)} \cdot \left[\sum_{\delta=1}^{\Delta} w_\delta \cdot \Theta(n,\delta) \right],$$

where $l(n,T)$ is the number of performance levels achieved at the deadline T for the design task n.

Then a simple choice the value per a performance level for the design task n is: $V(n) := \frac{1}{L_{\max}(n)} \cdot \sum\limits_{\delta=1}^{\Delta} w_\delta \cdot \Theta(n,\delta), \forall\ n = 1, ..., n$, where w_δ is the normalized weight of the customer need δ ($\forall\ \delta = 1, ..., \Delta$), and $\Theta(n,\delta)$ is the normalized maximal contribution of the design task n in fulfilling the customer need δ ($\Theta(n,\delta) \in [0,1)$; $\forall\ n = 1, ..., N, \forall\ \delta = 1, ..., \Delta$; $\sum\limits_{\delta=1}^{\Delta} \Theta(n,\delta) = 1$).

All the other assumptions and notation are given in Section 5.3.

The reduced state space and the action space: The state set $X(t)$ at moment t and the action set $A_t(x_t)$ in the state $x_t \in X(t)$ are defined by simplifying the ones from the general Markov decision process from Section 5.3, by keeping only the completion time constraint among the main three ones:

- the target performance level of each planned design task n is smaller than $L_{\max}(n,t)$ and greater than $\min(l_{\min}(n,t), l(n,t))$,
- the completion time of the NPD project must be smaller than the remaining time until the deadline with a probability greater than safety margin $\alpha(t)$.

Similar with the general Markov decision process from Section 5.3 for the state space the completion time constraint is computed using the minimal performance levels, $l_{\min}(\cdot,t)$, while for the action space it is used the target performance levels, $l(\cdot,t)+a(\cdot,t)$.

For $t=0: X(0)=\{x_0\}=\{0_{\mathbb{R}^N}\}$

For $t \in \{1,...T-1\}$ the state $x(t) \in X(t)$ indicates $n(t)$ as the current not yet finished design task from the initial sequence, and describes how many performance levels were solved for it, as well as for the finished design tasks.

$$X(t)=\left\{ x_t \left| \begin{array}{c} x_t=(l(1,t),...,l(n(t),t),0,...,0)\in\mathbb{N}^N, n(t)\in\{1,\dots,N\} \\ 0<l(i,t)\le L_{\max}(i), i\in\{1,...,n(t)\} \\ \Pr\left\{\sum\limits_{n=n(t)+1}^{N}\sum\limits_{i=1}^{l_{\min}(n)} S_n+\sum\limits_{i=l(n,t)+1}^{l_{\min}(n)} S_{n(t)}\le M(T-t)\right\}\ge\alpha \end{array} \right. \right\} \tag{5.7}$$

For $t=T$:

$$X(T)=\left\{ x_T \left| \begin{array}{c} x_T=(l(1,T),...,l(N,T))\in\mathbb{N}^N, \\ l_{\min}(n)\le l(n,T)\le L_{\max}(n,T), n\in\{1,...,N)\} \end{array} \right. \right\}$$

For $t \in \{0,...,T-1\}$ and $x_t=(l(1,t),...,l(n(t),t),0,...,0)\in\mathbb{N}^N, n(t)\in\{1,\dots,N\}$ the action a_t decides how many other levels above x_t we want to perform. The level up to which the design task n may be performed after the action a_t was taken is: $l(n,t)+a_t(n)$.

$$A_t(x_t)=\left\{ a_t \left| \begin{array}{c} a_t=(0,...,0,a_t(n(t)),...,a_t(N))\in\mathbb{N}^N \\ 0\le l_{\min}(n)-l(n,t)\le a_t(n), \forall n\in\{1,...,N\} \\ l(n,t)+a_t(n)\le L_{\max}(n,t) \\ \Pr\left\{\sum\limits_{n=n(t)+1}^{N}\sum\limits_{l=1}^{a_t(n)} S_n+\sum\limits_{l=1}^{a_t(n(t))} S_{n(t)}\le M(T-t)\right\}\ge\alpha \end{array} \right. \right\} \tag{5.8}$$

In the probability constraint from (5.8) the analytical evaluation of the completion time distribution of a sequence of design tasks is given by the convolution product of the distributions of the design tasks solving times. This restriction has a simplified form because $\sum\limits_{i=l(n(t),t)+1}^{l(n(t),t)+a_t(n(t))} S_{n(t)}(i)=\sum\limits_{i=1}^{a_t(n(t))} S_{n(t)}$. Moreover, in a sequential NPD the team of engineers solves the design tasks one by one, and never comes back to a design task for which it was achieved the performance level required by the controller.

In the the state and action space descriptions, (5.7) and (5.8), of this particular case we gave up to the workload constraint since due to the analytical form of the completion time distribution of a sequence of design tasks the two inequalities become similar again. In order

to avoid more assumptions on the relationships in between α, β, and M we kept only the most natural constraint in this particular case: the completion time distribution of a sequence of design tasks inequality. The former workload inequality may be used to obtain for each design task its maximal achievable performance level.

Chapter 6

Supplementary Chapter–Monotonic and Weakly Monotonic Nondecreasing Optimal Policies

Monotonic optimal policies constitute one of the most well-known and useful characterizations (Chapter 8 in (Heyman and Sobel, 1984), or Section 4.7.3 in (Puterman, 1994)) for reducing the exponential growth with respect to the size of a sequential decision process, and enabling the derivation of numerical solutions. Our approach for obtaining weak monotonicity is based on the monotonicity research of (White, 1980; Topkis, 1998). We formulate the problem in a dynamic programming setting, and we show that the optimal policy follows a weakly monotonic optimal control by establishing the supermodularity of the objective function. This is a new result, extending the monotonicity theory and partial ordering programming techniques to bounded subsets without holes of integer vector lattices.

The main structural property making all these mathematical tools applicable is the presence of partial ordering in the state and respectively action spaces. Partial orderings have been recognized as being important in many fields, and such structures have received an increasing interest and spread out from mathematics to biology, economics and also physics. In economics, the partial orderings, doubled by the lattice programming techniques of (Topkis, 1998), encompass many applications in many production planning models ((Hopenhayn and Prescott, 1992; Garcia and Smith, 2000) for discrete-time production planning with stochastic demand, (Athey, 2002), and (Athey and Schmutzler, 1995) for the analysis of several attributes of a firm's short-run innovative activity).

In physics, the presence of partial orderings stands in the same class as spatial symmetries and Hamiltonian structures for the study of dynamic systems. These properties make them have specific types of behavior (for example, spatial symmetries give rise to conservation laws and multiple bifurcations). As pointed out in (Landsberg and Friedman, 1996), the presence of partial orderings restrains significantly the behavior, yet it allows for interesting trajectories and even chaotic ones. However, due to special consequences, the chaotic be-

havior is unstable and not present for most initial conditions when physically observing the system in the long run. With results such as the ones presented here, the presence of partial orderings can also contribute to the control of such systems, especially if the optimal solution changes non-chaotically upon changes of values for exogenous parameters.

Our approach for obtaining monotonic and weakly monotonic optimal policies is based on the monotonicity research of (White, 1980; Topkis, 1998). This means that we formulate our control problem in a dynamic programming setting, and we try to establish the supermodularity of the objective function. General approaches to monotonicity results can be also found in (Hinderer, 1984), (Hinderer and Stieglitz, 1994), and (Serfozo, 1976), while for results on monotone properties of the optimal control policies in the context of queueing systems we refer to (Naor, 1969), (Weber and Stidham, 1987), (Altman et al., 1992), (Glasserman and Yao, 1994), (Altman and Stidham Jr., 1995), and (Altman and Koole, 1998).

In microeconomics, and in theories of production and consumer choice, supermodularity of an utility function is equivalent to products being *substitutes*. The substitute notion appears very intuitive in the case of a concurrent NPD situation, where after we finish all the design tasks at their minimal performance levels, having done more levels of performance for one design task compensates for doing less performance levels of a different one.

This chapter is organized as follows. In Section 6.1, first we give a short review of the main definitions, and of the basic notions needed to connect the lattice programming techniques with the Markov decision processes (an exhaustive discussion can be found in (Topkis, 1998)). In Section 7.2 we first state the sufficient conditions for the existence optimal monotonic policies. Following their line of reasoning, in Section 6.3 we give new sufficient conditions for weakly-monotonic non-decreasing policies, and in Section 6.4 we present a new weakly monotonic non-decreasing backward induction algorithm and illustrate its behavior through an example. We also compare it to a monotonic backward induction algorithm, which would require a much larger action space, thus a very impractically high time to be run. In section 6.5 we explain previous and new conditions for the robustness of the optimal monotonic nondecreasing and weakly monotonic nondecreasing solutions.

6.1 Preliminaries

The results developed in the following Sections 6.3, 6.4, 6.5 hold for a large class of discrete-time nonstationary Markov decision processes with finite horizon, and can be easily extended in the case of infinite horizon, but due to the computational complexity, the interest of such a generalization is reduced. Let $\mathbb{N}$ the set of natural numbers including zero. We consider as:

The decision time points: They are equidistant and the decision point t corresponds to the beginning of the review period $t+1$. Say $t \in \{0, 1, ..., T-1\}$.

The finite state space and the finite action space: $X\left(t\right) \subset \mathbb{N}^q$ and the action set $A_t\left(x_t\right) \subset \mathbb{N}^q$ in any state $x_t \in X(t)$ for $t \in \{0, ..., T-1\}$.

The immediate rewards: For any $(x_t, a_t) \in X(t) \times A_t\left(x_t\right)$, the immediate reward is $\rho_t\left(x_t, a_t\right)$ and the final reward is $\rho_T(x_T)$.

The transition probabilities: The nonstationary transition probabilities depend only on the decision time point, the observed state and the chosen action and not on the history of the process: $p_t\left(x_t, a_t, x_{t+1}\right)$, for any $(x_t, a_t, x_{t+1}) \in X(t) \times A_t\left(x_t\right) \times X(t+1)$.

We give first some definitions concerning the partial ordered Markov decision processes.

Definition 24 *A binary relation on the sets S and T is a subset R of $S \times T$. When $T = S$, we refer to just the binary relation on the set S.*

We say that the binary relation has the properties of being
antisymmetric - if $(x, y) \in R$ and $x \neq y$ imply $(y, x) \notin R$, for any $x, y \in S$
reflexive -if $(x, x) \in R$, for any $x \in S$
transitive - if (x, y) and $(y, z) \in R$ imply $(x, z) \in R$, for any $x, y, z \in S$

Definition 25 *A set X endowed with a reflexive, antisymmetric and transitive binary relation "$\leq_X$" on X is called a partially ordered set (poset).*

Definition 26 *(see (Puterman, 1994)) A nonstationary Markovian (deterministic) policy is a sequence of decision rules, i.e. $\pi = (a_0, \ldots, a_{T-1})$, where $a_t(x_t) \in A_t(x_t)$ for any $x_t \in X(t)$.*

A nonstationary Markovian (deterministic) policy is said to be a monotonic non-decreasing policy if for any $x_t \leq_{\mathbb{N}^q} \tilde{x}_t$ we have $a_t(x_t) \leq_{\mathbb{N}^q} a_t(\tilde{x}_t)$.

A nonstationary Markovian (deterministic) policy is said to be a weakly monotonic non-decreasing policy if for any $x_t \leq_{\mathbb{N}^q} \tilde{x}_t$ either $a_t(x_t) \leq_{\mathbb{N}^q} a_t(\tilde{x}_t)$, or $a_t(\tilde{x}_t)$ and $a_t(x_t)$ are not comparable.

We recall now some well-known definitions concerning the partial ordered structures.

Definition 27 *Let $(X, \leq_X)$ be a finite poset. An antichain (resp. chain) in X is a set of pairwise incomparable (resp. comparable) elements. The size of the longest antichain (resp. chain) is called the partial order width (resp. length).*

Definition 28 *Let $(X, \leq_X)$ be a finite poset. A chain C is called complete in X if for any $x, y \in C$ such that $x \leq_X y, \nexists z \in X$ such that $x <_X z <_X y$.*

Lemma 29 *(Dilworth's Lemma and its dual (Dilworth, 1950; Mirsky, 1971)) The partial order width of a finite poset $(X, \leq_X)$ is equal to the minimum number of chains needed to cover X. The partial order length of a finite poset $(X, \leq_X)$ is equal to the minimum number of antichains needed to cover X. If N be the cardinality of X, W the partial order width, and L the partial order length then $N \leq LW$.*

Definition 30 *Let K a subset of a poset $(X, \leq_X)$ and $x \in X$. We say that x is an upper bound (resp. lower bound) of K, denoted by $x \geq_X K$ (resp. $x \leq_X K$), if for all $y \in K$, we have $x \geq_X y$ (resp. $x \leq_X y$).*

Definition 31 *A lattice is a partially ordered set $(X, \leq_X)$ where for any pair of elements there is a least upper bound and greatest lower bound (belonging to X). Where a is the supremum (or least upper bound) of $A \subset X$ if a is an upper bound of A and for all $b \in X$, if b is an upper bound of A then $a \leq b$. Similar a is the infimum (or greatest lower bound) of $A \subset X$ if a is a lower bound of A and for all $b \in X$, if b is a lower bound of A then $b \leq a$.*

Let $Y \subseteq X$ a subset in the lattice. Then Y is a complete sublattice of X if for any $a, b \in Y$, both their maximum (denoted by $a \vee b$), and their minimum (denoted by $a \wedge b$) taken in X are elements of Y.

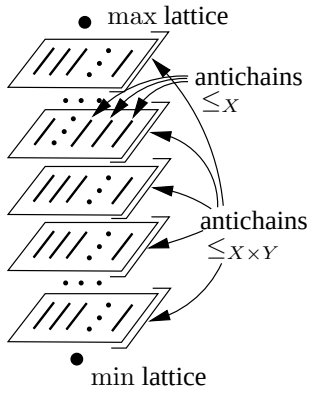

Figure 6.1. Representation of a vector-lattice $X \times Y$ as a set of antichains — layers, each of them in turn also represented as as a union of antichains according to $\leq_X$

Definition 32 *In the vector-lattice $(X_1 \times \ldots \times X_N, \leq_{X_1}, \ldots, \leq_{X_N})$ we have a natural extension of the partial orderings defined on each X_i, for any $i \in \{1, \ldots, N\}$, and we say that x, $y \in X_1 \times \ldots \times X_N$ are in the relation "$x \leq y$" if componentwise $x_i \leq_{X_i} y_i$, $\forall i \in \{1, \ldots, N\}$.*

Now, we want to introduce the notion of a subset without holes of a poset. A general definition using distances and discs can be found in (Neverman and I., 1985), but using Dilworth's decomposition Lemma (i.e Lemma (29)) one can introduce a more intuitive definition for the case of vector lattices.

As a consequence of Dilworth's Dual Lemma each poset can be decomposed into a fixed minimal number of antichains. Thus, one can decompose a vector-lattice $X \times Y$ (see Definition 32) first according to the partial order $\leq_{X\times Y}$, and afterwards all the elements of any obtained antichain with respect to $\leq_{X\times Y}$ can be decomposed further into antichains with respect to the order $\leq_X$ on X. An intuitive representation of such a decomposition is given by Figure 6.1. We notice that a complete chain in $X \times Y$ contains an element from each horizontal layer of the graphic decomposition of the lattice $X \times Y$.

Definition 33 *Let $(X \times Y, \leq_{X\times Y})$ be a vector-lattice and $Z \subseteq X \times Y$. We say that Z is a subset without holes if for any $(x, y), (\tilde{x}, \tilde{y}) \in Z$, Z includes all the minimal length complete chains containing elements $(a, b) \in X \times Y$ such that $(x, y) \leq_{X\times Y} (a, b) \leq_{X\times Y} (\tilde{x}, \tilde{y})$ (see Figure 6.2).*

Definition 34 *Let $(X \times Y, \leq_{X\times Y})$ be a vector-lattice and $Z \subseteq X \times Y$ a subset without holes. Then Bot(Z) is the bottom of Z if any $y \in$ Bot(Z) is such that $\nexists x \in Z$ with $x < y$ (i.e. a minimal element of Z) and any two elements of Bot(Z) are not comparable. Likewise, by reversing the order we define that Top(Z) is the top of $Z \subseteq X \times Y$.*

Definition 35 *A subset K of a poset $(X, \leq_X)$ is said to be increasing if $x \in K, \tilde{x} \in X$ and $x \leq_X \tilde{x}$ imply $\tilde{x} \in K$.*

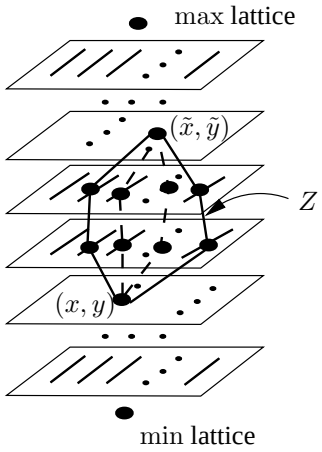

Figure 6.2. Representation of a set Z without holes of the vector-lattice $X \times Y$

Definition 36 *Let X, Y be posets. A function $g : X \times Y \to \mathbb{R}$ is supermodular/superadditive on $X \times Y$. if it satisfies: $g(x^+, y^+) + g(x^-, y^-) \geq g(x^+, y^-) + g(x^-, y^+)$ for any $x^+ \geq_X x^-$ in X, and $y^+ \geq_Y y^-$ in Y. (Section 4.7.2. (Puterman, 1994))*

Definition 37 *Let X, Y be posets. Let $S \subseteq X \times Y$, we say that $S_y = \{x | (x, y) \in S\}$ is the section of S at $y \in Y$.*

Definition 38 *Let X, Y be posets, $S \subseteq X \times Y$, and S_y the section of S at $y \in Y$. A function $f : S \to \mathbb{R}$ has increasing differences in (x, y) on S if $f(x, y^+) - f(x, y^-)$ is increasing in x on $S_{y^-} \cap S_{y^+}$, for all $y^+ \geq_Y y^-$ in Y. (Section 2.6.1. (Topkis, 1998))*

We also have the following definition which consistent with Definition 36 for X a lattice.

Definition 39 *Let X be a lattice. A function $f : X \to \mathbb{R}$ is supermodular on X if it satisfies $f(x) + f(\tilde{x}) \leq f(x \vee \tilde{x}) + f(x \wedge \tilde{x})$ for all x,$\tilde{x} \in X$, where $\vee$ denotes the maximum operator, and $\wedge$ denotes the minimum lattice operator (Section 2.6.1. (Topkis, 1998)).*

As we mentioned in the introduction there is an economic interpretation of supermodularity. Suppose that f specifies production costs in an economic system with n kinds of inputs and suppose that all inputs are used as much as possible. If f is supermodular on $\mathbb{R}^n_+$, then

$$f(x + \gamma e_i) - f(x) \leq f(x + \lambda e_j + \gamma e_i) - f(x + \lambda e_j)$$

for $x \in \mathbb{R}^n_+$, $\gamma > 0$, $\lambda > 0$, and $i \neq j$. This inequality states that the extra cost of using γ extra units of input i is raised by having λ extra units of input j (notice that here we have no economies of scales, but the opposite effect). In other words, having more of input j may reduce the effectiveness of using more of input i. In this sense, inputs i and j substitute each other's effectiveness. In economics, nondecreasing differences, hence supermodularity, is equivalent to commodities being *substitutes*. Similarly, submodularity, is equivalent to commodities being *complementary*.

Notions of "substitute" and "complementary" products arise in microeconomics in theories of production and consumer choice. Two products are regarded as complements (substitutes) if having more of one does not induce you to choose less (more) of the other. Shoes and shoelaces are complementary, whereas spaghetti and macaroni are substitutes. Observe that the word definition of "complementary" ("substitute") is essentially the stipulation that a utility function have decreasing (increasing) differences, hence that it is submodular (supermodular).

6.2 Sufficient conditions for monotonic nondecreasing policies

The goal of this subsection is to review the sufficient conditions for the existence of optimal monotonic policies from (Topkis, 1998; White, 1980), in the case of a nonstationary discrete-time Markov decision model with bounded partially ordered state space, and action space in the vector-lattice $\mathbb{R}^q_+$ endowed with the componentwise partial ordering (see Definition 32).

Definition 40 *Let $S \subseteq \mathbb{R}^q$ and $J \subseteq \mathbb{R}^m$, and let $\int_S dF_j(w)$ be the probability measure of S with respect to the distribution function $F_j(w)$ on $\mathbb{R}^q$. $F_j(w)$ is said to be stochastically increasing in j on J if $\int_S dF_j(w)$ is increasing in j on J for each increasing subset S in $\mathbb{R}^q$.*

Lemma 41 *(general form in Theorem 3.9.1.(Topkis, 1998), (Athey and Schmutzler, 1995)) Let J a subset of $\mathbb{R}^m$and $\{F_j(w) : j \in J\}$ a collection of distribution functions on $\mathbb{R}^q$. $F_j(w)$ is stochastically increasing in j on J if and only if $\int h(w)dF_j(w)$ is increasing in j on J for each increasing real-valued function $h(w)$ on $\mathbb{R}^q$.*

Corollary 42 *(generalization of Lemma 4.7.2 from (Puterman, 1994)) Let $\{z_j\}_{j\in Y}$, $\{\tilde{z}_j\}_{j\in Y}$, $\{v_i\}_{i\in Y}$ be real-valued non-negative numbers indexed after $Y \in \mathbb{N}^q$. Suppose $\sum_{j\in K} z_j \geq \sum_{j\in K} \tilde{z}_j$ for any increasing subset $K \subseteq Y$ and the sums $\sum_{j\in \mathbb{Y}} z_j$, $\sum_{j\in \mathbb{Y}} \tilde{z}_j$ are less than one. If for any $i, j \in (Y, \leq_{\mathbb{N}^q})$ such that $i \geq_{\mathbb{N}^q} j$ we have $v_i \geq v_j$ then $\sum_{j\in Y} v_j z_j \geq \sum_{j\in Y} v_j \tilde{z}_j$.*

Proof. If we take in the previous lemma the distribution function $F_j(w)$ on $\mathbb{N}^q \subset \mathbb{R}^q$ is discrete, the expected value of the real-valued function $h(w)$ on $\mathbb{N}^q$,$\int h(w)dF_t(w)$, becomes $\sum h(w)P\{X_j = w\}$. By taking $J = \{1, 2\}$ $F_j(w)$ is stochastically increasing in j on J is equivalent according to the Definition 40 with $\sum_{w\in K} h(w)P\{X_2 = w\}$ is larger than $\sum_{w\in K} h(w)P\{X_1 = w\}$ for any increasing subset $K \subseteq \mathbb{N}^q$.

By taking in the previous Lemma the function $h(j) = v_j$ as an increasing one and $P\{X_t = j\} = z_j$ we can conclude the proof. ■

Theorem 43 *(Theorem 2.8.1 (Topkis, 1998)) Let Y is a lattice, X is a partially ordered set and $Y_x \subseteq Y$ for any $x \in X$ If we have*

1. *Y_x is increasing in $x \in X$ ($Y_x \subseteq Y_{\tilde{x}}$ for $x \leq_X \tilde{x} \in X$)*
2. *$f(\cdot, x) : Y \to \mathbb{R}$ is supermodular in $y \in Y$ for each $x \in X$, and has increasing differences in $(y, x) \in Y \times X$*

then $\arg\max_{y\in Y_x} f(y,x)$ *is increasing in* x *on* $\left\{x \in X; \arg\max_{y\in Y_x} f(y,x) \text{ is nonempty}\right\}$.

Proposition 44 *(Lemma 3.9.4. from (Topkis, 1998)) Consider a discrete-time nonstationary Markov decision process with finite horizon* T*, and finite state and action spaces as defined in Section 6.1, but on* $\mathbb{R}^q$ *instead of* $\mathbb{N}^q$*. If we have for any* $t \in \{0,1,...T-1\}$

1. $A_t(x_t) \subset A_t(\tilde{x}_t)$ *for any* $t \in \{0,1,...T-1\}$ *and all states* $x_t \leq_{\mathbb{R}^q} \tilde{x}_t \in X(t) \subset \mathbb{R}^q$
2. $\rho_t(\cdot, a_t), \rho_T(\cdot) : X(t) \to \mathbb{R}$ *are nondecreasing in* x_t*, for any* $(x_t, a_t) \in X(t) \times A_t(x_t)$
3. *the distribution function for the state* w *in the period* $t+1$ *given the state* x *and the action* a*,* $F_{x,a,t}(w)$*, is stochastically increasing in* x*, for any* $(x_t, a_t) \in X(t) \times A_t(x_t)$

Then $f_t(\cdot) : X(t) \to \mathbb{R}$ *is nondecreasing for any* $t \in \{0,1,...T\}$ *where*

$$f_t(x_t) = \max_{a_t \in A_t(x_t)} \left\{ \rho_t(x_t, a_t) + \gamma \int f_{t+1}(w) dF_{x,a,t}(w) \right\}.$$

Definition 45 *Let* $S \subseteq \mathbb{R}^q$ *and* $T \subseteq \mathbb{R}^m$*, and let* $\int_S dF_t(w)$ *be the probability measure of* S *with respect to the distribution function* $F_t(w)$ *on* $\mathbb{R}^q$*.* $F_t(w)$ *is said to be stochastically supermodular in* t *on* T *if* $\int_S dF_t(w)$ *is supermodular in* t *on* T *for each increasing subset* S *in* $\mathbb{R}^q$*.*

Theorem 46 *(Theorem 3.9.2. from (Topkis, 1998)) Consider a discrete-time nonstationary Markov decision process with finite horizon* T*, and finite state and action spaces as defined in Section 6.1, but on* $\mathbb{R}^q$ *instead of* $\mathbb{N}^q$*. If we have for any* $t \in \{0,1,...T-1\}$

1. $S_t := \{(x_t, a_t) | (x_t, a_t) \in X(t) \times A_t(x_t)\}$ *is a sublattice of the vector lattice* $\mathbb{R}^{2q}$*.*
2. $A_t(x_t) \subset A_t(\tilde{x}_t)$ *for all states* $x_t \leq_{\mathbb{R}^q} \tilde{x}_t \in X(t) \subset \mathbb{R}^q$
3. $\rho_t(\cdot, a_t), \rho_T(\cdot) : X(t) \to \mathbb{R}$ *are nondecreasing in* x_t*, for any* $(x_t, a_t) \in X(t) \times A_t(x_t)$
4. $\rho_t(\cdot, a_t), \rho_T(\cdot) : X(t) \to \mathbb{R}$ *are supermodular in* $(x_t, a_t) \in S_t$
5. *the distribution function for the state* w *in the period* $t+1$ *given the state* x *and the action* a*,* $F_{x,a,t}(w)$*, is stochastically increasing in* x*, for any* $(x_t, a_t) \in X(t) \times A_t(x_t)$*, and is stochastically supermodular in* $(x_t, a_t) \in S_t$

Then for any $t \in \{0,1,...T-1\}$

1. *the function* $\rho_t(x_t, a_t) + \gamma \int f_{t+1}(w) dF_{x,a,t}(w)$ *is supermodular in* $(x_t, a_t) \in S_t$
2. $f_t(x_t) = \max_{a_t \in A_t(x_t)} \left\{ \rho_t(x_t, a_t) + \gamma \int f_{t+1}(w) dF_{x,a,t}(w) \right\}$ *is supermodular in* $x_t \in X(t)$
3. *the set of optimal decisions* $\arg\max_{a_t \in A_t(x_t)} \left\{ \rho_t(x_t, a_t) + \gamma \int f_{t+1}(w) dF_{x,a,t}(w) \right\}$ *is nondecreasing in the state* $x_t \in X(t)$*, and*
4. *there is a greatest (least) optimal decision for each state* x_t *and this greatest (least) optimal decision is increasing in* $x_t \in X(t)$ *(i.e. there exists at least one monotonic nondecreasing optimal policy).*

In Chapter 7 Section 7.2 we will use a discrete version (i.e. on $\mathbb{N}^q$ instead of $\mathbb{R}^q$) of Theorem 46 to prove the existence of monotonic nondecreasing optimal policies for a slightly modified version of the NPD problem without precedence constraints.

A more general form of Theorem 46 can be found in (White, 1980). The theorem in (White, 1980) states when there exist optimal decision rules which are structured in the state of the system for discrete Markov decision processes with $S_t := \{(x_t, a_t)|(x_t, a_t) \in X(t) \times A_t(x_t)\}$ sublattice of an arbitrary lattice, and with summarized utility functions. The usual additive utility function used in Chapter 5 and in Chapter 7 is a special case of the summarized utility functions. Also, some of general conditions from (White, 1980) are used to prove the existence of an optimal policy for an infinite horizon problem

In the case of bounded subsets without holes of the infinite vector-lattice $\mathbb{N}^q$ we can use as in (Puterman, 1994) the terminology superadditive instead of supermodular, nondecreasing instead of isotone, while considering maximization problems. The existence of weakly monotonic optimal policies can be obtained under less restrictive requirements than the one from (White, 1980), (Topkis, 1998): for any $t \in \{0, 1, ...T-1\}$, $X(t) \times A_t(x_t)$ is a sublattice of the vector lattice $\mathbb{R}^{2q}$.

6.3 Sufficient conditions for weakly-monotonic nondecreasing policies

The goal of this subsection is to give new sufficient conditions for the existence of weakly optimal monotonic policies, in the case of a nonstationary discrete-time Markov decision model with bounded partially ordered state space, and action space in the vector-lattice $\mathbb{N}^q$ endowed with the componentwise partial ordering (see Definition 32).

Lemma 47 *(extension of Theorem 43 Theorem 2.8.1 (Topkis, 1998),and Lemma 4.7.1 from (Puterman, 1994)) Let X be a poset and let $Y := \cup_{x \in X} Y_x$ be a poset indexed after X, and $g : X \times Y \to \mathbb{R}$ a real valued superadditive function on $(\tilde{x}, \tilde{y}) \in X \times Y_x$, for each $x \in X$, with $g(x, y) = 0$, for any $(x, y) \notin X \times Y_x$. If we have*

1. *for each $x \in X$ there exists $\max_{y \in Y} g(x, y)$*
2. *$Y_x \subset Y_{\tilde{x}}$ for any $x \leq_X \tilde{x} \in X$, and $Y_x = Y_{\tilde{x}}$ for any $x, \tilde{x} \in X$ not comparable (i.e. the family $\{Y_x | x \in X\}$ is expanding)*
3. *for any $x \leq_X \tilde{x} \in X$, and $y \leq \tilde{y} \in Y_x \cup Y_{\tilde{x}}$ we have that $y \in Y_x$ and $\tilde{y} \in Y_{\tilde{x}}$ (i.e. the family is ascending)*

then for any $x^+ \geq_X x^-$ in X, and any $y^- \in \operatorname{Top} \arg\max_{y \in Y_{x^-}} g(x^-, y)$ either there exists $y^+ \in \operatorname{Top} \arg\max_{y \in Y_{x^+}} g(x^+, y)$ such that $y^+ \geq_Y y^-$ in Y, or there is no element in $\operatorname{Top} \arg\max_{y \in Y_{x^+}} g(x^+, y)$ comparable with y^-.

Proof. Let $x^+ \geq_X x^-$ in X, and choose $y \leq_Y y_{x^-} \in \operatorname{Top} \arg\max_{y \in Y_{x^-}} g(x^-, y)$ and $y \in Y$. Then there exists an $x \in X$ such that $y \in Y_x \subset Y$.

By definition of $\arg\max_{y \in Y_{x^-}} g(x^-, y)$ we have $g(x^-, y_{x^-}) - g(x^-, y) \geq 0$.

Because $y \leq_Y y_{x^-}$ we have that $y \in Y_{x^-}$ by using hypothesis 3 with $Y_x \cup Y_{x^-}$ if $x \geq_X x^-$, or by using hypothesis 2 otherwise. Since $Y_{x^-} \subseteq Y_{x^+}$ (by hypothesis 2) we have

that both $y, y_{x^-} \in Y_{x^+}$. Now, since g is a superadditive function on $(\tilde{x}, \tilde{y}) \in X \times Y_{x^+}$ and all the pairs are in the definition domain, we also have: $g(x^+, y_{x^-})$ + $g(x^-, y) \geq g(x^+, y)$ +$g(x^-, y_{x^-})$.

Rewriting the second inequality as $g(x^+, y_{x^-}) \geq g(x^+, y)+[g(x^-, y_{x^-})-g(x^-, y)]$ and using afterwards the first inequality we obtain that: $g(x^+, y_{x^-}) \geq g(x^+, y)$ for all $y \leq_Y y_{x^-} \in \operatorname{Top}\arg\max_{y \in Y_{x^-}} g(x^-, y)$, which concludes the proof by maximizing after y_{x^-} and y simultaneously. ■

Proposition 48 *Consider a discrete-time nonstationary Markov decision process with finite horizon T, and finite state and action spaces as defined in Section 6.1 and endowed with the componentwise partial order of $\mathbb{N}^q$. If we have for any $t \in \{0, 1, ...T-1\}$*

1. *$X(t), A_t(x_t)$, for any $x_t \in X(t)$ are bounded subsets without holes of the infinite vector-lattice $\mathbb{N}^q$*
2. *$A_t(x_t) \subset A_t(\tilde{x}_t)$ for any $x_t \leq_{\mathbb{N}^q} \tilde{x}_t \in X(t)$, and $A_t(x_t) = A_t(\tilde{x}_t)$ for any $x_t, \tilde{x}_t \in X(t)$ not comparable (i.e. the family $\{A_t(x_t) | x_t \in X(t)\}$ is expanding)*
3. *for any $x_t \leq_{\mathbb{N}^q} \tilde{x}_t \in X(t)$, and $a \leq_{\mathbb{N}^q} \tilde{a} \in (A_t(x_t) \times A_t(\tilde{x}_t)) \cup (A_t(\tilde{x}_t) \times A_t(x_t))$ we have that $a \in A_t(x_t)$ and $\tilde{a} \in A_t(\tilde{x}_t)$ (i.e. the family is ascending)*
4. *$\rho_t(\cdot, \cdot)$ is superadditive in $(x_t, a_{x_t}) \in X(t) \times A_t(x_t)$*
5. *$\rho_t(\cdot, a_t), \rho_T(\cdot) : X(t) \to \mathbb{R}$ are nondecreasing*
   and if for any increasing subset $K \subseteq X(t+1)$
6. *$F(\cdot, \cdot) := \sum_{x_{t+1} \in K} p_t(\cdot, \cdot, x_{t+1})$ is superadditive in $(x_t, a_t) \in X(t) \times A_t(x_t)$*
7. *$F(\cdot, a_t) := \sum_{x_{t+1} \in K} p_t(\cdot, a_t, x_{t+1})$ is nondecreasing in $x_t \in X(t)$, for any $a_t \in A_t(x_t)$*
8. *if $(\exists) x_t \neq \tilde{x}_t \in X(t)$ such that $A_t(x_t) \neq A_t(\tilde{x}_t)$ then $\sum_{x_{t+1} \in K} p_t(x, a_t, x_{t+1})$ is nondecreasing in $a_t \in A_t(x)$, for any $x \in X(t)$*

Then there exists optimal decision policies which are weakly monotonic nondecreasing in the state, for any $t \in \{0, 1, ...T-1\}$ (i.e. for any $x_t \leq_{\mathbb{N}^q} \tilde{x}_t \in X(t)$, and any $a_t \in \operatorname{Top}\arg\max_{a \in A_t(x_t)} w_t(x_t, a)$ either there exists $\tilde{a}_t \in \operatorname{Top}\arg\max_{\tilde{a} \in A_t(\tilde{x}_t)} w_t(\tilde{x}_t, \tilde{a})$ such that $\tilde{a}_t \geq_{\mathbb{N}^q} a_t$, or there is no element in $\operatorname{Top}\arg\max_{\tilde{a} \in A_t(\tilde{x}_t)} w_t(\tilde{x}_t, \tilde{a})$ comparable with a_t).

Proof. Let us define
$u_t^*(x_t) = \max_{a_t \in A_t(x_t)} \left\{ \rho_t(x_t, a_t) + \sum_{x_{t+1} \in X(t+1)} p_t(x_t, a_t, x_{t+1}) \cdot u_{t+1}^*(x_{t+1}) \right\}$, and
$u_T^*(x_T) = \rho_T(x_T)$.

If we prove $w_t(x_t, a_t) := \rho_t(x_t, a_t) + \sum_{x_{t+1} \in X(t+1)} p_t(x_t, a_t, x_{t+1}) \cdot u_{t+1}^*(x_{t+1})$ is superadditive in $(x_t, a_t) \in X(t) \times A_t(x_t)$ we can apply Lemma 47 since $\max_{a_t \in A(x_t)} w_t(x_t, a_t)$ is attained for finite action space and state space. Thus, for any $\tilde{x}_t \geq_{\mathbb{N}^q} x_t \in X(t)$, and

any $a^*_{x_t} \in \text{Top}\arg\max_{a_{x_t}\in A(x_t)} w_t(x_t, a_{x_t})$, $A_t(x_t) \subset A_t(\tilde{x}_t)$ and there exists $a^*_{\tilde{x}_t} \in \text{Top}\arg\max_{a_{\tilde{x}_t}\in A_t(\tilde{x}_t)} w_t(\tilde{x}_t, a_{\tilde{x}_t})$ such that $a^*_{\tilde{x}_t} \geq_{\mathbb{N}^q} a^*_{x_t}$ or $a^*_{x_t}$ is not comparable with any element in $\text{Top}\arg\max_{a_{\tilde{x}_t}\in A_t(\tilde{x}_t)} g(\tilde{x}_t, a_{\tilde{x}_t})$. This property ensures the existence of weakly monotonic nondecreasing optimal policies.

Let $\tilde{x}_t \in X(t)$ such that $\tilde{x}_t \geq_{\mathbb{N}^q} x_t$.

We consider in Proposition 44 a distribution function on $\mathbb{N}^q \subset \mathbb{R}^q$ which is discrete. Then the expected value of the real-valued function $f_{t+1}(w)$ on $\mathbb{R}^q$, $\int f_{t+1}(w)dF_{x,a,t}(w)$, becomes $\sum f_{t+1}(w)P\{X_{x,a,t} = w\}$.

Thus, from Proposition 44, $u^*_{t+1}(\cdot) : X(t+1) \to \mathbb{R}$ is nondecreasing (according to the partial order on $\mathbb{N}^q$) for any $t \in \{0, 1, ...T-1\}$ because we have $\rho_t(\cdot, a_t)$, $\rho_T(\cdot)$, $F(\cdot, a_t) := \sum_{x_{t+1}\in K} p_t(\cdot, a_t, x_{t+1}) : X(t) \to \mathbb{R}$ are nondecreasing in x_t, for any $(x_t, a_t) \in X(t) \times A_t(x_t)$, and for any increasing subset $K \subseteq X(t+1)$.

Let $a_t \leq_{\mathbb{N}^q} \tilde{a}_t \in (A_t(x_t) \times A_t(\tilde{x}_t)) \cup (A_t(\tilde{x}_t) \times A_t(x_t))$ arbitrary. By hypothesis, for any $x_t \leq_{\mathbb{N}^q} \tilde{x}_t \in X(t) \subset \mathbb{N}^q$, and $a_t \leq_{\mathbb{N}^q} \tilde{a}_t$ as above, we have that $a_t \in A_t(x_t)$ and $\tilde{a}_t \in A_t(\tilde{x}_t)$. Since $A_t(x_t) \subset A_t(\tilde{x}_t)$ for any $x_t \leq_{\mathbb{N}^q} \tilde{x}_t \in X(t) \subset \mathbb{N}^q$ we also have that $a_t \in A_t(\tilde{x}_t)$. If $\tilde{a}_t \in A_t(x_t)$ then for any increasing subset $K \subseteq X(t+1)$ $\sum_{x_{t+1}\in K} [p_t(\tilde{x}_t, \tilde{a}_t, x_{t+1}) + p_t(x_t, a_t, x_{t+1})] \geq \sum_{x_{t+1}\in K} [p_t(x_t, \tilde{a}_t, x_{t+1}) + p_t(\tilde{x}_t, a_t, x_{t+1})]$ holds directly because $\sum_{x_{t+1}\in K} p_t(x_t, a_t, x_{t+1})$ is superadditive in $(x_t, a_t) \in X(t) \times A_t(x_t)$, for any increasing subset $K \subseteq X(t+1)$.

If $\tilde{a}_t \notin A_t(x_t)$ then $\sum_{x_{t+1}\in K} p_t(x_t, \tilde{a}_t, x_{t+1}) = 0$, and $A_t(x_t) \neq A_t(\tilde{x}_t)$. But in this case we have that $a_t \leq_{\mathbb{N}^q} \tilde{a}_t$, $a_t, \tilde{a}_t \in A_t(\tilde{x}_t)$, and $\sum_{x_{t+1}\in K} p_t(\tilde{x}_t, a_t, x_{t+1})$ is nondecreasing in $a_t \in A_t(\tilde{x}_t)$ for any increasing subset $K \subseteq X(t+1)$. This implies that $\sum_{x_{t+1}\in K} [p_t(\tilde{x}_t, \tilde{a}_t, x_{t+1}) + p_t(x_t, a_t, x_{t+1})] \geq \sum_{x_{t+1}\in K} p_t(\tilde{x}_t, \tilde{a}_t, x_{t+1}) \geq \sum_{x_{t+1}\in K} p_t(\tilde{x}_t, a_t, x_{t+1})$. For any $j \in X(t+1)$ we denote by $z_j := [p_t(\tilde{x}_t, \tilde{a}_t, j) + p_t(x_t, a_t, j)]$, $\tilde{z}_j := [p_t(x_t, \tilde{a}_t, j) + p_t(\tilde{x}_t, a_t, j)]$, and $v_j := u^*_{t+1}(j)$, in order to apply for them Corollary 42 . Its hypotheses are fulfilled since $X(t+1) \subseteq R^q$, $\sum_{j\in K} z_j \geq \sum_{j\in K} \tilde{z}_j$, for any increasing subset $K \subseteq X(t+1)$ and the sums are finite. Thus, we have $\sum_{j\in X(t+1)} [p_t(\tilde{x}_t, \tilde{a}_t, j) + p_t(x_t, a_t, j)]u^*_{t+1}(j) \geq \sum_{j\in X(t+1)} [p_t(x_t, \tilde{a}_t, j) + p_t(\tilde{x}_t, a_t, j)]u^*_{t+1}(j)$, which implies that the function $\sum_{x_{t+1}\in X(t+1)} p_t(x_t, a_t, x_{t+1})\, u^*_{t+1}(x_{t+1})$ is superadditive in $(x_t, a_{x_t}) \in X(t) \times A_t(x_t)$, for any $t \in \{0, 1, ...T-1\}$.

We assumed that $\rho_t(\cdot, \cdot)$ is superadditive too in $(x_t, a_{x_t}) \in X(t) \times A_t(x_t)$, for any $t \in \{0, 1, ...T-1\}$. Since the sum of superadditive functions defined on the same domain remains superadditive $w_t(x_t, a_t)$ is superadditive in $(x_t, a_t) \in X(t) \times A_t(x_t)$. ■

6.4 Weakly monotonic nondecreasing backward induction

An optimal policy has to give optimal actions for each state. This can be done through a recursive computation, starting from the latest moments in time and working towards the beginning of time, via a general backward induction algorithm. Even when mono-

tonic policies can be found, as in (Puterman, 1994), a monotonic backwards induction algorithm can be used only for very particular nonstationary Markov decision processes (that is, only the states and the actions are not time-dependent) with $X = \{0, 1, \ldots, M\}$ with M finite and with $A(x) = A$ for all $x \in X$. Such a *monotonic backward induction algorithm is preferred to a normal backward induction algorithm because leads to computational advantages from considering only nondecreasing control actions. This restricts the backward induction algorithm to look for candidate actions for the* $\arg\max$ *for a given state* x *only in the superior cones of the elements of the* $\arg\max$ *of all states* $\widetilde{x}$ *which are smaller than* x.

However, here, from the results presented in Subsection 6.3, we can construct a new algorithm from this family, which works on very general partially-ordered state and respectively action spaces, speeding up the whole processing by a significant factor. We call this new algorithm a weakly monotonic nondecreasing backward induction algorithm. One of its main characteristics is the ordered sweeping of the state and action spaces when looking for the $\max$ and the $\arg\max$, namely allowed by Lemma 47 and Proposition 48. These results basically allow the algorithm to look for candidate actions for the $\arg\max$ for a given state x in the action space from which we extract all the inferior cones of the elements of the $\arg\max$ of all states x' which are smaller than x, instead of looking for it in the whole available action space.

The algorithm, with an illustrating example and comments

The goal of this part is to provide an easy-to-implement version of our algorithm, as well as to illustrate through an example the characteristics and the differences between our algorithm and a monotonic backwards induction algorithm. Our algorithm benefits from the fact that both the state space and action space are subsets of a lattice without holes. Even small lattices (e.g. subsets of $\mathbb{N}^3$) cannot be entirely handled in the main computer memory (with all the couples state-action subsets needed by any classical backward induction algorithm). Thus, we use the fact that both $X(t)$ and $A(t)$ can be covered by disjoint antichains (Lemma 29, by Dilworth and Mirsky), exploring them in this "ordered" manner. For each segment of lattice explored during the execution of a backward induction algorithm , we only keep track of the bottom and top, looking for the optimal action through an antichain sweep between them. This is done in the **while** loop (lines $8 - 18$), which thus maintains a current antichain. The first such antichain is the very bottom of $X(t)$ (line 3): thus for each t, this bottom needs to be computed in advance, from the definition of $X(t)$. As the **while** loops, the other antichains are recursively computed (line 13). Then, inside this loop, for each state of the current antichain, by looking in its reduced action space $\widehat{A}$ (set in line 10), its optimal actions and the reward are computed (line 11), in the first **for** loop (lines 9-12). Thereafter, based on that, the boundaries of the reduced action space of the new antichain (which will become the current one) are iteratively computed, in the second **for** loop (lines $14 - 16$).

```
1   for each x ∈ X(T) do u*(x) ← ρ_T(x) endfor
2   for t from T − 1 to 0 do
3     Current_State_Antichain_0 ←Bottom_{X(t)}
4     for each x ∈ Current_State_Antichain_0 do
5       initialize Current_Action_Bottom[x]
6     endfor
7     i ← 0
```

while Current_State_Antichain$_i \neq \emptyset$ **do**
  for each $x \in$Current_State_Antichain$_i$ **do**
    $\widehat{A} \leftarrow \{a \in A(t) \,|\forall c \in \text{Current\_Action\_Bottom}[x], a \not\prec c\}$
    $$(u^*(x), A^*(x)) \leftarrow (\max, \underset{a\in\widehat{A}}{\text{Top}\arg\max}) \left\{ \rho_t(x,a) + \sum_{y\in X(t+1)} p_t(x,a,y)\cdot u^*(y) \right\}$$
  endfor
  $$\text{Current\_State\_Antichain}_{i+1} \leftarrow \text{Bot} \bigcup_{x\in\text{Current\_State\_Antichain}_i} \left\{ y \in \text{Succ}(x) | y \leq \text{Top}_{X(t)} \right\}$$
  for each $x \in$Current_State_Antichain$_{i+1}$**do**
    $$\text{Current\_Action\_Bottom}[x] \leftarrow \text{Top} \bigcup_{y\in\text{Current\_State\_Antichain}_i, \text{ with } y\in\text{Pred}(x)} A^*(y)$$
  endfor
  $i \leftarrow i+1$
  endwhile
endfor
end

By Succ(x) we mean the set of direct successors of the element x in the poset $X(t)$ (i.e. in $\mathbb{N}_q$ Succ$(x) := \{y|\exists\ 1 \leq i \leq q$ s.t. $y = x + e_i$,where $e_i(i) = 1$, and $e_i(j) = 0, \forall j \neq i\}$

Let us follow the algorithm through a simple NPD project example. We consider a NPD problem without precedence constraints as described in Chapter 5 Section 5.4 Subsection 5.4.1. A more in depth discussion of this type of NPD follows in Chapter 7.

Let $M = 2$ engineers solving a number of $N = 3$ concurrent design tasks available at time $t = 0$, with $\mu = 100$ and $\lambda = 50$ for both $t = 0$ and $t = 1$. The deadline is $T = 2$. In this example, we do not consider any design tasks arrival process.

If the product is fully functional at the deadline $T = 2$ (i.e. the levels achieved will be greater than or equal to the minimum levels $l_{\min}(n)$ fixed in advance for each task $n \in \{1, 2, 3\}$) we choose to reward the designed product with a multiplicative market payoff function as in (Yoshimura, 1996). This type of reward function is to the best of our knowledge the most general $S-$type market payoff function encountered in literature. A discussion on the market payoff functions appears in 2 Subsection 2.3.3, and the market payoff related parameters are fully described in Chapter 3 Section 3.3. The reason for the "$S-$type" name comes from the fact that the graph of the function is somehow "S"-shaped.

We consider a number of $\Delta = 2$ customer needs, with the first task contributing only to the first customer need, that is $\theta_{1,1} = 1$ and $\theta_{2,1} = 0$, and with the second and third task contributing only to the second customer need, with $\theta_{2,2} = 0.6$ and $\theta_{2,3} = 0.4$, and $\theta_{1,2} = \theta_{1,3} = 0$. These values stay the same after normalization. We then use the multiplicative formula (from (Yoshimura, 1996)) given here on page 30: $\prod_{\delta=1}^{\Delta} \left[\overline{S}_\delta(t, l(\cdot,t))\right]^{w_\delta(t)}$. We recall from page 30 that $\overline{S}_\delta$ should be an $S-$type function giving the distance between the cumulated design tasks contribution per customer need δ, $\sum_{n=1,..,N} \Theta(n,t,\delta) \cdot \frac{l(n,t)}{L_{\max}(n,t)}$, and the customer need ideal value. Since in this example there is no new design tasks arrival we have that $\overline{N} = N$ and according to the notation from page 28 for all the planned design tasks $L_{\max}(n,t) = L_{\max}(n)$. For the $S-$type function, we choose the cdf of the normal distribu-

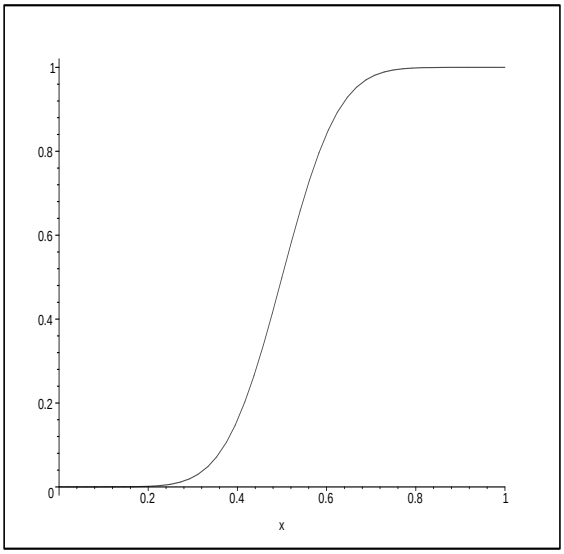

Figure 6.3. The shape of the second customer need payoff function

tion $\Pr_{\mathcal{N}(0,1)}(X < x)$, appropriately scaled and centered on a positive x. Thus, the formula to compute the reward for a final state $(l(1,2), l(2,2), l(3,2))$ (i.e. at $t = 2$) is

$$\prod_{i=1}^{2} \left(a_i \cdot \Pr_{\mathcal{N}(0,1)} \left(10X - 5 < \sum_{n=1}^{3} \theta_{i,n} \cdot \frac{l(n,2)}{L_{\max}(n)} \right) + b_i \right)^{w_i}$$

with $\mathcal{N}(0,1)$ being the normal distribution function with mean 0 and standard deviation 1. In the program implementation of the algorithm, we used the library error function erf defined as

$$erf(x) = \frac{2}{\sqrt{\pi}} \cdot \int_0^x \mathrm{e}^{-t^2} dt \quad \text{where} \quad \Pr_{\mathcal{N}(0,1)}(X < x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \mathrm{e}^{\frac{-t^2}{2}} dt = \frac{1 + \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)}{2}.$$

The exponents have the values $w_1 = 0.3$ and $w_2 = 0.7$. the coefficients for the final linear transformation before raising exponentiation are $a_1 = 100$ and $a_2 = 300$, and the free terms are $b_1 = 0$ and $b_2 = 50$.

Thus the shape of the second customer need payoff is as in Figure 6.3, and the shape of the total market payoff becomes as in Figure 6.4.

We consider for each of the tasks, triples with the number of levels, the number of activities per level (unit of work), and the minimum level required:

The task #1 has 13 levels, 80 activities per level, and minimum level equal to 1.

The task #2 has 17 levels, 90 activities per level, and minimum level equal to 2.

The task #3 has 14 levels, 73 activities per level, and minimum level equal to 0.

The manager decides the levels up to which the design tasks should be performed. He wants to be sure with the safety level $\beta = 0.85$ that at the deadline the decided levels will be achieved.

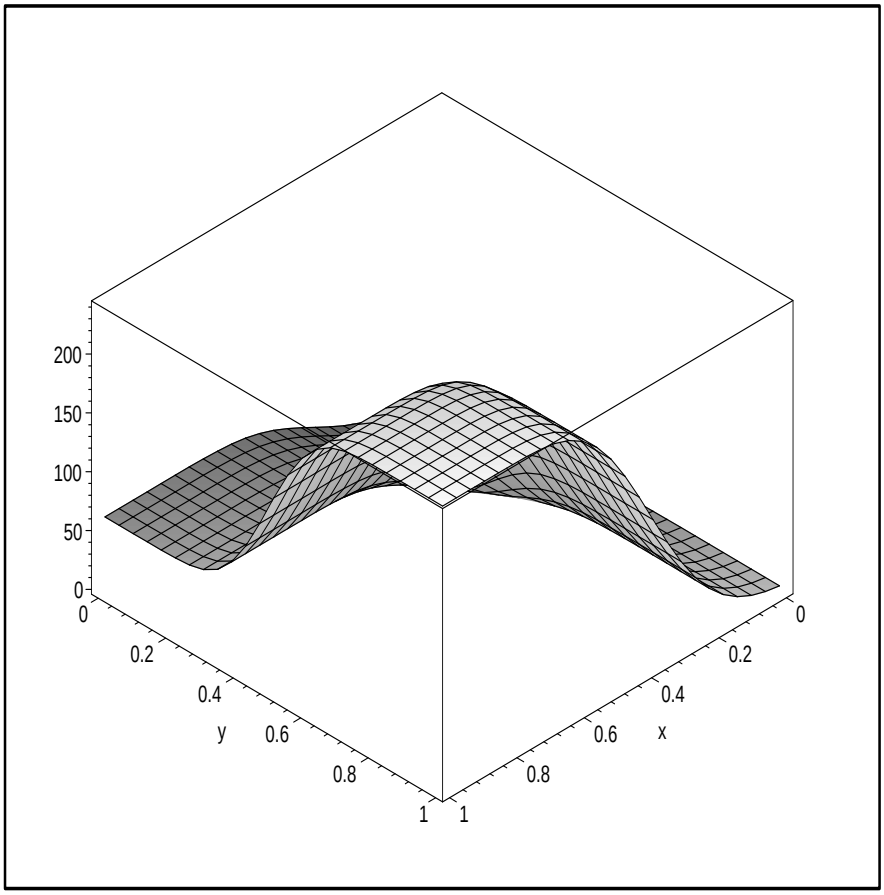

Figure 6.4. The shape of the total market payoff function

By computing the workload probability constraint for this NPD without precedence constraints (see Chapter 5 Section 5.4 Subsection 5.4.1) we have the following state space descriptions for the review periods:
$X(0) = \{(0,0,0)\}$
$X(1) = \{(0,1,0),(0,1,1),(0,2,0),(1,0,0),(1,0,1),(1,1,0),(2,0,0)\}$
$X(2) = \{(1,2,0),(1,2,1),(1,3,0),(2,2,0)\}$

The action space description will be independent of the state of the system. This happens because an action a_t in the state x_t decides how many other levels above x_t we want to perform, and not the performance level up to which the design task n may be solved until the end of the review period. A formal proof of this fact appears in Chapter 7 Proposition 57. Thus, the action spaces are:
$A(0) = \{a|\,(0,0,0) \leq a \leq b, \text{ with } b \in B\}$, where
$B = \{(0,0,5),(0,1,3),(0,2,2),(0,3,1),(0,4,0),(1,0,4),(1,1,2),(1,2,1),$
$(1,3,0),(2,0,2),(2,0,3),(2,1,1),(2,2,0),(3,0,1),(3,1,0),(4,0,0)\}$
$A(1) = \{a|\,(0,0,0) \leq a \leq \{(0,0,2),(0,1,1),(0,2,0),(1,0,1),(1,1,0),(2,0,0)\}\}$

Let us now follow step by step the algorithm.

The rewards, for the four states of $X(2)$, are respectively 2.03767, 2.03785, 2.03794 and 5.22398, assigned in line 1 of the algorithm.

The loop of lines from 2 to 19 starts with $t = 1$. We have that $\text{Bottom}_{X(1)}$ is $\{(0,1,0),(1,0,0)\}$, from the β confidence threshold constraint.

Thus, we set $\text{Current\_State\_Antichain}_0$.

Let us look farther, at line 10, starting now the inner loops. The top of actions is

$\text{Top}_{A_1(1)} = \{(0,0,2),(0,1,1),(0,2,0),(1,0,1),(1,1,0),(2,0,0)\}$ (also from the β confidence threshold constraint). Thus, all actions between $(0,0,0) = 0_{\mathbb{R}^3}$ and this top are gathered in the set $\widehat{A}$ in line 10, because initially Current_Action_Bottom has been set to zero for all the states in line 5. This gives us for $\widehat{A}$ the set

$$\{0_{\mathbb{R}^3},(0,0,1),(0,1,0),(1,0,0),(0,0,2),(0,1,1),(0,2,0),(1,0,1),(1,1,0),(2,0,0)\}.$$

Since the minimum levels for the three tasks are $(1,2,0)$, the only actions leading to rewarded states (that is with levels above the minimum) among these ten are the action $(1,1,0)$ for the state $(0,1,0)$ and the action $(0,2,0)$ for the state $(1,0,0)$, thus forming their $\arg\max$ in line 11, with $u^*((0,1,0)) = 0.403761$ and $u^*((1,0,0)) = 0.265104$.

Now the for each loop of lines 9 to 12 has ended and a new antichain has to be constructed, from the immediate successors of these two states which have just been processed.

Thus the line 13 sets

$$\text{Current\_State\_Antichain}_1 = \{(0,1,1),(0,2,0),(1,1,0),(1,0,1),(2,0,0)\},$$

and in the loop from the lines 14 to 16, the first two states have the bottom of actions set to $(1,1,0)$ being successors only of $(0,1,0)$, the last two states to $(0,2,0)$ and the state $(1,1,0)$ to both.

The line-8-to-line-18 while loops with $i = 1$ from line 17, but this time, the set $\widehat{A}$ is smaller, showing now the advantage of this type of algorithm.

For instance, for the state $(0,2,0)$, the actions $(0,1,0)$, $(1,0,0)$ and $(0,0,0)$ are eliminated thanks to the weak monotonicity, because they are smaller than the action $(1,1,0)$ which has been set to the bottom of actions of $(0,2,0)$ in lines 14 to 16, as we have said above. Thus $\widehat{A}$ only contains seven actions. Among them, the actions $(0,0,2)$, $(0,1,1)$, $(0,2,0)$ and $(0,0,1)$ do not lead to states having at least the minimum levels (because of the first component), thus not to rewarded states. The action $(1,0,1)$ leads to the rewarded states $(1,2,0)$ and $(1,2,1)$, with a total value (after computing the transition probability) of 3.02609. The action $(1,1,0)$ leads to the rewarded states $(1,2,0)$ and $(1,3,0)$ with a total value of 2.44148. Finally, the action $(2,0,0)$ leads to the rewarded states $(1,2,0)$ and $(2,2,0)$ with a total value of 3.82039. This is the greatest one, so $u^*((0,2,0))$ is set to it, and $A^*((0,2,0)) = \{(2,0,0)\}$.

This computation in lines 9 to 12 continues for the other four elements of the current antichain, which is also the last one. After considering a total of 55 actions, it ends with the

Table 6.1. Comparative results: our algorithm versus the natural extension of the unidimensional one from (Puterman, 1994) in the sense of (Topkis, 1998) and (White, 1980)

parameters		states searched by		actions searched by	
T	N	our algo	nat. ext.	our algo.	nat. ext
2	3	12	6901	94	3956
2	5	21	> 5000	2881	> 27800
2	10	26	> 6000	10748	> 69000
2	15	28	> 12000	19457	> 132000
5	3	66	21902	498	18580
5	5	146	> 38700	2996	> 25700
5	10	168	> 118700	44993	> 756000
5	15	240	> 155000	204734	> 2640000

following values for $x \in X(1)$:

x	$u^*(x)$	$A^*(x)$
$(0,1,0)$	0.403761	$(1,1,0)$
$(0,1,1)$	0.403798	$(1,1,0)$
$(0,2,0)$	3.82039	$(2,0,0)$
$(1,0,0)$	0.265104	$(0,2,0)$
$(1,0,1)$	0.265128	$(0,2,0)$
$(1,1,0)$	3.07276	$(1,1,0)$
$(2,0,0)$	0.679649	$(0,2,0)$

Now the for from lines 2 to 19 loops once, the last time, with $t = 0$; all this computation restarts, with only one state, namely $(0,0,0)$, and 39 possible actions, and it ends with $u^*((0,0,0)) = 2.02032$ and $A^*(0) = \{(1,2,1)\}$.

Now, if we want to solve this problem with a monotonic multi-dimensional algorithm in the sense of (Topkis, 1998) and (White, 1980), by naturally extending the unidimensional algorithm presented in (Puterman, 1994), we have to complete the action space up to a lattice. That is, for the values presented before, the action space contains now all triples up to $(4,4,5)$ for $t = 0$ and up to $(2,2,2)$ for $t = 1$, which obviously adds many more actions to test for. Moreover, this entails an increase of the state space, since the engineers were supposed to work up to the required levels, and the only transitions with zero probability from a state x_t upon action a being those for states outside the interval $[x_t, x_t + a_t]$. Here thus there are actions greater than in the setting of our algorithm, leading to a serious extension of the state space. We have run simulations for several values of the above parameters, and compared the increase of number of considered states, and respectively actions when computing the maximum, in both cases, of our algorithm and of the natural extension of (Puterman, 1994) in the sense of (Topkis, 1998) and (White, 1980). We give in table 6.1 these results for a comparison. Our algorithm has finished in all cases, while the natural extension has been able to finish only for $N = 3$ (and $T = 2$, $T = 5$) ; for the other cases, we decided to extend the initially decided period of simulation from several minutes to several days, and thus only reported lower bounds.

Moreover, the weakly monotonic actions are more natural in a lot of practical situations like this one, and the weakly monotonic backward induction algorithm preserves the robustness of the optimal policies (see Proposition 52 and Theorem 51).

6.5 Robustness of the optimal (weakly) monotonic nondecreasing policies

Dynamic programming problems provide optimal policies, which are very often too complicated to be used for deriving insights about the given problem (see a survey in (Dixit and Pindyck, 1996)). By studying only particular classes of structured optimal policies (as the monotonic nondecreasing, and respectively weakly monotonic nondecreasing), we are able to describe how the optimal policies change in response to some of the changes of the exogenous parameters.

Intuitively one might expect that the monotonicity of the decision rules would imply a stable structure of the optimal paths (i.e. an increase in a exogenous parameter θ will lead to a uniform increase of the optimal path). While this statement is true in the unidimensional stationary case (see (Denekere and Pelikan, 1986) for a proof), this is not necessarily true in higher dimensions, since the optimal paths arising from monotonic decision rules can be extremely complicated.

In (Friedman and Johnson, 1997) a two-dimensional example of a chaotic optimal path for a monotonic decision rule is given, as well as an analysis of connections between monotonic policies and the optimal path when $A_t(x_t)$ are sublattices of $\mathbb{R}^m$ for any $x_t \in X(t)$. The authors consider a general dynamic decision problem (DDP), where in each period the set of allowable actions may depend on the action from the previous period, as well as of the current state, and a set of exogenous parameters $\Theta := \Theta_1 \times \ldots \times \Theta_l$, where $\Theta_i \subseteq \mathbb{R}$ is locally compact. For the DDP considered is described by: an action space $A = A_1 \times \ldots A_n$, $A_i \subseteq \mathbb{R}$, and A_i locally compact, such that A is a vector-lattice, a finite set of random stationary Markovian processes $x_i^t \in X_i$, having a positive persistence of uncertainty, where $X = X_1 \times \ldots X_m$, $X_i \subseteq \mathbb{R}$, and X_i locally compact, and payoff per period $F(a^t, a^{t-1}, x^t; \theta) + \sum_i \kappa_i(a_i^t)$, where both $F(a^t, a^{t-1}, x^t; \theta)$, and $\sum_i \kappa_i(a_i^t)$ are upper semi-continuous. The assumptions considered were:

1. the value function exists and is well defined under total expected reward criterion
2. the graph $\Gamma := \{(a, \grave{a}, x; \theta) | a \in A(\grave{a}, x; \theta)\}$ is a sublattice of $A \times A \times X \times \Theta$ (in the product order), where $A(\grave{a}, x; \theta)$ is the set of allowable actions in state x.
3. $A(\grave{a}, x; \theta)$ is a complete sublattice of A.
4. $F(a, \grave{a}, x; \theta)$ is supermodular in $(a, \grave{a}) \in \Gamma(x; \theta) := \{(a, \grave{a}) | a \in A(\grave{a}, x; \theta)\}$ and has nondecreasing differences in $((a, \grave{a}), (x; \theta))$ on $A(\grave{a}, x; \theta)$ (i.e. full complementarities)

Proposition 49 *(see Corollary 1 (Friedman and Johnson, 1997)) Assume a Dynamic Decision Problem with a monotone decision rule (e.g. full complementarities). Then for any initial conditions s^0,x^0 and any two parameter choices $\theta^+ \geq \theta^-$, for all $t > 0$, $a^t(\theta^+) \geq a^t(\theta^-)$ along any sample path, where in every period the largest optimal action is always chosen.*

As a consequence of this result we have a robustness result for the nondecreasing monotonic optimal policies of the discrete-time nonstationary Markov decision processes with a simplified maximal work constraint from Equation (7.4) described in Section 7.2. This result, given below, can be derived independently as a direct consequence of Theorem 43. However, we included for the sake of completeness the Proposition 49.

Corollary 50 *Let Y be a lattice, $\Theta \subset \mathbb{R}^m$ a sublattice, and X a partially ordered set with $Y_{x,\theta} \subseteq Y$ for any $(x,\theta) \in X \times \Theta$. If we have*

1. *for each $(x,\theta) \in X \times \Theta$ there exists $\max_{y \in Y} g(x,\theta,y)$*
2. *$Y_{x,\theta}$ is increasing in $(x,\theta) \in X \times \Theta$*
3. *$g(\cdot,x,\theta) : Y \to \mathbb{R}$ is supermodular in $y \in Y$ for each $(x,\theta) \in X \times \Theta$, and has increasing differences in $(y;x,\theta) \in Y \times (X \times \Theta)$*

then for any $x^+ \geq_X x^- \in X$, $\theta^- \leq_{\mathbb{R}^m} \theta^+ \in \Theta$, we have that
$\max \arg\max_{y \in Y} g(x^-,\theta^-,y) \leq \max \arg\max_{y \in Y} g(x^+,\theta^+,y)$.

Our weakly nondecreasing monotonic policies have an even more complicated structure since the set $\operatorname{Top} \arg\max_{a_{x_t} \in A_t(x_t)} w_t(x_t,a_{x_t})$ may contain points that are unordered with respect to each other, and then it can happen for some optimal decisions of the same review period to be unordered. If we strengthen the assumptions on the action space and we require it to be a lattice instead of partially bounded subset of $\mathbb{N}^q$, for all these unordered actions there will exist a greater action (their maximum) in the vector order that will be optimal. The problem is that the existence of such "max " actions implies in a lot of real-life situations (i.e. economics, physics) a large increase of the action space, and implicitly of the state space, as exemplified in Subsection 6.4. However, we are able to prove a non-chaotic behavior even without the lattice requirement.

Theorem 51 *Let $\Theta \subset \mathbb{R}^m$ be a sublattice, $X,Y := \cup_{(x,\theta) \in X \times \Theta} Y_{x,\theta}$ be posets (i.e. partially ordered sets), and $g : X \times \Theta \times Y \to \mathbb{R}$ a real valued superadditive function on $((\tilde{x},\tilde{\theta}),y) \in X \times \Theta \times Y_{x,\theta}$, , with $g(x,\theta,y) = 0$, for any $(x,\theta,y) \notin X \times \Theta \times Y_{x,\theta}$. If we have*

1. *for each $(x,\theta) \in X \times \Theta$ there exists $\max_{y \in Y} g(x,\theta,y)$*
2. *for any $\tilde{y} \in Y_{\tilde{x},\tilde{\theta}}$ and $y \leq_Y \tilde{y}$ there exists $x \in X \subset \mathbb{N}^q$, and $\theta \leq_{\mathbb{R}^m} \tilde{\theta} \in \Theta$ such that $y \in Y_{x,\theta}$*
3. *$Y_{x,\theta} \subset Y_{\tilde{x},\tilde{\theta}}$ for any $x \leq_X \tilde{x} \in X$, $\theta \leq_{\mathbb{R}^m} \tilde{\theta} \in \Theta$, and $Y_{x,\theta} = Y_{\tilde{x},\theta}$ for any $x,\tilde{x} \in X$ not comparable*
4. *for any $x \leq_X \tilde{x} \in X \subset \mathbb{N}^q$, $\theta \leq_{\mathbb{R}^m} \tilde{\theta} \in \Theta$, and $y \leq_Y \tilde{y}$, $(y,\tilde{y}) \in \left(Y_{\tilde{x},\tilde{\theta}} \times Y_{x,\theta}\right) \cup \left(Y_{x,\theta} \times Y_{\tilde{x},\tilde{\theta}}\right)$ we have that $y \in Y_{x,\theta}$ and $\tilde{y} \in Y_{\tilde{x},\tilde{\theta}}$ (i.e. the family is ascending)*

then for any $x^+ \geq_X x^-$ in X, $\theta^- \leq_{\mathbb{R}^m} \theta^+ \in \Theta$ and any $y^- \in \operatorname{Top} \arg \max_{y \in Y_{x^-,\theta^-}} g(x^-,\theta^-,y)$ there exists $y^+ \in \operatorname{Top} \arg \max_{y \in Y_{x^+,\theta^+}} g(x^+,\theta^+,y)$ such that $y^+ \geq_Y y^-$ in Y, or there is no element in $\operatorname{Top} \arg\max_{y \in Y_{x^+,\theta^+}} g(x^+,\theta^+,y)$ comparable with y^-.

Proof. Let $x^+ \geq_X x^- \in X$, let $\theta^- <_{\mathbb{R}^m} \theta^+ \in \Theta$, and choose $y \leq_Y y_{x^-}$ in the set $\text{Top}\arg\max_{y \in Y_{x^-,\theta^-}} g(x^-, \theta^-, y)$. Then there exists $x \in X$, and $\theta \leq_{\mathbb{R}^m} \theta^- \in \Theta$ such that $y \in Y_{x,\theta}$ (by hypothesis 2). Thus, $y \in Y_{x,\theta^-}$ (by hypothesis 3).

By definition of $\arg\max_{y \in Y} g(x^-, \theta^-, y)$ we have $g(x^-, \theta^-, y_{x^-}) - g(x^-, \theta^-, y) \geq 0$. Because $y \leq_Y y_{x^-}$ we have that $y \in Y_{x,\theta^-} = Y_{x^-,\theta^-}$, if x and x^- are not comparable (hypothesis 3); $y \in Y_{x,\theta^-} \subset Y_{x^-,\theta^-}$, for $x \leq_X x^-$ (by hypothesis 3); and $y \in Y_{x^-,\theta^-}$ for $x >_X x^-$ (by hypothesis 4 with Y_{x,θ^-}, Y_{x^-,θ^-}). Since $Y_{x^-,\theta^-} \subseteq Y_{x^+,\theta^+}$ (by hypothesis 3) we have that both $y, y_{x^-} \in Y_{x^+,\theta^+}$.

Since g is a superadditive function we also have: $g(x^+, \theta^+, y_{x^-}) + g(x^-, \theta^-, y) \geq g(x^+, \theta^+, y) + g(x^-, \theta^-, y_{x^-})$. Rewriting the second inequality as

$$g(x^+, \theta^+, y_{x^-}) \geq g(x^+, \theta^+, y) + [g(x^-, \theta^-, y_{x^-}) - g(x^-, \theta^-, y)]$$

and using afterwards the first inequality we obtain that: $g(x^+, \theta^+, y_{x^-}) \geq g(x^+, \theta^+, y)$ for all $y \leq_Y y_{x^-}$ in $\text{Top}\arg\max_{y \in Y_{x^-,\theta^-}} g(x^-, \theta^-, y)$. This concludes the proof. ■

A simpler behavior of the optimal sample paths occurs when in the parameterized Markov process we have separable objectives, in the sense that $w_t(x_t, a_t, \theta) = g(x_t, a_t) + h_\theta(x_t)$.

Proposition 52 *Let $X, Y \in \mathbb{R}^q$ be posets, $g : X \times Y \to \mathbb{R}$ a supermodular/superadditive function, and $h : X \to \mathbb{R}$ a nondecreasing function with respect to the partial order on X. If for each $x \in X$ there exists $\max_{y \in Y} g(x, y)$ then for any $x^+ \geq_X x^-$ in X, and any $y^- \in \left\{ \text{Top}\arg\max_{y \in Y} g(x^-, y) + h(x^-) \right\}$ there exists $y^+ \in \left\{ \text{Top}\arg\max_{y \in Y} g(x^+, y) + h(x^+) \right\}$ such that $y^+ \geq_Y y^-$ in Y, or there is no action in $\text{Top}\arg\max_{y \in Y} g(x^+, y)$ comparable with y^-.*

Proof. $h : X \to \mathbb{R}$ is a constant function in $y \in Y$. Thus, we have the following $\text{Top}\arg\max_{y \in Y} [g(x^-, y) + h(x^-)] = \text{Top}\arg\max_{y \in Y} g(x^-, y)$ and we can apply now Theorem 51 to conclude the proof. ■

Chapter 7

Markovian Control of Concurrent NPD

7.1 Introduction

In this chapter we focus on a concurrent NPD (i.e. without precedence constraints), consisting of N concurrent design tasks as described in Chapter 5 Subsection 5.4.1. To control it we use a discrete-time, finite horizon non-stationary Markov decision process.

For enabling a more efficient computation of optimal policies in Markov models of sequential decision processes, one is often interested in finding structured policies (monotonic, convex, etc.). When monotonic policies can be found, as in (Puterman, 1994), a monotonic backwards induction algorithm can be used. It is known that in general dynamic programming problems often the solutions are extremely complicated to be used for deriving insights about the given problem (see for a survey (Dixit and Pindyck, 1996)). Moreover, in general in higher dimensions the monotonicity of the decision rules does not imply a stable structure of the optimal paths (i.e. an increase in an exogenous parameter will imply to a uniform increase of the optimal path). (Friedman and Johnson, 1997) prove that the existence of monotonic policies leads to robust optimal paths, under several assumptions including the one that the action spaces are sublattices of $\mathbb{R}^m$ for any state. This lattice type of action set is required also by the monotonic backwards induction algorithm. We show that such a strong condition can in real-life eliminate all the possible computational gains due to the existence of monotonic optimal policies. Thus, assuming no lattice constraint, we derive the general conditions of obtaining instead of monotonic optimal policies, weakly monotonic optimal policies. This will lead us to a new weakly-monotonic backwards induction algorithm, and we are able to prove, in this chapter, that our results exhibit as well this sought-after property of robustness.

The NPD related results are the optimality of monotonic (in the partial order on the state space) policies in the case of a simplified workload constraint, and respectively weakly monotonic otherwise. The existence of the first type of monotonic nondecreasing optimal policies confirms an intuitive "greedy" feature of the optimal control policy "The performance requirement of a controller increases with the number of performance levels already achieved by the engineers". The existence of the weakly nondecreasing optimal control poli-

cies refines the early heuristic. According to them, the NPD aggregate decision maker may take actions which are not directly comparable (neither increasing, nor decreasing) from the point of view of performance levels required. The actions are only comparable from the potential cost/reward point of view, even if the states in which these decisions are taken are comparable from the performance levels achieved point of view. The second type of optimal policy leads to significant improvements in the computational efficiency as one can see from the weakly monotone backward induction algorithm 6.4.

We recall from Chapter 6 that our approach for obtaining monotonic and weakly monotonic optimal policies is based on the monotonicity research of (White, 1980; Topkis, 1998). This means that we formulate our control problem in a dynamic programming setting, and we try to establish the supermodularity of the objective function. In microeconomics, and in theories of production and consumer choice, supermodularity of an utility function is equivalent to products being *substitutes*. The substitute notion appears very intuitive in the case of a concurrent NPD situation, where after we finish all the design tasks at their minimal performance levels, having done more levels of performance for one design task compensates for doing less performance levels of a different one.

Finally, we prove that the weakly monotonic optimal paths are robust to the variation of the safety margin for achieving the new product at the deadline. Using simulation studies, we shown the robustness of the weakly-monotonic optimal paths with respect to small variations of the solving rate of the design activities. We also investigated the optimal value variation function of the degree of specification of the characteristics of the new product at the beginning of the NPD.

This chapter is organized as follows. In Section 7.2 we prove that the Markov decision model constructed for the parallel case of the NPD control problem (respectively a simplified version of it) satisfies all the conditions for the existence of weakly monotonic (respectively of monotonic) nondecreasing optimal policies. In Section 7.3 we prove that it also satisfies the robustness conditions for its essential parameters.

7.2 Nondecreasing optimal policies for the control of NPD without precedence constraints

In this section we investigate the properties of the value function arising in the complex nonstationary discrete-time Markov decision model constructed for the control of a NPD project without precedence constraints in Chapter 5 Subsection 5.4.1. We prove for it the existence of weakly monotonic nondecreasing optimal policies. Monotonic nondecreasing optimal policies exist as well for a slightly modified form of the Markov decision process from Subsection 5.4.1, but as shown in Section 6.4 their existence does not lead to computational gains.

7.2.1 Problem setting

We recall from Chapter 5 Subsection 5.4.1 that the NPD situation considered consists of N concurrent design tasks allocated to team of engineers from the first review period. Unplanned design tasks may arrive during each review period, and they will become available to the team of engineers at the beginning of the next review period. For each review period we take into account only an ”average” type of task for that review period (i.e. statistically

identical design tasks with a common performance level structure, and an identical market payoff structure). They are all concurrent with the initial design tasks allocated to the team of engineers. Their arrival is given by a Poisson review period-dependent arrival process of rate $\varsigma(t)$. In later review periods the rate $\varsigma(t)$ decreases. Moreover, as in the conditions of Proposition 19 during each review period $[t, t+1)$ we take into account at most a number K_t of unplanned design tasks which arrive with a probability greater than a given threshold $\vartheta(t) > 0$ and $\sum_{t=0}^{T-1} K_t = N - \overline{N}$.

Due to the curvilinear dependency between the productivity and the workload of the team of engineers, the targets on design tasks realization have to satisfy a minimal requirement of work during one review period (see equation (5.4) Remark 21, Subsection 5.4.1). No budget constraint will be considered in the Markov decision process. *An important* supplementary *assumption concerning only this NPD case* was made in Chapter 5 Subsection 5.4.1.

Supplementary Assumption:

For each design task n its maximal performance level $L_{\max}(n,t)$ is large enough so that even if the team of engineers will work with all capacities on it, $L_{\max}(n,t)$ cannot be achieved earlier than the deadline with the probability $\min_{t=0,\ldots,T} \beta(t)$. However, this holds only for the planned or already arrived unplanned design tasks. Since N is an upper bound, we have to initially set to zero all the parameters depending on an virtual $n \in \{\overline{N}+1, \ldots, N\}$.

This assumption requires no artificial preset bounding of the action spaces at each decision point. Moreover since there is a total possible number of activities to be performed during one review period (depending on the values of $\beta(t)$), there will a total possible number of levels to be done for each design task. Thus, the action space will be upper bounded. Without this enlargement of the action space, we are not able to prove in all the cases the existence of structured optimal policies.

All the other assumptions and notation are given in Section 5.3. Thus, the team of engineers will always perform the tasks allocated to it, according to a priority order (see Chapter 5 Section 5.2). We recall from Chapter 2 Section 2.3 that each design task has different levels of performance, giving the quality of its execution. Each performance level consists of a *list of planned activities* with solving times random variables independent identically exponentially distributed. To attain a performance level, we assume that the engineer has to sequentially execute the design task at all previous performance levels. For each design task a *minimal performance level* has to be achieved, in order to have a fully functional new product. New activities arrive at each of the design tasks in progress during the review period t according to a Poisson process of rate $\lambda(t)$.

We recall from Chapter 5 Section 5.3 the assumptions and the notation related to this particular case of NPD project. The notation was influenced as well by the fact that we gave up to the budget constraint and by the assumptions we made in Chapter 5 Section 5.4.1. We also assumed that there is an identical number of activities for all performance levels of the same design task. Thus, assuming the previous levels already solved, the solving times of all the performance levels of the same design task will be i.i.d. random variables Erlang distributed.

Input parameters (global variables) **:**

T : the total number of review periods (review periods are numbered from 0 to $T-1$);

M : the total number of engineers;

$\overline{N}$: the initial number of design tasks;

N : an upper bound for the maximum number of design tasks during the whole project;

$L_{\max}(n)$: the maximal number of performance levels of the initial design task n; $n = 1, \ldots, \overline{N}$;

$l_{\min}(n)$: the minimal performance level at which the design task n must be performed in order to obtain a functional product; $n = 1, \ldots, N$

$n_a(n)$: the number of sequential activities planned for solving any performance level l of the design task n, assuming the previous levels already solved; $\forall\ n = 1, \ldots, N$; $l = 1, \ldots, L_{\max}(n)$;

$a_{\min}(n)$: the minimal requirement of work to be done during a review period (see Chapter 5);

μ : the rate of the exponential distribution of an activity solving time.

Since N is an upper bound, at time $t = 0$ we set to zero all the parameters depending on a virtual $n \in \{\overline{N}+1, \ldots, N\}$.

Input parameters (at the beginning of review period t)**:**

$\beta(t)$: the required current safety margin for the probability of exceeding the maximal team solving capacity; $\beta(t) \in (0,1)$;

$\varsigma(t)$: the current rate of the Poisson review period-dependent arrival process of unplanned design tasks; $\varsigma(t) > 0$;

$\Lambda_0 := \{1, \ldots, \overline{N}\}$: the set of design tasks set allocated at $t = 0$;

$\Omega(t-1)$ is the set of newly arrived design tasks (during review period $t-1$) concurrent with design tasks allocated initially;

$L_{\max}(n,t)$: the maximal number of performance levels of the design task n; $L_{\max}(n,t) = L_{\max}(n)$ for $n = 1, ..., \overline{N}$ and $L_{\max}(n,t) = 0$ if $n \notin \Lambda_0 \cup \Omega(0) \ldots \cup \Omega(t-1)$ (i.e. there is place reserved for design tasks not planned or not arrived yet up to the upper bound N but we set to zero the maximal performance level depending on a virtual design task n);

$N(t-1)$: the random variable giving the unplanned design tasks number added since the NPD project beginning until the end of review period $t-1$, $[t-1, t)$;

$l(\cdot, t) : \{1, ...N\} \rightarrow \{\mathbb{N} \cup \{-1\}\}$: the achieved performance design task level function, where $0 \leq l(n,t) \leq L_{\max}(n,t)$, for $n \in \{1, ..., \overline{N} + N(t-1)\}$ and by convention we define $l(n,t) = -1$ for $n \in \{\overline{N} + N(t-1) + 1, \ldots, N\}$;

$\lambda(t)$: the review-period dependent Poisson arrival of unplanned activities for all the design tasks allocated to the engineers ($\lambda(t)/\mu < M$).

Notation (at the beginning of review period t**):**

$\digamma_t := \left\{ n \geq \overline{N} \,|\overline{N} + E[N(t-1)] + 1 \leq\ n\ \leq \overline{N} + E[N(T-1)] \right\}$: the estimated set of all unplanned design tasks for NPD project;

S_n : the solving time of an arbitrary performance level of the design task n, assuming the previous levels already solved.; $n = 1, ..., N$. They are independent random variables Erlang-$(n_a(n), \mu)$.

The decision time points: The decision points are equidistant and the horizon of the problem is finite. The decision point t corresponds to the beginning of review $t+1$. Say $t \in \{0, 1, ...T-1\}$.

The immediate **rewards,** the final reward, **the nonstationary transition probabilities**, and the **expected total reward criterion** of this process, as well as a discussion on the constraints defining the state and action space are provided in Section 5.3.

The state space and the action space: The state set $X(t)$ at moment t and the action set $A_t(x_t)$ in the state $x_t \in X(t)$ are defined two main constraints:

- the target performance level of each planned or newly arrived design task n is greater than $\min(l_{\min}(n,t), l(n,t))$, and smaller than $L_{\max}(n,t)$
- the remaining workload of the team of engineers should not exceed their maximal solving capacity with the probability greater than safety margin $\beta(t)$.

For the state space the workload constraint is computed using the minimal performance levels, $l_{\min}(\cdot,t)$, while for the action space it is used the target performance levels, $(l(\cdot,t)+a(\cdot,t))^+$.

For $t=0$: $X(0)=\{x_0\}=(0_{\mathbb{R}^{\overline{N}}}, -1_{\mathbb{N}^{N-\overline{N}}})$.

For $t \in \{1,\dots,T\}$ the state $x_t \in X(t)$ describes how many performance levels $l(n,t)$ were solved for each design task n:

$$X(t)=\left\{ x_t \left| \begin{array}{c} x_t=(l(1,t),..,l(n,t),...,l(N,t)) \in \{\mathbb{N}\cup\{-1\}\}^N \text{ and} \\ -1 \le l(n,t) \le L_{\max}(n,t),\ n \in \{1,...,N\} \\ l(n,t) \ge 0, \forall n \in \{1,...,N\} \text{ s.t. } L_{\max}(n,t)>0 \\ \Pr\left\{ \sum\limits_{n\in\{1,...,\overline{N}+N(t-1)\}\cup F_t} \sum\limits_{i=l(n,t)+1}^{l_{\min}(n)} S_n \le M(T-t) \right\} \ge \beta(t) \end{array} \right. \right\} \quad (7.1)$$

$$X(T)=\left\{ x_T \left| \begin{array}{c} x_T=(l(1,T),..,l(n,T),...,l(N,T)) \in \{\mathbb{N}\cup\{-1\}\}^N \text{ and} \\ l_{\min}(n) \le l(n,T) \le L_{\max}(n,t), \forall n \in \{1,...,N\} \text{s.t. } l(n,T) \ge 0 \end{array} \right. \right\}$$

For $t \in \{0,...T-1\}$, $x_t=(l(1,t),..,l(n,t),...,l(N,t)) \in X(t)$ the action a_t in the state x_t decides how many other levels above x_t we want to perform. The performance level up to which the design task n may be solved after the action a_t was taken is $(l(n,t)+a_t(n))^+$.

$$A_t(x_t)=\left\{ a_t \left| \begin{array}{c} a_t=(a_t(1),..,a_t(n),...,a_t(N)) \in \mathbb{N}^N \\ 0 < a_{\min}(n,t) \le a_t(n), \forall n \in \{1,...,\overline{N}+N(t-1)\} \\ l(n,t)+a_t(n) \le L_{\max}(n,t), \forall n \in \{1,...,N\} \text{ s.t. } l(n,t) \ge 0 \\ \Pr\left\{ \sum\limits_{n\in\{1,...,\overline{N}+N(t-1)\}\cup F_t} \sum\limits_{i=1}^{a_t(n)} S_n \le M\cdot(T-t) \right\} \ge \beta(t) \end{array} \right. \right\} \quad (7.2)$$

We recall from Section 5.3 and Subsection 5.4.1 that the state and action space descriptions are sensitive to the arrival of unplanned design tasks, up to the upper bound N. If at a decision point t an unplanned design task n just appeared it will be numbered in an increasing way from $\overline{N}+N(t-1)$ up to N, and $l(n,t)$ is changed from -1 to zero. Thus, at the next decision point its corresponding actions might be greater than zero, and the team of the engineers might start working on it.

We also recall that this *Markov decision process* has *partially ordered state space, and action space*. They are bounded subsets without holes of the lattices $\{\mathbb{N}\cup\{-1\}\}^N$ and respectively $\mathbb{N}^N$ endowed with the *componentwise partial ordering* (i.e. $x,y \in \mathbb{N}^N$ and $x \le y$ iff $x_n \le y_n\ \forall n \in \{1,\dots,N\}$, as in Definition 32). We remark that the spaces

$\{\mathbb{N} \cup \{-1\}\}^N$ and $\mathbb{N}^N$ are isomorphic and thus all the properties from the previous chapter hold.

The action space description (7.2) of our Markov model may lead to the existence of unordered points in the set $\arg\max_{a_{x_t} \in A(x_t)} \left\{ \sum_{x_{t+1} \in X(t+1)} p_t(x_t, a_t, x_{t+1}) \cdot u^*(x_{t+1}) \right\}$, as well as in its Top (see Definition 31). These sets, obtained from the optimality equations, are searched by all the backward induction algorithms. So, some optimal decisions of the same review period may be unordered. If we enlarge the action space to a lattice instead of a partially bounded subset of $\mathbb{N}^N$, for all these unordered actions there will exist a greater action (their maximum) in the componentwise partial ordering that will be optimal. This will happen, however, at the expense of a large increase of the action space, which leads in our case to a numerical intractable model.

We may prove the existence of monotonic optimal policies for our NPD control problem, by using instead of the finite solving capacity requirement constraint for the current review period

$$\Pr\left\{ \sum_{n \in \{1,\ldots,\overline{N}+N(t-1)\} \cup \boldsymbol{F}_t} \sum_{l=1}^{a_t(n)} S_n \leq M\,(T-t) \right\} \geq \beta(t) \tag{7.3}$$

a simplified maximal work requirement of the type:

$$\text{where } \Pr\left\{ \sum_{n \in \{1,\ldots,\overline{N}+N(t-1)\} \cup \boldsymbol{F}_t} \sum_{l=1}^{a_{\max,t}(n)} S_n \leq M\,(T-t) \right\} \geq \beta(t) \tag{7.4}$$
$$a_{\max,t}(n) \geq a_t(n) \geq a_{\min}(n) \geq 0, \forall n = 1, ..., N$$

If in the action space description (7.2) we consider the simplified maximal work requirement restriction (7.4) the action space will be a sublattice and we can apply the results of (Topkis, 1998) (see Subsection 6.2) to prove the existence of optimal decision rules which are nondecreasing in the state of the system. Otherwise, the action space will not be anymore a sublattice, but will be an upper bounded inf-sublattice. In this case we can use the less restrictive set of requirements from Proposition 48 to prove the existence of optimal decision rules which are weakly nondecreasing in the state of the system.

Consequently, in the case of using a simplified maximal work requirement we may solve this problem with a monotonic multi-dimensional algorithm in the sense of (Topkis, 1998), by naturally extending the unidimensional algorithm presented in (Puterman, 1994). Otherwise, we may use the new weakly monotonic nondecreasing backward induction algorithm, speeding up the whole processing by a significant factor. The last one will not use the existence of monotonic optimal policies, and hence nor the "max" actions existence, but the existence of weakly monotonic ones. This allows it to look for candidate actions for the $\arg\max$ for a given state x in the action space from which we extract all the inferior cones of the elements of the $\arg\max$ of all states x' which are smaller than x, instead of looking for it in the whole available action space.

7.2.2 Analytical results

This subsection is organized as follows. First we prove that our Markov decision model satisfies function of the type of action space description (either with the constraint

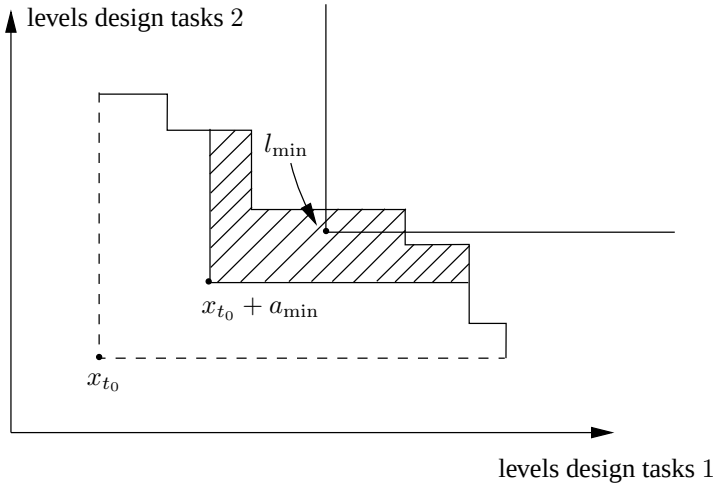

Figure 7.1. Action spaces $A(x_{t_0})$ for $N = 2$, case 1

(7.4), or (7.3)) the conditions given in subsections 6.2, 6.3 for nondecreasing monotonic, and respectively weakly monotonic optimal control policies. Thus we establish the following NPD related results: the optimality of policies that are nondecreasing monotonic for the case of a simplified maximal work requirement, and respectively weakly monotonic nondecreasing otherwise (in the partial order on the state space) for the finite horizon. Refining the intuitive "greedy" feature of the nondecreasing monotonic optimal policy "The performance requirement of a controller increases with the number of performance levels already achieved by the engineers", the second type of optimal policy leads to gains in computational efficiency as one can see from the weakly monotone backward induction algorithm 6.4.

Lemma 53 *All the increasing subsets of $X(t) \subseteq \{\mathbb{N} \cup \{-1\}\}^N$, $t \in \{1, ..., T\}$ are of the form $K_{s_1,...,s_q} = \{x \in X(t) \,|\exists i \in \{1, ..., k\}$ s.t. $x \geq s_i\}$, where $s_1, ..., s_q$ are not comparable elements of $\{\mathbb{N} \cup \{-1\}\}^N$, and $q \in \mathbb{N}$ arbitrary.*

Proof. "$\supseteq$" This implication holds obviously since for any $\tilde{x} \in X(t)$ and $x \leq \tilde{x}$ we have from the transitivity property of a partial order relationship that $\tilde{x} \geq x \geq s_i \rightarrow \tilde{x} \geq s_i$ for an arbitrary $i = 1, ..., k$

"$\subseteq$" $X(t)$ is a subset of the infinite infimum-lattice $\{\mathbb{N} \cup \{-1\}\}^N$. If we can prove that $X(t)$, $t \in \{1, ..., T\}$ is a bounded, without holes subset of $\{\mathbb{N} \cup \{-1\}\}^N$ we are almost done. Under this assumption there are only a finite number of incomparable elements in any subset of $X(t+1)$. Thus, K will be spanned by those elements since being an increasing set if it contains s_i, s_j incomparable will contain any $x \geq s_i$ or s_j.

The constraint (7.2) leads to upper bounded, without holes action spaces for our Markov decision process. Moreover since there is a total maximal number of activities to be performed during one review period (depending on the value of β), there will a total maximal number of levels to be done for each design task. Thus, the action space will have as an upper bound a nondecreasing surface. Then $X(t)$ is a bounded subset of the infinite lattice $\{\mathbb{N} \cup \{-1\}\}^N$ since from the definition of $x(t) = (l(1,t), .., l(n,t), ..., l(N,t)) \in X(t)$ we

have that $l(n,t) \leq l(n,t-1) + a(n,t-1)$, $n = 1,...,N$. Thus, each state space will also have as an upper bound a nondecreasing surface, and in $\{\mathbb{N} \cup \{-1\}\}^2$ (i.e. for two design tasks) this will have the shape of a step function. ■

Definition 54 *A set of the form $K_s = \{x \in X(t) \,|x \geq s\}$ is called an elementary increasing subset of the action space $X(t)$.*

Corollary 55 *All the increasing subsets of $X(t)$, $t \in 1,...,T$ are of the form $\bigcup_{j=1}^{q} K_{s_j}$, where $s_1,...,s_q \in \{\mathbb{N} \cup \{-1\}\}^N$.*

Proof. Obvious due to the construction of the increasing set
$K_{s_1,...,s_q} = \{x \in X(t) \,|\exists i \in \{1,...,k\} \text{ s.t. } x \geq s_i\}$ ■

Proposition 56 *For the Markov decision process constructed for the control of a NPD project without precedence constraints as described Subsection 7.2.1 we have (with respect to the partial order on $X(t) \subset \{\mathbb{N} \cup \{-1\}\}^N$):*

a) $\rho_t(\cdot,\cdot)$ is supermodular in $(x_t, a_t) \in X(t) \times A_t(x_t)$, $t \in \{0,1,...T-1\}$.

b) $\rho_t(\cdot, a_t) : X(t) \to \mathbb{R}$ is nondecreasing, $t \in \{0,1,...T-1\}$

c) $\rho_T(\cdot) : X(T) \to \mathbb{R}$ is nondecreasing.

Proof. The properties hold obviously due to the definition of the rewards and to the expected total reward criterion we use.

a) +b) $\rho_t(x_t, a_t, x_{t+1}) \overset{def}{=} 0$, for each $(x_t, a_t) \in X(t) \times A_t(x_t)$, $x_{t+1} \in X(t+1)$, $t \in \{0,1,...T-1\}$

c) $\rho_T(\cdot)$ is by definition a nondecreasing function with respect to the partial order on $\{\mathbb{N} \cup \{-1\}\}^N$ ■

Proposition 57 *For the Markov decision process constructed for the control of a NPD project without precedence constraints as described in Subsection 7.2.1 we have that $A_t(x_t) = A_t(\tilde{x}_t) = A_t$, for any $x, \tilde{x} \in X(t) \subset \{\mathbb{N} \cup \{-1\}\}^N$, $t \in \{0,1,...T-1\}$.(i.e. as the state of the system increases the number of decisions available to the decision-maker during one review period does not decrease)*

Proof. From its definition $A_t(x_t)$ is a bounded, without holes subset of $K_{a_{\min}} \subset \mathbb{N}^N$, having the same lower frontier as the elementary increasing set $K_{a_{\min}} := \{x | x \geq a_{\min}\}$.

If $x, \tilde{x} \in X(t)$ means that from both x, $\tilde{x}$ we can arrive in the rewarded region: $R = \{(l_1,...,l_N) | l_n \geq l_{\min}(n), \forall n = 1,...,N\}$ in $T-t$ stages.

Since the solving times of the design tasks are considered independent random variables, and each of them is a sum of independent identically exponentially distributed random variables (i.e. the solving times of the activities), the $\sum_{n \in \{1,...,N\} \cup \Gamma_t} \sum_{i=1}^{a_{\max,t}(n)} S_n$ from the inequality (7.2) is distributed Erlang-$\left(\overline{k}, \mu\right)$,where $\overline{k} := \sum_{n \in \{1,...,N\} \cup \Gamma_t} \sum_{i=1}^{a_{\max,t}(n)} n_a(n)$. Thus, for a maximal chosen action, $a_{\max,t}$, the inequality (7.2) is equivalent to

$$\sum_{j=0}^{\overline{k}-1} \frac{[\mu(T-t)M]^j}{j!} e^{-\mu(T-t)M} \leq 1 - \beta(t),$$

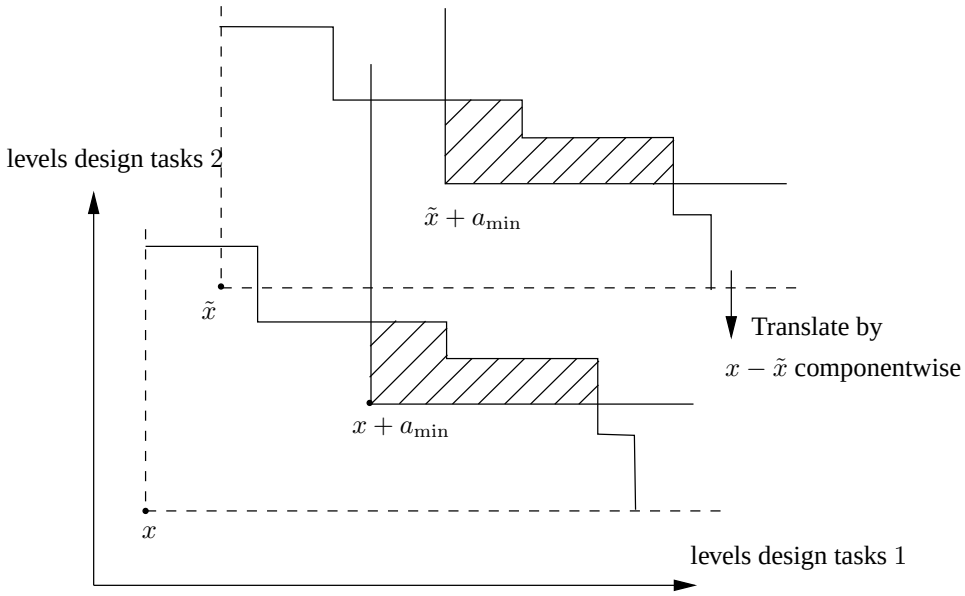

Figure 7.2. Non-decreasing action spaces for $N = 2$

which implies a maximal number of design activities to be done in $(T - t)$ review periods. Since we assumed that for any particular design task the number of activities needed to solve each of its levels is the constant, the maximal number of possible levels to be achieved in $(T - t)$ review periods is constant as well. There exists as well a maximal number of possible levels of the set of actions which decides that the engineers should perform more than the minimal requirement $a_{\min}(t)$.

According to *the supplementary assumption concerning the control of NPD without precedence constraints* for each already arrived design task n its maximal performance level $L_{\max}(n, t)$ is large enough so that even if the team of engineers will work with all capacities on it, $L_{\max}(n, t)$ cannot be achieved earlier than the deadline with the probability $\min_{t=0,...,T} \beta(t)$. So, there is no preset bounding of the action spaces at each decision point. Thus, they are only naturally bounded by the engineers finite capacity and we have $A_t(x_t) = A_t(\tilde{x}_t)$ (see Figure 7.2).

For $N = 2$ we can easily see that as the state of the system increases the number of decisions available to the decision-maker does not decrease. ■

Proposition 58 *Let us consider no unplanned design tasks arrival in the Markov decision process constructed for the control of a NPD project without precedence constraints as described in Subsection 7.2.1. Then the function defined by* $\sum_{x_{t+1} \in K} p_t(x_t, a_t, x_{t+1})$ *is nondecreasing in* $x_t \in X(t)$*, for any* K *increasing subset of* $X(t + 1)$*,* $t \in \{0, 1, ...T - 1\}$ *(with respect to the partial order on* $X(t) \subseteq \{\mathbb{N} \cup \{-1\}\}^N$*).*

Proof. In the case of no unplanned design tasks arrival, from the Corollary 20 the points in $X(t+1)$ for which we have $p_t(x_t, a_t, x_{t+1}) \neq 0$ belong to the set

$$R_{x_t,a_t+x_t} \stackrel{def}{=} \left\{x \in \mathbb{N}^N | x_t \leq x \leq a_t + x_t\right\}.$$

Consider $x \leq \tilde{x}$ such that $x, \tilde{x} \in X(t)$. We first prove that for any elementary increasing subset K_s we have:

$$\sum_{x_{t+1} \in K_s} p_t(x, a_t, x_{t+1}) \leq \sum_{x_{t+1} \in K_s} p_t(\tilde{x}, a_t, x_{t+1}) \tag{7.5}$$

For proving (7.5) we will discuss several cases:

Case 1. $x, \tilde{x} \in K_s$

Then we have $x + a_t$, $\tilde{x} + a_t \in K_s$. As a consequence R_{x,a_t+x}, $R_{\tilde{x},a_t+\tilde{x}} \subset K_s$ and both sums in (7.5) are taken over all their terms and according to Corollary 20 they are equal.

Case 2. $x + a_t$, $\tilde{x} + a_t \notin K_s$

Then $x, \tilde{x} \notin K_s$ and we have $R_{x,a_t+x} \cap K_s = R_{\tilde{x},a_t+\tilde{x}} \cap K_s = \varnothing$. Thus, both sides equal zero.

Case 3. x, $\tilde{x} \notin K_s$ but $\tilde{x} + a_t \in K_s$.

Since $x \leq \tilde{x}$, $x + a_t$ may or may not belong to K_s .

If $x + a_t \notin K_s$, $R_{x,a_t+x} \cap K_s = \varnothing$ and

$$\sum_{x_{t+1} \in K_s} p_t(x, a_t, x_{t+1}) = 0 \leq \sum_{x_{t+1} \in R_{\tilde{x},a_t+\tilde{x}} \cap K_s} p_t(\tilde{x}, a_t, x_{t+1})$$

since $R_{\tilde{x},a_t+\tilde{x}} \cap K_s \neq \varnothing$ and $p_t(\tilde{x}, a_t, x_{t+1}) \geq 0$.

If $x + a_t \in K_s$ we prove that for each point in the intersection $R_{x,a_t+x} \cap K_s$ we have a unique point of equal probability in the intersection $R_{\tilde{x},a_t+\tilde{x}} \cap K_s$. And since the terms of nonzero probability in $\sum_{x(t+1) \in K_s} p_t(x, a_t, x_{t+1})$ are the ones belonging to $R_{x,a_t+x} \cap K_s$ we conclude the proof. For $N = 2$ we can draw an easy graphical description of the sets involved in the proof.

Let $z \in R_{x,a_t+x} \cap K_s$. Then $z = x + y(t)$ (i.e. componentwise addition), where $a_{\min}(t) \leq y(t) \leq a_t$, and $x + y(t) \geq s$. Now consider the uniquely determined $\tilde{z} = \tilde{x} + y(t)$ which belongs to $R_{\tilde{x},a_t+\tilde{x}}$. Since K_s is an increasing set and $x \leq \tilde{x}$ implies $x + y(t) \leq \tilde{x} + y(t)$ we have $\tilde{z} \in K_s$. We claim that the $p_t(\tilde{x}, a_t), \tilde{z}) = p_t(x, a_t, z)$.

Since $x \leq z \leq x + a_t$, and $\tilde{x} \leq \tilde{z} \leq \tilde{x} + a_t$ from the Corollary 20 we have the equality: $p_t(x, a_t, z) = p_t(z - x, \lambda(t), i_{a_t}) = p_t(y(t), \lambda(t), i_{a_t}) = p_t(\tilde{z} - \tilde{x}, \lambda(t), i_{a_t}) = p_t(\tilde{x}, a_t, \tilde{z})$.

Finally, if we take an arbitrary K an increasing set in $X(t+1)$ from Corollary 55 there exist $s_1, ..., s_q \in \mathbb{N}^N$ such that $K = \bigcup_{j=1}^{q} K_{s_j}$. Since the elements $s_1, ..., s_q$ are not comparable one with the other the number of possible cases remains the same, as well as the proofs which are based on translation arguments. ■

Proposition 59 *Let us consider no unplanned design tasks arrival in the Markov decision process constructed for the control of a NPD project without precedence constraints as*

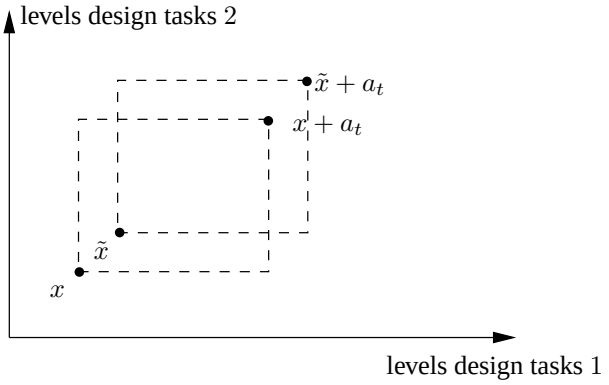

Figure 7.3. Verifying the nondecreasingness of $\sum_{x_{t+1}} p(x_t, a_t, x_{t+1})$

described in Subsection 7.2.1. Then the function defined by $\sum_{x_{t+1}\in K} p_t\left(x_t, a_t, x_{t+1}\right)$ *is superadditive in* $(x_t, a_t) \in X\left(t\right) \times A_t\left(x_t\right)$*, for any* K *increasing subset of* $X(t+1)$*,* $t \in \{0, 1, ..., T-1\}$ *(with respect to the partial order on* $X(t) \subseteq \{\mathbb{N} \cup \{-1\}\}^N$*).*

Proof. In the case of no unplanned design tasks arrival, from the Corollary 20 the points in $X(t+1)$ s.t. $p_t\left(x_t, a_t, x_{t+1}\right) \neq 0$ belong to

$$R_{x_t, a_t+x_t} \stackrel{def}{=} \left\{x \in \mathbb{N}^N | x_t \leq x \leq a_t + x_t\right\}.$$

Consider $x \leq \tilde{x}$ such that $x, \tilde{x} \in X\left(t\right)$, $a \leq \tilde{a} \in A_t(x) = A_t(\tilde{x})$ (see Proposition 57). Let K an increasing set in $X\left(t+1\right)$. Let us first prove that for any elementary increasing subset K_s we have:

$$\sum_{x_{t+1}\in K_s} p_t\left(\tilde{x}, \tilde{a}, x_{t+1}\right) - \sum_{x_{t+1}\in K_s} p_t\left(\tilde{x}, a, x_{t+1}\right) \geq$$

$$\geq \sum_{x_{t+1}\in K_s} p_t\left(x, \tilde{a}, x_{t+1}\right) - \sum_{x_{t+1}\in K_s} p_t\left(x, a, x_{t+1}\right) \tag{7.6}$$

then due to the structure of K the superadditivity is proved.

For proving (7.6) we will discuss several cases. But first we make a general observation on the structure of the probability sums involved in the inequality (7.6)

$$\sum_{x_{t+1}\in K_s} p_t\left(x, \tilde{a}, x_{t+1}\right) - \sum_{x_{t+1}\in K_s} p_t\left(x, a, x_{t+1}\right)$$

$$= \sum_{\substack{x\leq x_{t+1}\leq x+\tilde{a} \\ x_{t+1}\in K_s}} p_t\left(x, x_{t+1}, \lambda(t), i_{\tilde{a}}\right) - \sum_{\substack{x\leq x_{t+1}\leq x+a \\ x_{t+1}\in K_s}} p_t\left(x, x_{t+1}, \lambda(t), i_a\right)$$

According to Corollary 20 we have for a fixed μ and $\lambda(t)$ that:

$$p_t\left(x, a, x_{t+1}\right) = p_t(x_{t+1} - x, i_a) = p \cdot P_t(x_{t+1} - x) + (1-p) \cdot \frac{P_t(i_a)}{\varkappa(x_{t+1} - x)},$$

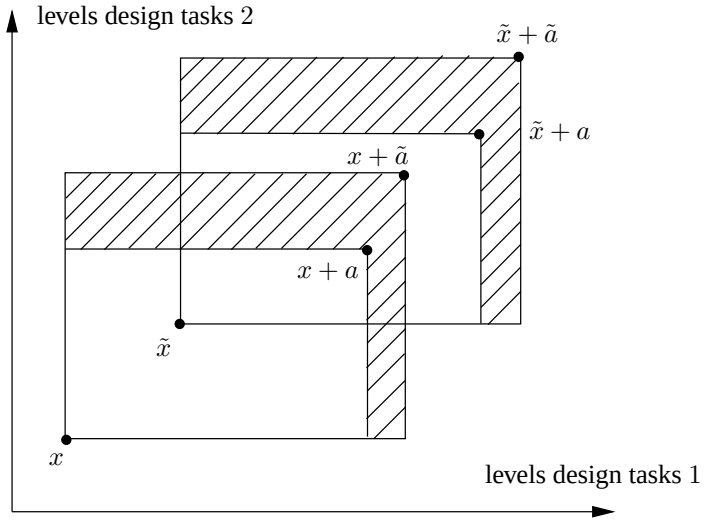

Figure 7.4. Verifying the superadditivity of $\sum_{x_{t+1}} p(x_t, a_t, x_{t+1})$ (the case $\tilde{x} \leq x+a \leq \tilde{x}+a$)

for $i_a \geq 2$, and $p_t(x, a, x_{t+1}) = p_t(x, x_{t+1})$ otherwise. In this last case the inequality (7.6) can be rewritten as:

$$\sum_{\substack{x_{t+1}\in K_s \\ x_{t+1}\in R_{\tilde{x},\tilde{x}+\tilde{a}}\setminus R_{\tilde{x},\tilde{x}+a}}} P_t(x_{t+1}-\tilde{x}) \geq \sum_{\substack{x_{t+1}\in K_s \\ x_{t+1}\in R_{x,x+\tilde{a}}\setminus R_{x,x+a}}} p_t(x_{t+1}-x) \geq 0 \qquad (7.7)$$

Case 1. $x+\tilde{a}, \tilde{x}+\tilde{a} \notin K_s$

It holds trivially since then both sides of 7.6 equal zero.

Case 2. $x, \tilde{x} \in K_s$

Then since K_s is an increasing set $R_{x,x+\tilde{a}}\setminus R_{x,x+a}, R_{\tilde{x},\tilde{x}+\tilde{a}}\setminus R_{\tilde{x},\tilde{x}+a} \subset K_s$. As a consequence both sums in (7.6) are taken over all their terms and we have two subcases.

Case 2a.

If the two actions have the same number of zero components, or $i_a < 2$ than using the inequality (7.7) where for $\mu, \lambda(t)$ constant $P_t(x_{t+1}-x)$ depends only on the difference of activities in between x_{t+1} and x. This inequality has an obvious graphical interpretation in any dimension.

We claim that for each point in the $R_{x,x+\tilde{a}}\setminus R_{x,x+a}$ we have a point of equal probability in the $R_{\tilde{x},\tilde{x}+\tilde{a}}\setminus R_{\tilde{x},\tilde{x}+a}$. If this claim is true we conclude the proof, since the terms of nonzero probability in (7.6) are the ones belonging to $R_{x,x+\tilde{a}}\setminus R_{x,x+a}$ we conclude the proof. For $N=2$ we can draw an easy graphical description of the sets involved in the proof.

Let $z_0 \in R_{s,x+\tilde{a}}\setminus R_{s,x+a}$. Now consider the translation $T(z) = (z-x)+\tilde{x}$ on z_0 and on the domain $R_{s,x+\tilde{a}}\setminus R_{s,x+a}$. Then $T((R_{x,x+\tilde{a}}\setminus R_{x,x+a})) = (R_{\tilde{x},\tilde{x}+\tilde{a}}\setminus R_{\tilde{x},\tilde{x}+a})$ Since a translation is a linear injective function $\tilde{z} = T(z_0) \in R_{\tilde{x},\tilde{x}+\tilde{a}}\setminus R_{\tilde{x},\tilde{x}+a}$ and $z_0 - x = \tilde{z} - \tilde{x}$.

Case 2b.

According to Corollary 20 we have for a fixed μ and $\lambda(t)$ that:

$$p_t\left(x,a,x_{t+1}\right)=p_t(x_{t+1}-x,i_a)=p\cdot P_t(x_{t+1}-x)+(1-p)\cdot\frac{P_t(i_a)}{\varkappa(x_{t+1}-x)}.$$

Then the inequality (7.6) can be rewritten as:

$$\sum_{x(t+1)\in R_{\tilde{x},\tilde{x}+\tilde{a}}}p_t\left(\tilde{x},x_{t+1},i_{\tilde{a}}\right)+\sum_{x(t+1)\in R_{x,x+a}}p_t\left(x,x_{t+1},i_a\right)$$
$$\geq\sum_{x_{t+1}\in R_{\tilde{x},\tilde{x}+a}}p_t\left(\tilde{x},x_{t+1},i_a\right)+\sum_{x(t+1)\in R_{x,x+\tilde{a}}}p_t\left(x,x_{t+1},i_{\tilde{a}}\right)$$

This equals further:

$$0\leq\sum_{x_{t+1}\in R_{\tilde{x},\tilde{x}+a}}\left[p_t\left(x_{t+1}-\tilde{x},i_{\tilde{a}}\right)-p_t\left(x_{t+1}-\tilde{x},i_a\right)\right]+\sum_{x_{t+1}\in R_{\tilde{x},\tilde{x}+\tilde{a}}\setminus R_{\tilde{x},\tilde{x}+a}}p_t\left(x_{t+1}-\tilde{x},i_{\tilde{a}}\right)+$$
$$+\sum_{x_{t+1}\in R_{x,x+a}}\left[p_t\left(x_{t+1}-x,i_a\right)-p_t\left(x_{t+1}-x,i_{\tilde{a}}\right)\right]-\sum_{x_{t+1}\in R_{x,x+\tilde{a}}\setminus R_{x,x+a}}p_t\left(x_{t+1}-x,i_{\tilde{a}}\right).$$

But $\sum_{x_{t+1}\in R_{\tilde{x},\tilde{x}+\tilde{a}}\setminus R_{\tilde{x},\tilde{x}+a}}p_t\left(x_{t+1}-\tilde{x},i_{\tilde{a}}\right)=\sum_{x(t+1)\in R_{x,x+\tilde{a}}\setminus R_{x,x+a}}p_t\left(x_{t+1}-x,i_{\tilde{a}}\right)$ and $\sum_{x_{t+1}\in R_{\tilde{x},\tilde{x}+a}}\left[p_t\left(x_{t+1}-\tilde{x},i_{\tilde{a}}\right)-p_t\left(x_{t+1}-\tilde{x},i_a\right)\right]=-\sum_{x_{t+1}\in R_{x,x+a}}\left[p_t\left(x_{t+1}-x,i_a\right)-p_t\left(x_{t+1}-x,i_{\tilde{a}}\right)\right]$.

Thus, the above inequality holds by proving the equality to zero.

Case 3. $x+\tilde{a}\notin K_s$

Then since $a\leq\tilde{a}$ we have that $x+a\notin K_s$, and consequently we have $R_{x,x+\tilde{a}}\cap K_s=R_{x,x+a}\cap K_s=\varnothing$, and the right-hand side of (7.6) is zero.

Then since $x\leq\tilde{x}$, both $\tilde{x},\ \tilde{x}+\tilde{a}$ may or may not belong to K_s, but the inequality (7.6) holds since $\sum_{x_{t+1}\in K_s\cap R_{\tilde{x},\tilde{x}+\tilde{a}}}p_t\left(\tilde{x},\tilde{a},x_{t+1}\right)-\sum_{x_{t+1}\in K_s\cap R_{\tilde{x},\tilde{x}+a}}p_t\left(\tilde{x},a,x_{t+1}\right)=$

$$=p\sum_{\substack{x_{t+1}\in K_s\\ x_{t+1}\in R_{\tilde{x},\tilde{x}+\tilde{a}}\setminus R_{\tilde{x},\tilde{x}+a}}}P_t\left(x_{t+1}-\tilde{x}\right)+(1-p)\sum_{\substack{x_{t+1}\in K_s\\ x_{t+1}\in R_{\tilde{x},\tilde{x}+\tilde{a}}\setminus R_{\tilde{x},\tilde{x}+a}}}\frac{P_t(i_{\tilde{a}})}{\varkappa(x_{t+1}-\tilde{x})}+(1-p)\sum_{\substack{x_{t+1}\in K_s\\ \tilde{x}\leq x_{t+1}\leq\tilde{x}+a}}\frac{[P_t(i_{\tilde{a}})-P_t(i_a)]}{\varkappa(x_{t+1}-\tilde{x})}$$

If the first two terms are clearly positive for the last one we have to use the special form of the $P_t(i_a)=\int_{1-\varepsilon/2}^{1+\varepsilon/2}f_{Y_{Erlang(i_{a_t}-1)}}(z)dz$.

Since $a\leq\tilde{a}$ we have also $i_a\leq i_{\tilde{a}}$ and $\frac{(\eta t)^{i_{\tilde{a}}-2}}{(i_{\tilde{a}}-2)!}\geq\frac{(\eta t)^{i_a-2}}{(i_a-2)!}$ since we can consider that $\eta t\geq i_{i_{\tilde{a}}}-2\geq 1$, for $t\in[1-\varepsilon/2,1+\varepsilon/2]$.

It is reasonable to have the scale factor of the solving time distribution larger than the number of concurrent design tasks.

Thus, we have an extension of the translation argument for the action-dependent sums as well (i.e. by translating upwards the points we obtain a higher probability instead of the same probability).

Case 4. $x\notin K_s$ but $x+\tilde{a}\in K_s$.

Because $a\leq\tilde{a}$ we have that $x+a$ may or may not belong to K_s.

Because $x\leq\tilde{x},\ \tilde{x}+\tilde{a}\in K_s$, but $\tilde{x},\tilde{x}+a$ may or may not belong to K_s.

As in Case 2 we have two subcases.

Case 4a. We consider that the two actions have the same number of zero components, or $i_{a_t}<2$. Similar to Case 2a. for each point in the intersection $R_{x,x+\tilde{a}}\setminus R_{x,x+a}\cap K_s$ we have a point of equal probability in the intersection $R_{\tilde{x},\tilde{x}+\tilde{a}}\setminus R_{\tilde{x},\tilde{x}+a}\cap K_s$, which gives us the desired equality.

Case 4b. Otherwise, we have to use the structure of the transition probabilities and we detail out the equation 7.6 as:

$$p \sum_{\substack{x_{t+1}\in K_s \\ x_{t+1}\in R_{\tilde{x}_t,\tilde{x}_t+\tilde{a}_t}\setminus R_{\tilde{x}_t,\tilde{x}_t+a_t}}} P_t\left(x_{t+1}-\tilde{x}\right)+(1-p)\sum_{\substack{x_{t+1}\in K_s \\ x_{t+1}\in R_{\tilde{x},\tilde{x}+\tilde{a}}\setminus R_{\tilde{x},\tilde{x}+a}}} \frac{P_t(i_{\tilde{a}})}{\varkappa(x_{t+1}-\tilde{x})}+(1-p)\sum_{\substack{x_{t+1}\in K_s \\ \tilde{x}\le x_{t+1}\le\tilde{x}+a}} \frac{[P_t(i_{\tilde{a}})-P_t(i_a)]}{\varkappa(x_{t+1}-\tilde{x})} \geq$$

$$p \sum_{\substack{x_{t+1}\in K_s \\ x_{t+1}\in R_{x_t,x_t+\tilde{a}_t}\setminus R_{x_t,x^t+a_t}}} P_t\left(x_{t+1}-x\right)+(1-p)\sum_{\substack{x_{t+1}\in K_s \\ x_{t+1}\in R_{x,x+\tilde{a}}\setminus R_{x,x+a}}} \frac{P_t(i_{\tilde{a}})}{\varkappa(x_{t+1}-x)}+(1-p)\sum_{\substack{x_{t+1}\in K_s \\ x\le x_{t+1}\le x+a}} \frac{[P_t(i_{\tilde{a}})-P_t(i_a)]}{\varkappa(x_{t+1}-x)}$$

Since K_s is an increasing set of $X(t+1)$ the set $R_{x,x+\tilde{a}}\setminus R_{x,x+a}\cap K_s$ contains less points than $R_{\tilde{x},\tilde{x}+\tilde{a}}\setminus R_{\tilde{x},\tilde{x}+a}\cap K_s$. Also, as we have shown in before for each point in the intersection $R_{x,x+\tilde{a}}\setminus R_{x,x+a}\cap K_s$ we have a point of equal probability $P_t\left(x_{t+1}-\tilde{x}\right)$ in the intersection $R_{\tilde{x},\tilde{x}+\tilde{a}}\setminus R_{\tilde{x},\tilde{x}+a}\cap K_s$ (i.e. using the translation argument). Consequently,

$$\sum_{\substack{x_{t+1}\in K_s \\ x_{t+1}\in R_{\tilde{x},\tilde{x}+\tilde{a}}\setminus R_{\tilde{x},\tilde{x}+a}}} \frac{P_t(x_{t+1}-\tilde{x})}{\varkappa(x_{t+1}-\tilde{x})} \geq \sum_{\substack{x_{t+1}\in K_s \\ x_{t+1}\in R_{x,x+\tilde{a}}\setminus R_{x,x+a}}} \frac{P_t(x_{t+1}-x)}{\varkappa(x_{t+1}-x)}.$$

The same reasoning leads to $\sum_{\substack{x_{t+1}\in K_s \\ x_{t+1}\in R_{\tilde{x},\tilde{x}+\tilde{a}}\setminus R_{\tilde{x},\tilde{x}+a}}} \frac{P_t(i_{\tilde{a}})}{\varkappa(x_{t+1}-\tilde{x})} \geq \sum_{\substack{x_{t+1}\in K_s \\ x_{t+1}\in R_{x,x+\tilde{a}}\setminus R_{x,x+a}}} \frac{P_t(i_{\tilde{a}})}{\varkappa(x_{t+1}-x)}$ and $\sum_{\substack{x_{t+1}\in K_s \\ \tilde{x}\le x_{t+1}\le\tilde{x}+a}} \frac{[P_t(i_{\tilde{a}})-P_t(i_a)]}{\varkappa(x_{t+1}-\tilde{x})} \geq \sum_{\substack{x_{t+1}\in K_s \\ x\le x_{t+1}\le x+a}} \frac{[P_t(i_{\tilde{a}})-P_t(i_a)]}{\varkappa(x_{t+1}-x)} \geq 0$. For the last inequality we also take into account that the terms of the sum are positive as shown in Case 3.

Finally, if we take an arbitrary K an increasing set in $X(t+1)$ from Corollary 55 there exist $s_1,...,s_q\in\mathbb{N}^N$ such that $K=\bigcup_{j=1}^{q} K_{s_j}$. Since the elements $s_1,...,s_q$ are not comparable one with the other the number of possible cases remains the same, as well as the proofs which are based on translation arguments. ■

Corollary 60 *For the Markov decision process constructed for the control of a NPD project without precedence constraints as described in Subsection 7.2.1 let us assume that during each review period $[t,t+1)$ we take into account at most a number K_t of unplanned design tasks which arrive with a probability greater than a given threshold $\vartheta(t)>0$ and $\sum_{t=0}^{T-1} K_t = N-\overline{N}$. Then the function defined by $\sum_{x_{t+1}\in K} p_t\left(x_t,a_t,x_{t+1}\right)$ is*

1. *nondecreasing in $x_t\in X(t)$, for any K increasing set in $X(t+1)$*
2. *supermodular/superadditive in $(x_t,a_t)\in X(t)\times A_t(x_t)$, for any K increasing set in $X(t+1)$.*

(with respect to the partial order on $X(t)\subseteq\{\mathbb{N}\cup\{-1\}\}^N$)

Proof. During each review period there are two independent processes that take place. The first is the solving process of the design activities from the design tasks performance levels decided by the action a_t. The second process that takes place is the Poisson arrival process of unplanned design tasks to be added to the NPD project description. Moreover, the number of unplanned design tasks taken into account at the end of the review period $[t,t+1)$ is limited to at most K_t, with $\frac{[\varsigma(t)]^{K_t}}{K_t!}e^{-\varsigma(t)}\geq\vartheta(t)$ (i.e. the length of each review period is 1).

Thus, according to Proposition 19 the transition probabilities of our Markov decision process are given by

$$p_t\left(x_t, a_t, x_{t+1}\right) =$$

$$= \begin{cases} p_t\left(\xi_{t+1} - x_t, i_{a_t}\right) \cdot \frac{[\varsigma(t)]^k}{k!} e^{-\varsigma(t)} & , x_t \leq x_{t+1} \leq x_t + a_t + z_t \text{ and} \\ & k \leq K_t \stackrel{def}{:=} card\{n | z_t(n) = 1\} \\ 0 & , \quad \text{otherwise} \end{cases}$$

where

$\xi_{t+1}(n) = \begin{cases} x_{t+1}(n) & , x_t(n) = l(n,t) \geq 0 \\ -1 & , \quad \text{otherwise} \end{cases}$, $\forall n \in \{1, \ldots, N\}$ (i.e. x_{t+1} restricted to the progress made on the previous existing design tasks),

i_{a_t} is the number of nonempty components in the action a_t, corresponding to concurrent design tasks (i.e. the number of concurrent design tasks given to the team of engineers), and

$$z_t(n) = \begin{cases} 0 & , x_t(n) = l(n,t) \geq 0 \\ 1 & , x_t(n) = -1, \forall n \in \{n_0, \ldots, n_0 + K_t\} \text{ and } x_t(n_0) \geq 0 \\ 0 & , \text{ otherwise} \end{cases} , \forall n \in \{1, \ldots, N\}$$

(i.e. counts the maximal number of unplanned design tasks that might be included in the project during the current review period).

The points in $X(t+1)$ for which we have $p_t\left(x_t, a_t, x_{t+1}\right) \neq 0$ belong to the set

$$R_{x_t, a_t + x_t} \stackrel{def}{=} \left\{ x \in \mathbb{N}^N | x_t \leq x \leq a_t + x_t + z_t \right\}.$$

Consider $x \leq \tilde{x}$ such that $x, \tilde{x} \in X\left(t\right)$. Let K an increasing set in $X\left(t+1\right)$. From Corollary 55 there exist $s_1, ..., s_q \in \{\mathbb{N} \cup \{-1\}\}^N$ such that $K = \bigcup_{j=1}^{q} K_{s_j}$. If now we are able to prove that for any elementary increasing subset K_s we have:

$\sum_{x_{t+1} \in K_s} p_t\left(x, a_t, x_{t+1}\right) \leq \sum_{x_{t+1} \in K_s} p_t\left(\tilde{x}, a_t, x_{t+1}\right)$, then due to the structure of K our corollary holds.

But

$$\sum_{x_t \leq x_{t+1} \leq x_t + a_t + z_t} p_t\left(x_t, a_t, x_{t+1}\right)$$

$$= \sum_{k=0}^{K_t} \left[\sum_{x_t \leq \xi_{t+1} \leq x_t + a_t} p_t(x_t, a_t, \xi_{t+1}) \right] \cdot \Pr\left(k \text{ arrived design tasks in } [t, t+1)\right)$$

Now, we consider two cases which allow us to re-write $\sum_{x_t \leq x_{t+1} \leq x_t + a_t + z_t} p_t\left(x_t, a_t, x_{t+1}\right)$ in a convenient form.

Case 1. $x_t \in K_s$ then we can write

$$\sum_{x_t \leq x_{t+1} \leq x_t + a_t + z_t} p_t\left(x_t, a_t, x_{t+1}\right) = \sum_{\substack{x_t \leq x_{t+1} \leq x_t + a_t + z_t \\ x_{t+1} \in K_s}} p_t\left(x_t, a_t, x_{t+1}\right).$$

Case 2. $x_t \notin K_s$ then the points in the bottom of the partially ordered set $\{x_{t+1} \in X(t+1) \cap K_s | \; p_t\left(x_t, a_t, x_{t+1}\right) \neq 0\}$ can be written as $x_{t+1} = \xi_{t+1}(n) + \overline{z}_t(n)$ where

$k_z := card\{n|\overline{z}_t(n) = 1\} \leq K_t$ and $x_t \leq \xi_{t+1} \leq x_t + a_t$, $\xi_{t+1}(n) \geq s(n)$ for any n such that $x_t(n) = l(n,t) \geq 0$. Thus, the sum becomes

$$\sum_{\substack{x_t \leq x_{t+1} \leq x_t + a_t + z_t \\ x_{t+1} \in K_s}} p_t\left(x_t, a_t, x_{t+1}\right) =$$

$$= \sum_{z \in B} \sum_{k=k_z}^{K_t} \left[\sum_{\substack{x_t \leq \xi_{t+1} \leq x_t + a_t \\ \xi_{t+1}(n) \geq s(n), \\ \forall n \text{ s.t. } x_t(n) = l(n,t) \geq 0}} p_t(x_t, a_t, \xi_{t+1}) \right] \Pr\left(k \text{ arrived design tasks in } [t, t+1)\right)$$

where $B = \text{Bot}\{x_{t+1} \in X(t+1) \cap K_s | p_t\left(x_t, a_t, x_{t+1}\right) \neq 0\}$. From Propositions 58 and 59 we have that $\sum_{x_t \leq \xi_{t+1} \leq x_t + a_t} p_t(x_t, a_t, \xi_{t+1})$ is nondecreasing in $x_t \in X\left(t\right)$ and supermodular/superadditive in $\left(x_t, a_t\right) \in X\left(t\right) \times A_t\left(x_t\right)$. Consequently, our corollary holds since these properties are preserved by the multiplication with a positive constant (i.e. in terms of $\left(x_t, a_t\right) \in X\left(t\right) \times A_t\left(x_t\right)$) and by summation of nondecreasing, respectively superadditive functions. ■

Theorem 61 *For the Markov decision process constructed for the control of a NPD project without precedence constraints as described in Subsection 7.2.1 there exist optimal decision rules which are nondecreasing in the state of the system if in the action space description (7.2) we consider the simplified work requirements restrictions (7.4). Otherwise (i.e. with the normal workload constraint given by (7.3)) there exist optimal decision rules which are weakly nondecreasing in the state of the system.*

Proof. For both Markov decision processes, there exists an optimal policy since the horizon, state and action space are finite, and the reward function is linearly additive. By summarizing the results of (White, 1980), (Puterman, 1994), (Topkis, 1998) given in Section 6 we have the following the general conditions under which for a discrete-time nonstationary Markov decision process, with finite horizon and the usual additive utility function.

With respect to componentwise partial order on the vector-lattices $\{\mathbb{N} \cup \{-1\}\}^N$ and respectively $\mathbb{N}^N$ we have to have for any $t \in \{0, 1, ...T-1\}$:

1. $\sum_{x_{t+1} \in K} p_t\left(x_t, a_t, x_{t+1}\right)$ is nondecreasing in $x_t \in X\left(t\right)$, for any K increasing set in $X\left(t+1\right)$
2. $\sum_{x_{t+1} \in K} p_t\left(x_t, a_t, x_{t+1}\right)$ is supermodular/superadditive in $\left(x_t, a_t\right) \in X\left(t\right) \times A_t\left(x_t\right)$, for any K increasing set in $X\left(t+1\right)$
3. (a) $\rho_t\left(\cdot, \cdot\right)$ is supermodular/superadditive in $\left(x_t, a_t\right) \in X\left(t\right) \times A_t\left(x_t\right)$
   (b) $\rho_t\left(\cdot, a_t\right) : X(t) \to \mathbb{R}$ is nondecreasing
   (c) $\rho_T(\cdot) : X(T) \to \mathbb{R}$ is nondecreasing
4. (a) $A_t\left(x_t\right) \subset A_t\left(\tilde{x}_t\right)$ for any $x_t \leq \tilde{x}_t \in X\left(t\right) \subset \{\mathbb{N} \cup \{-1\}\}^N$ (i.e. the family $\{A_t\left(x_t\right) | x_t \in X\left(t\right)\}$ is expanding)

(b) $S_t := X(t) \times A_t(x_t)$ is a lattice with respect to the componentwise partial order on the product space $\{\mathbb{N} \cup \{-1\}\}^N \times \mathbb{N}^N$.

All the requirements up to 4b are fulfilled due to Propositions 56, 57, and to Corollary 60 from above.

The state space of our Markov decision process constructed for the control of a NPD project without precedence constraints is only a sup-lattice, in the sense that for each pair of elements there exists only a least upper bound. However even if the Theorem 46 was formulated for a notation simplicity for the case in which S_t is a lattice, the proofs hold also for the case in which $X(t)$ is only a sup-lattice, while $A_t(x_t)$ is a lattice for any $x_t \in X(t)$. (see Theorem 43, and Proposition 44, as well as the proof of Proposition 48). Thus, the monotonic optimal policies can be obtained in a direct way if one enlarge the state space up to the vector-lattice $\{\mathbb{N} \cup \{-1\}\}^N$, and considers, in the action space description (7.2), the simplified maximal work requirement restriction (7.4), which leads to a lattice action space.

Otherwise (i.e. in the case of the normal workload constraint (7.3)), instead of a sublattice the action space is a bounded subset without holes. In this case we can use instead of the conditions from 4 the less restrictive set of requirements from Proposition 48 to prove the existence of optimal decision rules which are weakly nondecreasing in the state of the system. These conditions are:

1. $X(t), A_t(x_t)$, for any $x_t \in X(t)$ are bounded subsets without holes of the infinite vector-lattices $\{\mathbb{N} \cup \{-1\}\}^N$ and respectively $\mathbb{N}^N$
2. $A_t(x_t) \subset A_t(\tilde{x}_t)$ for any $x_t \leq \tilde{x}_t \in X(t)$, and $A_t(x_t) = A_t(\tilde{x}_t)$ for any $x_t, \tilde{x}_t \in X(t)$ not comparable (i.e. the family $\{A_t(x_t) | x_t \in X(t)\}$ is expanding)
3. for any $x_t \leq \tilde{x}_t \in X(t)$, and $a \leq \tilde{a} \in (A_t(x_t) \times A_t(\tilde{x}_t)) \cup (A_t(\tilde{x}_t) \times A_t(x_t))$ we have that $a \in A_t(x_t)$ and $\tilde{a} \in A_t(\tilde{x}_t)$ (i.e. the family is ascending)

These conditions are obviously fulfilled since we have from Proposition 57 that $A_t(x_t) = A_t(\tilde{x}_t)$, for any $x_t, \tilde{x}_t \in X(t)$. ■

There might exist other optimal policies which are not monotonic nondecreasing, or weakly monotonic nondecreasing.

7.3 Robust optimal nondecreasing policies for the control of NPD without precedence constraints

We recall from Subsection 6.5 that in general, not even the monotonicity of the decision rules does not always imply in higher dimensions a stable structure of the optimal paths (i.e. an increase in a exogenous parameter θ will lead to a uniform increase of the optimal path).

In this section we study how the optimal paths and optimal values are changing with respect to the important parameters of our NPD model without precedence constraints. We proved the robustness with respect to the variation of the safety margin for achieving the new product at the deadline in the case of both nondecreasing monotonic, and weakly nondecreasing monotonic optimal policies. We used simulation studies to investigate the robustness of the weakly monotonic optimal paths with respect to the variation of the solving

rate of design activities, as well as the variation of the optimal value as a function of the degree of specification of the characteristics of the new product at the beginning of the NPD.

7.3.1 Analytical results

If the action space is a lattice and the state space and parameter space form a complete sublattice under the product order then the Corollary 50 (consequence of Corollary 1 from (Friedman and Johnson, 1997)) states that the max and min selections of optimal actions will lead to non-chaotic optimal policies. As a direct consequence we have the following corollary.

Corollary 62 *For the Markov decision process constructed for the control of a NPD project without precedence constraints as described in Subsection 7.2.1 for which we consider the simplified work requirements restrictions (7.4), a decrease in the parameter β (i.e. an increase in $1/\beta$) leads to a uniform increase in the optimal decisions along any optimal sample path, where in every period the largest action is always chosen (with respect to the partial order on $\mathbb{N}^N, \{\mathbb{N} \cup \{-1\}\}^N$).*

Proof. We verify one by one the hypotheses of Corollary 50. Let $w_t(x_t, \beta, a_t) := \rho_t(x_t, \beta, a_t) + \sum_{x_{t+1} \in X(t+1)} p_t(x_t, \beta, a_t, x_{t+1}) \cdot u^*(x_{t+1})$ as given in a backward induction algorithm.

1.) $\max_{a_t \in A(x_t)} w_t(x_t, \beta, a_t)$ is attained for any $(x_t, \beta) \in X_t \times (0,1)$ since the decision process has finite action space and state space.

2.) Let $1/\beta \leq 1/\tilde{\beta}$ then $\beta \geq \tilde{\beta} \in (0,1)$ and with a smaller probability the team of engineers can perform more activities during the same review period. So for any $x_t \leq \tilde{x}_t \in X(t)$: $A_t(x_t, 1/\beta) \subset A_t(x_t, 1/\tilde{\beta}) = A_t\left(\tilde{x}_t, 1/\tilde{\beta}\right)$.

3.) Following the same reasoning as in Theorem 46 Subsection 6.3 we can prove the superadditivity of $w_t(x_t, \beta, a_t)$ by using that $u^*(x_t, \beta)$ is nondecreasing, and that the function $\sum_{x_{t+1} \in K} p_t(x_t, \beta, a_t, x_{t+1})$ is nondecreasing on $(x_t, \beta) \in X(t) \times \Theta$, and superadditive on $(x_t, \beta, a_t) \in X(t) \times \Theta \times A(x_t, \beta)$, for any K increasing set in $X(t+1)$.

According to Proposition 19 we have that the nonstationary transition probabilities $p_t(x_t, a_t, x_{t+1})$ do not depend on β. Thus the previous requirements hold true due to Proposition 44 and Proposition 59. To be in their hypotheses consider $x_t \leq \tilde{x}_t$ such that $x_t, \tilde{x}_t \in X(t)$, $a_t \leq \tilde{a}_t \in A(x_t, \beta)$. Now, for any $j \in X(t+1)$ we denote by $z_j := [p_t\left(\tilde{x}_t, \tilde{\beta}, \tilde{a}_t, j\right) + p_t(x_t, \beta, a_t, j)]$, $\tilde{z}_j := [p_t(x_t, \beta, \tilde{a}_t, j) + p_t\left(\tilde{x}_t, \tilde{\beta}, a_t, j\right)]$, and $v_j := u^*_{t+1}(j)$, in order to apply for them Corollary 42 . Its hypotheses are fulfilled since $X(t+1) \subseteq \mathbb{N}^N$, $\sum_{j \in K} z_j \geq \sum_{j \in K} \tilde{z}_j$, for any K increasing set in $X(t+1)$, and the sums are finite. Thus, we have $\sum_{j \in X(t+1)} [p_t\left(\tilde{x}_t, \tilde{\beta}, \tilde{a}_t, j\right) + p_t(x_t, \beta, a_t, j)] u^*_{t+1}(j) \geq \sum_{j \in X(t+1)} [p_t(x_t, \beta, \tilde{a}_t, j) + p_t\left(\tilde{x}_t, \tilde{\beta}, a_t, j\right)] u^*_{t+1}(j)$, which implies that $\sum_{x_{t+1} \in X(t+1)} p_t(x_t, \beta, a_t, x_{t+1})\, u^*_{t+1}(x_{t+1})$ is a superadditive function in $(x_t, \beta, a_t) \in X(t) \times \Theta \times A(x_t, \beta)$, for any $t \in \{0, 1, ...T-1\}$.

Since the sum of superadditive functions defined on the same domain remains superadditive $w_t(x_t,\beta,a_t)$ is superadditive in $(x_t,\beta,a_t)\in X(t)\times\Theta\times A(x_t,\beta)$. ■

If in the action space description (7.2) we do not consider the simplified maximal work requirement restriction (7.4) then there may be points that are unordered with respect to each other in $Top\ \arg\max_{a_{x_t}\in A_t(x_t)}\left\{\sum_{x_{t+1}\in X(t+1)} p_t(x_t,a_t,x_{t+1})\cdot u^*(x_{t+1})\right\}$. Thus, it can happen for some optimal decisions of the same review period to be unordered already, and thus the risk of chaotic behavior intuitively increases. However, using Theorem 51 we are also able to prove the robustness with respect to safety margin for achieving the new product at the deadline in the case of weakly nondecreasing monotonic optimal policies.

Theorem 63 *For the Markov decision process constructed for the control of a NPD project without precedence constraints as described in Subsection 7.2.1 for any two choices of the safety margin for achieving the new product at the deadline* $1/\beta\le 1/\tilde\beta\in(0,1)$*, at any decision point the optimal decision for* β *is either smaller or not comparable with the optimal action for* $\tilde\beta$*, along any optimal sample path, where in every period a maximal action is always chosen. (for any* $x^+\ge_X x^-$ *in* X*,* $1/\beta\le\tilde\beta$ *and any* $y^-\in \mathrm{Top}\arg\max_{y\in Y} g(x^-,\beta,y)$ *there exists* $y^+\in \mathrm{Top}\arg\max_{y\in Y} g(x^+,\tilde\beta,y)$ *such that* $y^+\ge_Y y^-$ *in* Y*, or there is no element in* $\mathrm{Top}\arg\max_{y\in Y} g(x^+,\tilde\beta,y)$ *comparable with* y^-*).*

Proof. In order to apply Theorem 51, Subsection 6.5 we have first to prove that $w_t(x_t,\beta,a_t):=\sum_{x_{t+1}\in X(t+1)} p_t((x_t,\beta),a_t,x_{t+1})\cdot u^*(x_{t+1},\beta)$ is superadditive on $(x_t,\beta,a_t)\in X(t)\times\Theta\times A(x_t,\beta)$, where we denote by $u^*(x_T,\beta)=\rho_T(x_T)$, and by $u^*(x_t,\beta):=\max_{a_t\in A(x_t)}\sum_{x_{t+1}\in X(t+1)} p_t((x_t,\beta),a_t,x_{t+1})\cdot u^*(x_{t+1},\beta)$.

Following the same reasoning as in Proposition 48 Subsection 6.3 we can prove the superadditivity of $w_t(x_t,\beta,a_t)$ by using that $u^*(x_t,\beta)$ is nondecreasing, and that the function $\sum_{x_{t+1}\in K} p_t(x_t,\beta,a_t,x_{t+1})$ is nondecreasing on $(x_t,\beta)\in X(t)\times\Theta$, and superadditive on $(x_t,\beta,a_t)\in X(t)\times\Theta\times A(x_t,\beta)$, for any K increasing set in $X(t+1)$.

According to Proposition 19 we have that the nonstationary transition probabilities $p_t(x_t,a_t,x_{t+1})$ do not depend on β. Thus the previous requirements hold true due to Proposition 44 and Proposition 59.

Now, for any $j\in X(t+1)$ we denote by $z_j:=[p_t\left(\tilde x_t,\tilde\beta,\tilde a_t,j\right)+p_t(x_t,\beta,a_t,j)]$, $\tilde z_j:=[p_t(x_t,\beta,\tilde a_t,j)+p_t\left(\tilde x_t,\tilde\beta,a_t,j\right)]$, and $v_j:=u^*_{t+1}(j)$, in order to apply for them Corollary 42 . Its hypotheses are fulfilled since $X(t+1)\subseteq\mathbb{N}^N$, $\sum_{j\in K} z_j\ge\sum_{j\in K}\tilde z_j$, for any K increasing set in $X(t+1)$, and the sums are finite. Thus, we have $\sum_{j\in X(t+1)}[p_t\left(\tilde x_t,\tilde\beta,\tilde a_t,j\right)+p_t(x_t,\beta,a_t,j)]u^*_{t+1}(j)\ge\sum_{j\in X(t+1)}[p_t(x_t,\beta,\tilde a_t,j)+p_t\left(\tilde x_t,\tilde\beta,a_t,j\right)]u^*_{t+1}(j)$.

This implies that $\sum_{x_{t+1}\in X(t+1)} p_t(x_t,\beta,a_t,x_{t+1})\,u^*_{t+1}(x_{t+1})$ is a superadditive function in $(x_t,\beta,a_t)\in X(t)\times\Theta\times A(x_t,\beta)$, for any $t\in\{0,1,...T-1\}$.

Since the sum of superadditive functions defined on the same domain remains superadditive $w_t(x_t,\beta,a_t)$ is superadditive in $(x_t,\beta,a_t)\in X(t)\times\Theta\times A(x_t,\beta)$.

We can now verify the other conditions of Theorem 51.

1.) $\max_{a_t \in A(x_t)} w_t(x_t, \beta, a_t)$ is attained for any $(x_t, \beta) \in X_t \times \Theta$ since the decision process has finite action space and state space.

2.) Let $1/\beta \leq 1/\tilde{\beta}$ then $\beta \geq \tilde{\beta} \in (0,1)$ and with a smaller probability the team of engineers can perform more activities during the same review period. So for any $x_t \in X(t)$: $A_t\left(x_t, 1/\beta\right) \subset A_t(x_t, 1/\tilde{\beta}) = A_t\left(\tilde{x}_t, 1/\tilde{\beta}\right)$, being by definition a bounded, without holes subset of $\mathbb{N}^N$, having as the same lower frontier as it. Consequently, for any $\tilde{a} \in A_t\left(\tilde{x}_t, 1/\tilde{\beta}\right)$ and $a \leq \tilde{a}$, a admissible action at the time instant t (i.e. $a > 0$) we have that $a \in A_t\left(\tilde{x}_t, 1/\tilde{\beta}\right)$.

3.) Let $x_t \leq \tilde{x}_t \in X(t)$, $1/\beta \leq 1/\tilde{\beta} \in (0,1)$, and $a \leq \tilde{a}$, $(a, \tilde{a}) \in A_t\left(\tilde{x}_t, 1/\tilde{\beta}\right) \times A_t\left(x_t, 1/\beta\right)$. Then $\tilde{a} \in A_t\left(x_t, 1/\beta\right) \subset A_t(x_t, 1/\tilde{\beta}) = A_t\left(\tilde{x}_t, 1/\tilde{\beta}\right)$ and $a \leq \tilde{a} \in A_t\left(x_t, 1/\beta\right)$ and by the definition of $A_t\left(x_t, 1/\beta\right)$ (i.e. a bounded, without holes subset of $\mathbb{N}^N$, having as the same lower frontier as it) $a \in A_t\left(x_t, 1/\beta\right)$. Thus, we have that $a \in A_t\left(x_t, 1/\beta\right)$ and $\tilde{a} \in A_t\left(\tilde{x}_t, 1/\tilde{\beta}\right)$ and the family is ascending.

The other requirements of Theorem 51 are fulfilled since for our Markov model $A_t\left(x_t, 1/\beta\right) = A_t\left(\tilde{x}_t, 1/\beta\right)$, for any $x_t, \tilde{x}_t \in X\left(t\right)$, and $\beta \in (0,1)$. ■

Corollary 64 *For the Markov decision process constructed for the control of a NPD project without precedence constraints as described in Subsection 7.2.1 a decrease in the parameter β (i.e. an increase in $1/\beta$) leads to weakly increasing optimal decisions along any optimal sample path, where in every period a maximal action is always chosen.*

7.3.2 Experimental results

We have used simulation studies for two main investigations: the optimal value variation function of the degree of specification of the characteristics of the new product at the beginning of the NPD, and the robustness of the weakly monotonic optimal paths with respect to the variation of the solving rate of design activities. We have considered that the characteristics of the new product are underspecified (respectively overspecified) if we have low (resp. high) values for the number of planned activities, and a high (resp. low) rate of arrival of unplanned activities during the design tasks solving process.

The experiments were performed using synthetic data. For each of the parameters whose influence we decided to study, we have established a set of values within a range. We have thus synthetically generated 20 data sets with design tasks of variable sizes for each $\mu \in \{8, 16, 24, 32\}$. We kept constant $\beta(t) = 0.85$, $M = 2$, $T = 5$, and $p = 0.90$.

For the arrival rate for new unplanned tasks during the first review period, i.e. $\lambda = \lambda(0)$, we have decided to compute its value for the first revision period as a fraction f_λ of μ, under the assumption that $\lambda(0)/\mu < M$. For the other revision periods, the $\lambda(t)$ value it is decreased with 10% per review period, the same during all the simulations.

For each design task the number of performance levels were generated independently by rounding up to the closest integer the random outcome of a uniform distribution inside the range $[10, 30]$. For each design task, the number of activities per performance level has been independently randomly chosen from the interval $[10, 100]$ in the following

way. Given the correlation mentioned in the first paragraph of this section, when the arrival rate of unplanned activities is high, the number of activities has to be low, and viceversa. Thus, we also introduced the proportion f_{act}, simultaneously varied with f_λ but in the opposite way: f_{act} took the percentage values $(150\%, 100\%, 50\%)$, while f_λ took the percentage values $(50\%, 100\%, 150\%)$. We have chosen these values in order to sample the space of possible values, when gradually going from underspecified characteristics of the new product to overspecified characteristics, having also a middle value.

We have also split the interval $[10, 100]$ in three disjoint and equal subintervals, corresponding to the f_{act} values. For each simulation having a certain value for f_{act} and its corresponding value for f_λ we independently randomly drawn values from the specific subinterval for the number of activities.

Then the optimal actions and values have been computed for each of the states, for each of the revision periods, according to the weakly-monotonic backwards-induction algorithm.

For the first part of the experimental study that we have performed, the goals were twofold: investigating the variation of the optimal value as a function of the specification degree of the characteristics of the new product, and comparing this variation for several values of the solving rate of the activities.

In the following, we give the tests results and we comment upon them. The graphs in Figures 7.5, 7.6, 7.7, and 7.8 give the optimal values as a function of the couple formed by f_{act} and f_λ (i.e. the percentages of the number of activities and respectively the percentage of lambda). Since they vary simultaneously, the x axis is labelled with the difference $100 - f_{\text{act}}$, the value of f_λ being easily obtained as $200 - f_{\text{act}}$. This gives for the x axis values $(-50, 0, 50)$, the values $(150, 100, 50)$ for f_{act} and the values $(50, 100, 150)$ for f_λ.

Thus we observe a rather surprising effect. Under the assumption that we have a correlation between the arrival rate of unplanned activities and the number of initially planned activities, it seems that independently of the value of μ, we have an increase of the optimal value with the increase of λ (and the decrease of the number of activities initially planned), i.e. if the product is less specified at the beginning.

However, this effect is less significant when the solving rate of one activity increases, i.e. the team of engineers is overall more performant.

For the second part of the experimental study, which concerned the robustness of the weakly monotonic optimal paths with respect to the variation of the solving rate of design activities, we did the following. We started with $\mu = 8$, and for each state x we have computed the set of optimal actions from $\text{Top} \arg\max_{y \in Y} g(x, \mu, y)$. Then we varied the value of μ, as $\tilde{\mu}$ from 16 to 32, we have computed, for each state x, how many actions among the ones of the set above satisfy one of the two following conditions, for each $y^- \in \text{Top} \arg\max_{y \in Y} g(x, 8, y)$:

- either there exists $y^+ \in \text{Top} \arg\max_{y \in Y} g(x, \tilde{\mu}, y)$ with $y^+ \geq y^-$
- or for all $y^+ \in \text{Top} \arg\max_{y \in Y} g(x, \tilde{\mu}, y)$, y^+ and y^- are not comparable.

If these percentages added up to 100%, then we can use Theorem 61 to say that an increase in the solving rate μ may lead to weakly increasing optimal actions along the optimal sample path. We experimentally see that in general these values are close to 100%

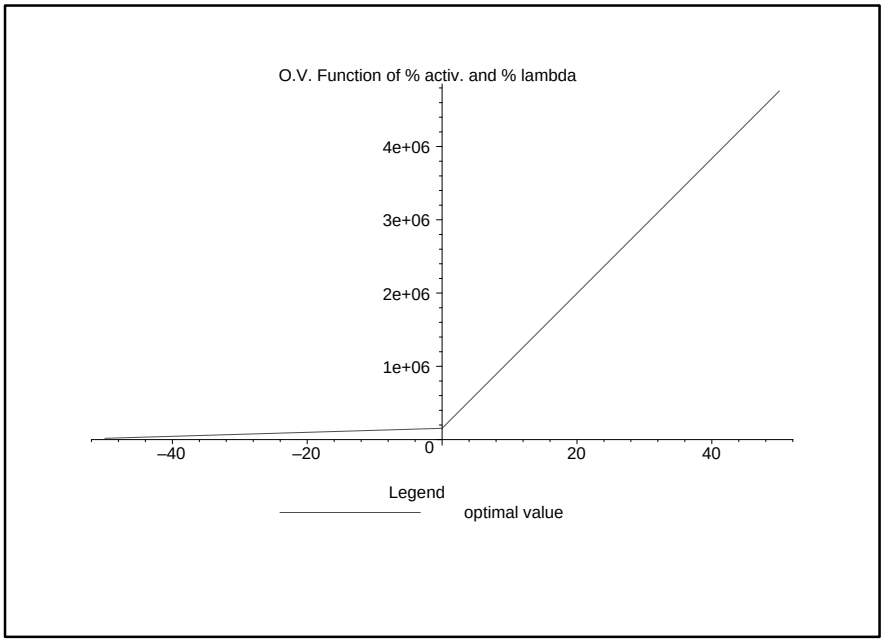

Figure 7.5. The optimal value variation as a function of f_{act} and of f_λ for $\mu = 8$

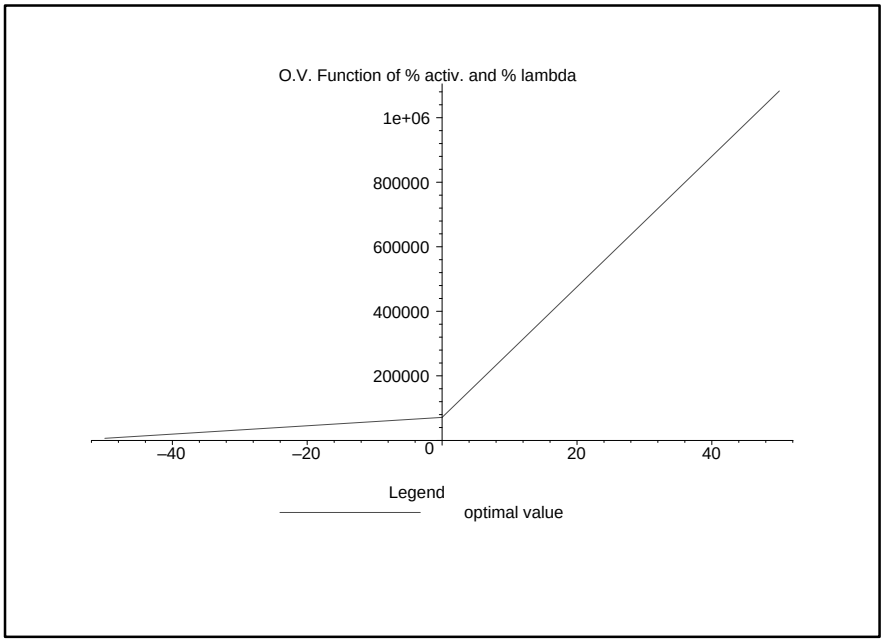

Figure 7.6. The optimal value variation as a function of f_{act} and of f_λ for $\mu = 16$

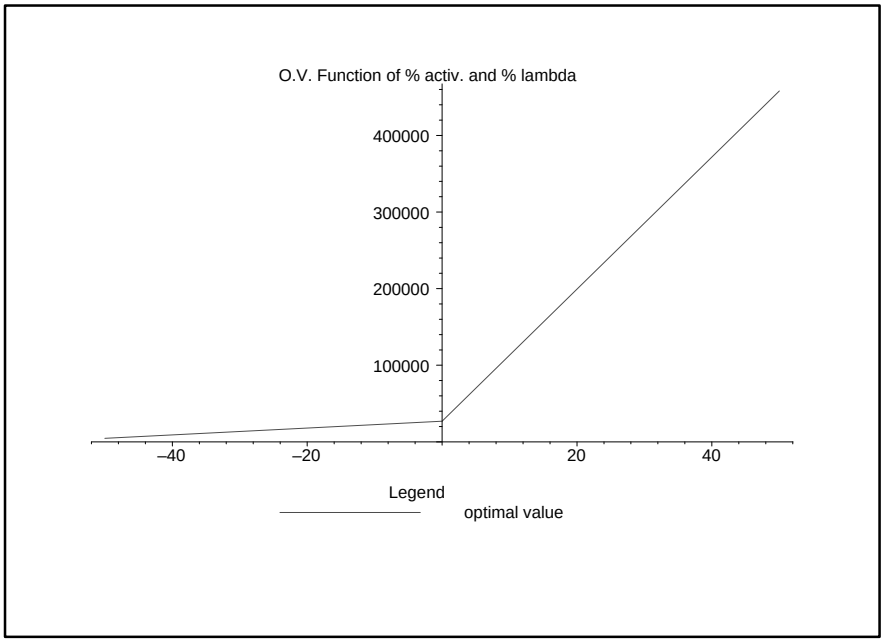

Figure 7.7. The optimal value variation as a function of f_{act} and of f_{λ} for $\mu = 24$

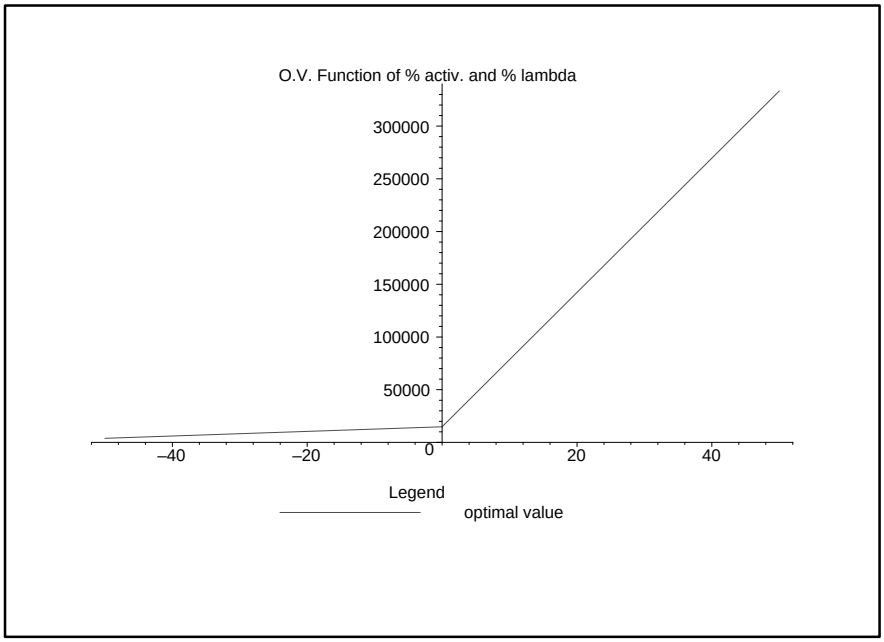

Figure 7.8. The optimal value variation as a function of f_{act} and of f_{λ} for $\mu = 32$

$(f_{\text{act}}, f_\lambda)$	$\tilde{\mu} = 16$ $\tilde{\mu} - \mu = 8$	$\tilde{\mu} = 24$ $\tilde{\mu} - \mu = 16$	$\tilde{\mu} = 32$ $\tilde{\mu} - \mu = 24$
$(150, 50)$	98%	96%	41%
$(100, 100)$	80%	96%	84%
$(50, 150)$	96%	99%	53%

Table 7.1. Table of sums of percentages of actions in the two cases for robustness study

for smaller values of $\tilde{\mu} - \mu$. On the other hand, from the cited theorem we have that given a μ, for any $x, \tilde{x} \in X$ with $x \leq \tilde{x}$, for all $y \in \text{Top} \arg\max_{x \in Y} g(x, \mu, z)$

- either there exists $\tilde{y} \in \text{Top} \arg\max_{z \in Y} g(\tilde{x}, \mu, z)$ such that $\tilde{y} \geq y$
- or for all $\tilde{y} \in \text{Top} \arg\max_{z \in Y} g(\tilde{x}, \mu, z)$, y and $\tilde{y}$ are not comparable.

Thus, using what the experimental data shows for x being the same and for two different values μ and $\tilde{\mu}$, we may combine them with the theorem, inferring that for any $x, \tilde{x} \in X$ with $x \leq \tilde{x}$, for any $\mu, \tilde{\mu}$ with $\mu \leq \tilde{\mu}$ and $\tilde{\mu} - \mu$ small, for all $y \in \text{Top} \arg\max_{x \in Y} g(x, \mu, z)$

- either there exists $\tilde{y} \in \text{Top} \arg\max_{z \in Y} g(\tilde{x}, \tilde{\mu}, z)$ such that $\tilde{y} \geq y$
- or for all $\tilde{y} \in \text{Top} \arg\max_{z \in Y} g(\tilde{x}, \tilde{\mu}, z)$, y and $\tilde{y}$ are not comparable.

This suggests that indeed a robustness result similar with the one in Corollary 64 may then held for small variations of the solving rate μ as well.

Moreover, during the experiments we have seen that their sum was having some variations for different values of μ. We then decided to investigate this in more detail, also as a function of the variation of the number of activities and of the variation of the rate of new activities arrival. These two parameters have previously been introduced as f_{act} and f_λ. We have used the sampling we already made for the first part of the experimental studies, and this yielded several values, summarized in Table 7.1, and detailed in Figures 7.9, 7.10 and 7.11.

The figures in Table 7.1 indicate that achieving the robustness of the optimal sample paths for larger variations of the solving rate ($\mu = 8$ versus $\tilde{\mu}$), might also depend on the degree of specification of the NPD project.

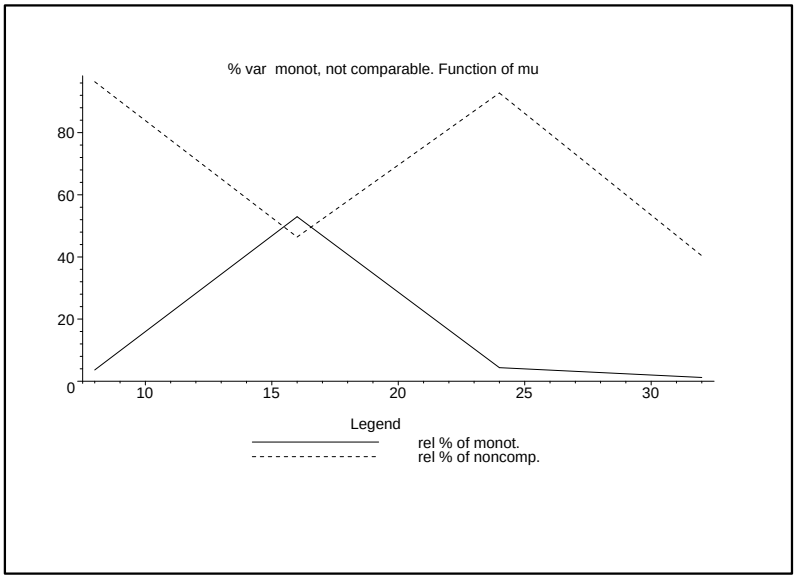

Figure 7.9. Proportion of actions satisfying each of the two conditions for $f_{\text{act}} = 150\%$ and $f_\lambda = 50\%$

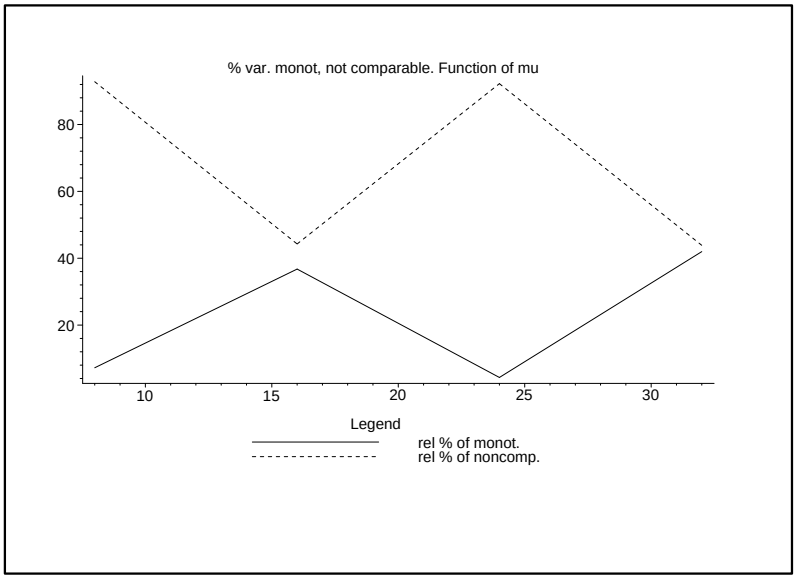

Figure 7.10. Proportion of actions satisfying each of the two conditions for $f_{\text{act}} = 100\%$ and $f_\lambda = 100\%$

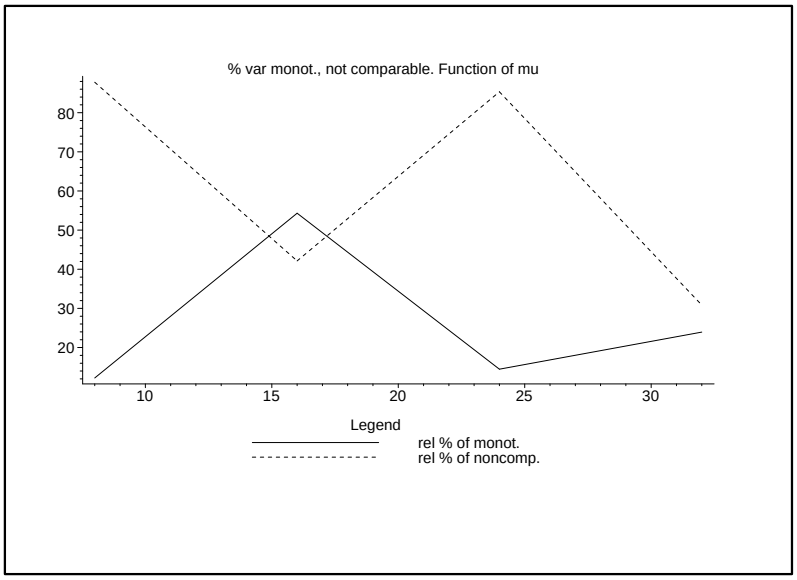

Figure 7.11. Proportion of actions satisfying each of the two conditions for $f_{\text{act}} = 50\%$ and $f_\lambda = 150\%$

Chapter 8

Markovian Control of Sequential NPD

8.1 Introduction

In this chapter we focus on a sequential NPD, i.e. a NPD project with precedence constraints, consisting of N sequential design tasks as described in Chapter 5 Subsection 5.4.2. To control it we use a discrete-time, finite horizon non-stationary Markov decision process.

A popular way of dealing with complex Markov decision processes is to investigate the structural properties of the optimal policies. The goal of such an approach is to describe in a relatively easy way the optimal policies and to translate their properties into practical procedures. The methodology used in this chapter is the one of the sample path analysis (see for a rigorous theoretical framework (Liu et al., 1995)). This technique aims at comparing sample path by sample path stochastic processes defined on a common probability space so that "characteristics of the optimal behavior" or even the "optimal behavior" can be identified. The proof techniques belong according to ((Liu et al., 1995)) to three classes: backward induction, forward induction and interchange of the arguments. We will use a mixture of the last two. The last one compares policies obtained through the interchange of control decisions and/or the order of some events in the input sequence. An important issue with respect to the sample path techniques is that not all the ways of describing the stochastic evolution of a system allow for a comparison between two given policies. Thus, as (Liu et al., 1995) the problem formulation should be carefully done. We carefully choose an appropriate state description in Chapter 5 Subsection 5.4.2. However this is not enough. To perform a sample path analysis we have to define in a very formal way the underlying probability space, the class of admissible actions at a certain time moment under a given history, and the reward function. Thus, in the rigorous framework of non-stationary models of (Hinderer, 1970) we identify our sequential NPD control problem as being a nonstationary stochastic dynamic decision model with stopping sets in the sense of (Hinderer, 1970) (see Section 8.3).

Due to an implicit probabilistic constraint on the sample paths of the process of sequentially completing design tasks through the completion of a minimal number of design activities, it can be shown that the optimal strategy belongs to a class of strategies deter-

mined by a set of "latest stopping times" for the design tasks. By restricting the action space, eliminating the sample paths with zero final reward, splitting the final reward into rewards to the transition moments, and finally by restricting simultaneously the number of decision moments, decision sets and the state space we obtain a sequence of four reformulations of the initial sequential NPD control problem. Using the sample path analysis we prove that the optimal policies of all those reformulations are optimal policies for the initial problem. The last reformulation reduces the initial multidimensional control problem to a unidimensional one in both state and action space. In the new control problem the optimal policy will decide only on how many review periods the team of engineers should work on each design task. We thus prove that it is optimal to choose always, while working on a design task, as decision the maximal performance level.

This chapter is organized as follows. In Section 8.2 we recall from Chapter 5 Subsection 5.4.2 the characteristics of the control problem of an NPD project with precedence constraints, consisting of N sequential design tasks. In Section 8.3, first we review the basic notions of non-stationary models from (Hinderer, 1970), and then we rigorously define the admissible histories of our sequential NPD control problem. In the following sections we construct the above mentioned reformulations of the initial sequential NPD control problem. In Section 8.4 we identify an implicit probability constraint which will help us to perform a first restriction of the action space of the initial problem. In Section 8.5 we eliminate from the control model the sample paths with zero reward, while in Section 8.6 we simplify the way of computing the expected value of the total reward, and we reduce the number of decision moments. Finally, in Section 8.7 we show there exists a sequence of sufficient statistics in the sense of (Hinderer, 1970), and (Dynkin, 1965) so that the solutions of a reduced unidimensional optimality equation are optimal solutions for our initial control problem.

8.2 The initial S_0- stochastic dynamic decision model

We recall from Chapter 5 Section 5.3 the assumptions and the notation related to this particular case of NPD project. We denote this initial stochastic dynamic decision model by S_0.

As in Subsection 5.4.2 we consider no budget constraint, and the team of engineers is assumed to work together on each of the design tasks. We focus on a NPD project with precedence constraints, consisting of $N \leq T$ sequential design tasks. We assume that the design tasks in the given sequence can be performed by the team of engineers one after the the other and they can start only at the beginning of a review period. The team may start working on first design task from the first review period. No arrival of unplanned design tasks takes place in the case of the sequential NPD project (i.e. $N = \overline{N}$).

An important supplementary *assumption concerning only this NPD case is related to the form of the final reward.*

Supplementary Assumptions:

1. The final reward is of the more restrictive form

$$\rho_T(x_T) = \begin{cases} 0, & \text{if } \exists\ 1 \leq n_0 \leq N, \quad l\,(n_0, T) < l_{\min}\,(n_0) \\ \sum\limits_{n\,=1,..,N} l\,(n, T) \cdot V(n) & \text{otherwise} \end{cases}$$

giving a linear (weighted additive) cumulated market payoff function in the arguments $V(n)$ which represent the scalable value functions associated with the design tasks. Such a function is similar with the one defined by (Askin and Dawson, 2000).

2. We do not consider anymore a time dependent safety margin for the probability of completing the project before the deadline $\alpha(t) = \alpha$, $t \in \{1, ...T-1\}$. However, our results can be generalized for a decreasing sequence of $\alpha(t)$, $t \in \{1, ...T-1\}$.

All the other assumptions and notation are given in Chapter 5, Section 5.3. The notation was influenced as well by the fact that we gave up to the budget constraint and by the assumptions we made in Chapter 5 Section 5.4.2. The first one was that no arrival of new design tasks takes place in the case of the sequential NPD project. The second one was that assuming the previous levels already solved there is an identical number of activities for any performance level of the same design task. Thus, assuming the previous levels already solved, the solving times of all the performance levels of the same design task will be i.i.d. random variables.

Input parameters (global variables) **:**

T : the total number of review periods (review periods are numbered from 0 to $T-1$);

M : the total number of engineers;

$\overline{N} = N \leq T$: the total number of design tasks;

$L_{\max}(n)$: the maximal number of performance levels of the design task n; $n = 1, \ldots, N$ (levels are numbered from 0 to L;

$l_{\min}(n)$: the minimal performance level at which the design task n must be performed in order to obtain a functional product; $n = 1, \ldots, N$;

$n_a(n)$: the number of sequential activities planned for solving the design task n, at the performance level l, assuming the previous levels already solved; $\forall\ n = 1, \ldots, N$; $l = 1, \ldots, L_{\max}(n)$;

μ : the rate of the exponential distribution of an activity solving time.

α : the required safety margin for the probability of completing the project before the deadline; $\alpha \in (0, 1)$

Input parameters (at the beginning of review period t)**:**

$l(\cdot, t) : \{1, ..N\} \to \mathbb{N}$: the achieved performance design task level function, where $0 \leq l(n, t) \leq L_{\max}(n)$;

$\lambda(t)$: the review-period dependent Poisson arrival of unplanned activities for all the design tasks allocated to the engineers.

Notation (at the beginning of review period t):

S_n : the solving time of an arbitrary performance level of the design task n, assuming the previous levels already solved.; $n = 1, ..., N$. They are independent random variables Erlang-$(n_a(n), M\mu)$.

The decision time points of S_0: The decision points are equidistant and the horizon of the problem is finite. The decision point t corresponds to the beginning of review $t+1$. Say $t \in \{0, 1, ...T-1\}$.

The state space and the action space of S_0: The state set $X(t)$ at moment t and the action set $A_t(x_t)$ in the state $x_t \in X(t)$ are defined by two main constraints:

- the target performance level of each planned design task n is smaller than $L_{\max}(n)$ and greater than $\min(l_{\min}(n), l(n, t))$

- the completion time of the NPD project must be smaller than the remaining time until the deadline with a probability greater than safety margin $\alpha(t)$.

For the state space the completion time constraint is computed using the minimal performance levels, $l_{\min}(\cdot,t)$, while for the action space it is used the target performance levels, $l(\cdot,t)+a(\cdot,t)$.

For $t=0: X(0)=\{x_0\}=\{0_{\mathbb{R}^N}\}$

For $t\in\{1,...T-1\}$ the state $x(t)\in X(t)$ indicates $n(t)$ as the current not yet finished design task from the initial sequence, and describes how many performance levels were solved for it, as well as for the finished design tasks.

$$X(t)=\left\{x_t\left|\begin{array}{c} x_t=(l(1,t),...,l(n(t),t),0,...,0)\in\mathbb{N}^N, n(t)\in\{1,\ldots,N\} \\ 0<l\left(i,t\right)\leq L_{\max}(i), i\in\{1,...,n(t)\} \\ \Pr\left\{\sum\limits_{n=n(t)+1}^{N}\sum\limits_{i=1}^{l_{\min}(n)}S_n+\sum\limits_{i=l(n,t)+1}^{l_{\min}(n)}S_{n(t)}\leq M(T-t)\right\}\geq\alpha \end{array}\right.\right\} \tag{8.1}$$

For $t=T$:

$$X(T)=\left\{x_T\left|\begin{array}{c} x_T=(l(1,T),...,l(N,T))\in\mathbb{N}^N, \\ l_{\min}(n)\leq l\left(n,T\right)\leq L_{\max}(n,T), n\in\{1,...,N)\} \end{array}\right.\right\}$$

For $t\in\{0,...T-1\}$ and $x_t=(l(1,t),...,l(n(t),t),0,...,0)\in\mathbb{N}^N, n(t)\in\{1,\ldots,N\}$ the action a_t decides how many other levels above x_t we want to perform. The level up to which the design task n may be performed after the action a_t was taken is: $l(n,t)+a_t(n)$.

$$A_t(x_t)=\left\{a_t\left|\begin{array}{c} a_t=(0,..,0,a_t(n\left(t\right)),...,a_t(N))\in\mathbb{N}^N \\ 0\leq l_{\min}(n)-l(n,t)\leq a_t(n),\forall n\in\{1,...,N\} \\ l\left(n,t\right)+a_t(n)\leq L_{\max}(n,t) \\ \Pr\left\{\sum\limits_{n=n(t)+1}^{N}\sum\limits_{l=1}^{a_t(n)}S_n+\sum\limits_{l=1}^{a_t(n(t))}S_{n(t)}\leq M(T-t)\right\}\geq\alpha \end{array}\right.\right\} \tag{8.2}$$

The immediate rewards of S_0: $\forall\, x_t\in X(t)$, $x_{t+1}\in X\left(t+1\right)$ and $a\in A\left(x_t\right)$, the immediate reward is

$$\rho_t\left(x_t,a,x_{t+1}\right)=0,\forall t=0,...,T-1 \tag{8.3}$$

and the final reward is according to the supplementary assumption concerning the sequential NPD case

$$\rho_T(x_T)=\left\{\begin{array}{cc} 0, & \text{if } \exists\ 1\leq n_0\leq N,\quad l\left(n_0,T\right)<l_{\min}\left(n_0\right) \\ \sum\limits_{n\,=1,..,N}l\left(n,T\right)\cdot V(n) & \text{otherwise} \end{array}\right. \tag{8.4}$$

If we consider a final reward of the form given in (Askin and Dawson, 2000) then as pointed out in Chapter 5 Subsection 5.4.2, a simple choice for the value per a performance level for the design task n is: $V\left(n\right):=\frac{1}{L_{\max}(n)}\cdot\sum\limits_{\delta=1}^{\Delta}w_\delta\cdot\Theta\left(n,\delta\right),\forall\, n=1,...,n$, where w_δ is the normalized weight of the customer need δ ($\forall\,\delta=1,...,\Delta$), and $\Theta\left(n,\delta\right)$ is the normalized maximal contribution of the design task n in fulfilling the customer need δ ($\Theta\left(n,\delta\right)\in[0,1)$; $\forall\, n=1,...,N,\forall\,\delta=1,...,\Delta$; $\sum\limits_{\delta=1}^{\Delta}\Theta\left(n,\delta\right)=1$).

The transition probabilities of S_0: The transitions probabilities depend only on the decision time point, the observed state and the chosen action and not on the whole history of the process. The probability that the next state is x_{t+1}, given that the state at the beginning of stage t is x_t and that the action $a_t \in A_t(x_t)$ is chosen, is the nonstationary probability: $p_t(x_t, a_t, x_{t+1})$ given in Corollary 20, Chapter 5.

Expected total reward criterion: The total expected reward) of a policy π, starting from initial state $x_0 = 0_{\mathbb{R}^N}$ is $v_0^\pi(x_0) = E_{x_0}^\pi \left[\sum_{t=0}^{T-1} \rho_t(x_t, a) + \rho_T(x_T) \right]$, where the expected reward during the time interval $[t, t+1)$ is:

$$\rho_t(x_t, a) = \sum_{x(t+1)} \rho_t(x_t, a, x_{t+1}) \, p_t(x_t, a, x_{t+1}) :\stackrel{def}{=} 0.$$

We are looking for maximal expected total reward.

8.3 Preliminaries

In this section we recall from (Hinderer, 1970) a series of definitions and properties of non-stationary dynamic programming models with discrete time parameter, and we present the relationships with the initial S_0- stochastic dynamic decision model. Due to the particular form of our sequential control problem, we focus on the implications of Hinderer's results for the models with finite states and actions, with bounded rewards, and finite horizon.

Definition 65 *The mathematical framework for a stochastic dynamic decision model with countable state space consists of a tuple $(X, A, D, p, (p_t)_{t\in\{0,1,2...\}}, (r_t)_{t\in\{0,1,2...\}})$ of objects of the following meaning:*

i) X, the so called state space, is a non-empty countable set,

ii) A, the so called space of actions, is either a non-empty countable or some non-empty Borel subset of R^n. The set $\overline{H}_t = (X \times A) \times (X \times A) \times ... \times X$ ($2t+1$ factors) is called the set of histories $h_t = (x_0, a_0, x_1, a_1, ..., x_t)$ at time t.

iii) D is a sequence of maps D_t, $t \in \{0, 1, 2...\}$ from certain sets $H_t \subseteq \overline{H}_t$ to the set of all non-empty subsets of A with the property that $H_0 = X$, $H_{t+1} = \{(h, a, x) \mid h \in H_t, a \in D_t(h), x \in X\}$, $t \in \{0, 1, ...\}$.

$D_t(h)$ is called the set of admissible actions at time t under history h, whereas H_t is called the set of admissible histories at time t. By K_t is denoted the set $\{(h, a) \mid h \in H_t, a \in D_t(h)\}$.

iv) p called the initial distribution, is a counting density on X, and $p_t(h, a, \cdot)$ called the transition law between time t and $t+1$, $t \in \{0, 1, ...\}$ is a counting density on X, $(h, a) \in K_t$.

v) r_t, the so called reward during the time interval $[t, t+1)$, $t \in \{0, 1, ...\}$ is an extended real valued function on K_t.

Any tuple $(X, A, D, p, (p_t)_{t\in\{0,1,2...\}}, (r_t)_{t\in\{0,1,2...\}})$ with the property (i)-(v) is called a stochastic dynamic decision model (without stopping sets).

Definition 66 *A (deterministic) policy for a stochastic dynamic decision model is a sequence $\pi = \{\pi_t\}_{t\in\{0,1,...\}}$ of maps $\pi_t : X \times ... \times X$ ($t+1$ factors) $\rightarrow A$ with the property $\pi_t(x_0, x_1, ..., x_t) \in D_t(h_{t\pi}(x_0, x_1, ..., x_t))$, where we denote the history at time t obtained*

by use of policy π when the sequence $x_0, x_1, \ldots, x_t$ of states (sample path or trajectory) occurred by $h_{t\pi}(x_0, x_1, \ldots, x_t) = \{x_0, \pi_0(x_0), \ldots, x_{t-1}, \pi_{t-1}(x_0, \ldots, x_{t-1}), x_t\}$.

Definition 67 *Models with finite horizon $T \in \mathbf{N}$ are models where $r_t = 0$ for all $t > T$.*

Definition 68 *The mathematical model for a finite horizon decision process determined by a policy π (see (Hinderer, 1970))*

is the probability space $(\Omega, \digamma, P_\pi)$ and a sequence $(\zeta_t)_{t\in\{0,..,T\}}$ of random variable on it where $(\zeta_t)_t$ describes the state of the system at time t. We have:

1) as sample space $\Omega = X^{T+1}$ the set of all sequences $\omega = (x_0, ..., x_T)$,

2) as $\sigma-$ algebra $\digamma$ on Ω is considered the finite product($T+1$ factors) of $\sigma-$ algebra determined by the factors $\mathcal{P}(X)$ (the system of all subsets of X),

3) ζ_t is the $t-$ th coordinate variable, i.e. $\zeta_t(\omega) = \zeta_t(x_0, ..., x_T) = x_t$, $t \in \{0, 1, .., T\}$

4) for the description of P_π is used the notation $p_{t\pi}(y, x) = p_t(h_{t\pi}(y), \pi_t(y), x)$ where $y = (x_0, ..., x_t) \in X^{t+1}$ and $x \in X$, i.e. $p_{t\pi}$ is the transition law between time t and $t+1$ that results from the application of the policy π.

Let η_t the random vector $(\zeta_0, .., \zeta_t)$ describing the history at time t. We have $P_\pi(\eta_t = y) = P_\pi((\zeta_0, .., \zeta_t) = (x_0, .., x_t))$ denoted by the following $P_\pi(x_0, .., x_t) = p(x_0)p_{0\pi}(x_0, x_1)...p_{t-1,\pi}(x_0, x_1, ..., x_t)$.

If the policy π is used and if the sequence $(x_0, .., x_t)$ of states (sample path or trajectory in between the moments 0 and t) has occurred, then the reward received during the time period $[t, t+1)$, $t \in \{0, .., T-1\}$ is $r_{t\pi}(y) = r_t(h_{t\pi}(y), \pi_t(y))$ and if the sequence $(x_0, .., x_T)$ of states (sample path or trajectory) has occurred, the final reward $(t = T)$ is $r_{T\pi}(y) = r_T(h_{T\pi}(y))$.

If the policy π is used and if the sample path $(x_0, .., x_T)$ has occurred, then the total reward received during the time $[0, T]$ is $R_\pi(x_0, .., x_T) = \sum_{t=0}^{T-1} r_t(h_{t\pi}(y), \pi_t(y)) + r_T(h_{t\pi}(y))$. If the policy π is used then the expected total reward (or the expectation of the total reward) is $\sum_{(x_0,..,x_T)} R_\pi(x_0, .., x_T)P_\pi(x_0, .., x_T) = \sum_{(x_0,..,x_T)} r_{t\pi}(\eta_t(x_0, .., x_T))P_\pi(x_0, .., x_T)$.

Remark 69 *The application of any policy π generates a stochastic process, the decision process determined by π. A stochastic dynamic decision model may be regarded as a family of decision processes.*

We are looking for the maximal expected total reward. The existence of a deterministic optimal policy is insured for a stochastic dynamic decision process with countable states and actions, with bounded rewards, and infinite horizon. Thus, the same holds in the particular case of a finite horizon. (see (Hinderer, 1970) pp. 7-11).

Remark 70 *In (Hinderer, 1970) (pp.12) there are presented two equivalent ways (i.e. they lead to the same reward under the maximal expected total reward) to reduce a model in which the reward between time t and $t+1$ depends also on x_{t+1} (i.e. $r'_t(h, a, x_{t+1})$) to the case of the stochastic dynamic decision model from Definition 65. The first way to reduce this case to the above model is to use the reward $r_t(h, a) = \sum_{x_{t+1}\in X} r'_t(h, a, x_{t+1})\, p_t(h, a, x_{t+1})$ as new reward function. However, in this chapter we will mostly use the second one, which consists in using $r_0 \equiv 0$ and $r_t(h, a) = r'_{t-1}(h)$.*

Remark 71 *Since, we deal with models with finite horizon T, we have $r_t = 0$ for all $t > T$. Thus, in the finite horizon case, if the reward between time t and $t+1$ depends also on x_{t+1} (i.e. $r'_t(h, a, x_{t+1})$), and we use the second way of reducing the model to the one from Definition 65 we have to take care not to forget the reward $r'_{T-1}(h, a, x_T)$. If the horizon would have been infinite this reward would have been the one of the period $[T, T+1)$. Thus, in this chapter, we will introduce a final reward, which will correspond to the moment T, and to the admissible history $(h, a, x_T) \in H_T$. An alternative possibility would have been to extend the model horizon with one unit.*

In general, for a stochastic dynamic decision model, the set of states at any decision moment is considered to be equal to X. For our S_0 model there exists a first restriction: at the initial decision moment we can consider only the state $x_0 = 0_{R^N}$ which leads to a first restriction on the set of admissible histories.

Such a situation occurs in most practical problems (see (Hinderer, 1970) pp.7). The existence of a fixed state x_0 in which the system starts with probability one ensures the existence of an optimal policy in the case of countable state and action spaces. This policy is called x_0- optimal.

The second restriction of our S_0 model is that at different decision moments we have different state spaces and different action spaces. According to (Dynkin, 1965), (Hinderer, 1970) pp.11, (Hinderer, 1967), and (White, 1969) pp.23 the fact that at the times $t + 0, 1, 2, ...$ different state spaces and/or different action spaces are needed would not complicate the theory. Nevertheless, the notation becomes a bit cumbersome.

Remark 72 *Thus, coming back to our S_0 model we consider*

$$X = \left\{(l_1, ..., l_i, 0, ..., 0) \in \mathbb{Z}^N, i \in \{1, \ldots, N\}, 0 < l_j \leq L_{\max}(j), j \in \{1, ..., i)\}\right\} \cup \{0_{\mathbb{Z}^N}\}$$

and we define alternatively the set of admissible histories. We start with $H_0 = \overline{X}(0) = \{x_0 \mid p(x_0) > 0\}$.

We denote by $A_0(h_0)$ the set of admissible actions under the history h_0 (it will be in fact an $A_0(x_0)$) and we define the set of admissible actions at the time instant 0 as being $A(0) = \bigcup_{h_0 \in H_0} A_0(h_0)$.

For $(h_0, a_0) \in H_0 \times A(0)$ we define the set $X(1, ((h_0, a_0)) = \{x_1 \in X \mid p_1(h_0, a_0, x_1) > 0\}$, and the set of possible states at time instant 1 is given by $\overline{X}(1) = \bigcup_{(h_0, a_0) \in H_0 \times A(0)} X(1, ((h_0, a_0))$, $H_1 = \overline{X}(0) \times A(0) \times \overline{X}(1)$.

In general, if we inductively determine H_t we may define $A(t) = \bigcup_{h_t \in H_t} A_t(h_t)$ (the history dependence in $A_t(h_t)$ is only given by the means of x_t), $X(t+1, ((h_t, a_t)) = \{x_{t+1} \in X \mid p_t(h_t, a_t, x_{t+1}) > 0\}$, where $(h_t, a_t) \in H_t \times A(t)$, and also $\overline{X}(t+1) = \bigcup_{(h_t, a_t) \in H_t \times A(t)} X(t+1, ((h_t, a_t))$, $H_{t+1} = \overline{X}(0) \times A(0) \times \cdots \times \overline{X}(t) \times A(t) \times \overline{X}(t+1)$.

From now on, we will use this construction for the admissible histories of the stochastic dynamic decision model S_0.

Remark 73 *The Definition 68 of a decision process determined by a policy can be easily adapted for a model defined by the admissible state sets $X(t),\ t \in \{0, .., T\}$ by replacing X^{T+1} with $X(0) \times ... \times X(T)$.*

We notice that at any decision moment t in the S_0 model we eliminated some states due to the condition $\Pr\left\{\sum_{n=n(t)+1}^{N}\sum_{i=1}^{l_{\min}(n)} S_n + \sum_{i=l(n,t)+1}^{l_{\min}(n)} S_{n(t)} \leq M(T-t)\right\} \geq \alpha$. Only after this elimination we obtain $X(t)$ the set of admissible states at the decision moment t. Not every state which can be reached from $X(t-1)$ with a positive probability p_{t-1} is an element of the set $X(t)$. Let $\overline{X}(t)$ (we call $\overline{X}(t)$ the set of possible states) the set of states which can be reached from $X(t-1)$ with a positive probability p_{t-1}. $\overline{X}(t) = \left\{ x_t \left| \begin{array}{c} x_{t-1} \leq x_t \leq x_{t-1} + a_{t-1}, x_t \in \mathbb{N}^N \\ x_t = (l(1,t), ..., l(n(t),t), 0, ..., 0) \in \mathbb{N}^N, n(t) \in \{1, \ldots, N\} \\ 0 < l(i,t) \leq L_{\max}(i), i \in \{1, ..., n(t)\} \\ \text{and } \Pr\left\{\sum_{n=n(t)+1}^{N}\sum_{i=1}^{l_{\min}(n)} S_n + \sum_{i=l(n(t),t)+1}^{l_{\min}(n(t))} S_{n(t)} \leq M(T-t)\right\} \geq \alpha \\ a_t = (0, .., 0, a_t(n(t)), ..., a_t(N)) \in \mathbb{N}^N \\ a_t(n) \geq l_{\min}(n) - l(n,t) \geq 0, \forall n \in \{1, ..., N\} \\ l(n,t) + a_t(n) \leq L_{\max}(n) \\ \Pr\left\{\sum_{n=n(t)+1}^{N}\sum_{l=1}^{a_t(n)} S_n + \sum_{l=1}^{a_t(n(t))} S_{n(t)} \leq M(T-t)\right\} \geq \alpha \end{array} \right. \right\}$

At any decision moment t are neglected the states from the set $\overline{X}(t) - X(t)$.

We also observe that a state which is not admissible at the decision moment t will not be admissible at any later decision moment. From a mathematical point of view this is equivalent with saying that $\overline{X}(t) - X(t)$ is a set of states in which we stop the process at the decision moment t.

Definition 74 *(Hinderer, 1970) Let $(X, A, D, \{p_t\}_{t=0,1,...}, \{r_t\}_{t=0,1,...})$ be a sequential dynamic decision process having $\{H_t\}_{t=0,1,...}$ as sets of the admissible histories. This decision process is said to be with stopping sets if there exists the sets (possibly empty) $\Phi_t \subset H_t$, together with the instruction to stop the process at the earliest time t for which h_t belongs to the set $\Phi_t \subset H_t$.*

(One could allow Φ_t to depend on (h_{t-1}, a_t); but if we define $\overline{\Phi}_t := \{h_t \in H_t : h_t \in \Phi_t(h_{t-1}, a_t)\}$, then we are back in our original case.)

As a example we quote the task to proceed to a given point $c \in S$ as rapidly as possible ((Boudarel et al., 1968), pp. 23, (Hinderer, 1970) pp.57).

Remark 75 *S_0 is a particular stochastic dynamic decision model with stopping sets. The sets $X(t)$ represent the sets of states in which the process is not stopped. Later, we will fully define the stopping sets for the S_1 model which will be constructed starting from the S_0 model. This will help us to obtain a series of properties of the S_1 model as a process with stopping sets.*

Remark 76 *The initial process has as admissible states the states from $X(t)$. It is not the initial process that we intend to formulate as a process in the sense of Hinderer. The idea is to show that the initial process is equivalent with a stopped process that can be obtained from the process with stopping sets in the sense Hinderer. The process in the sense of Hinderer has as sets of states $\bar{X}(t)$, and as the set of admissible histories the $H_t = \bar{X}(0) \times A(0) \times ... \times \bar{X}(t-1) \times A(t-1) \times \overline{X}(t)$*

In our S_0 model the reward is final. Thus, we can assume without any loss of generality that if the process enters in a possible, but not admissible state (i.e. a state in which the process is stopped) at the t decision moment, it will remain in that state until the T moment. This is equivalent with saying that any decision moment $t, \ldots, T-1$ in the above mentioned states we take the action 0_{R^N}, which does not allow the process to further evolve. With this assumption we can extend the set of states (from $\overline{X}(t)$ to X), as well as the action states at any decision moment t (with the action 0_{R^N} for all the possible, but not admissible states, no matter the decision moment at which they did became not admissible).

We remark that in our S_0 model for the final state $x_T = (l(1,T),..,l(N,T))$ we have $l(n,T) = l(n,t_n)$, where t_n is the moment in which we stop working on the $n-$th design task. Thus, if $l(n,T) \geq l_{\min}(n)$, for any $n \in \{1,\ldots,N\}$ the total expected final reward corresponding to x_T may be considered as a sum of rewards $\sum\sum_{n} V(n)\, l(n,t_n)$, where the first sum is taken after all the trajectories that end up into $x_T = (l(1,T),..,l(N,T))$.

Moreover, the level $l(n,t_n)$ may be considered as a sum of the levels performed in between the decision moments t_{n-1} and t_n. In this way the final reward becomes the sum of all rewards corresponding to the periods in between the decision moments t_{n-1} and t_n. If we assume that a possible, but not admissible state at the decision moment t remains unchanged until T, and if we think of the final reward as a sum of rewards per period, we have to consider not only that no reward will be accumulated from t to T, but also that the reward possibly obtained from the work of the engineers team up to the $t-$ th moment has to be subtracted. Only then the final reward of a non admissible state will be zero.

8.4 Restricting the action space of S_0

In this section we identify an implicit probability constraint which lead to a reformulation of the action space of the S_0 model. We denote the new model by S_1. First we determine a sequence of time moments $T_1,..,T_{N-1}$. A T_n moment is that moment in time after which we decide not to work anymore on the design task number n. Doing otherwise will prevent the team to finish the minimal performance levels for the remaining $(n+1,\ldots N)$ design tasks (i.e. achieving a functional new product) until the deadline, with the required probability. In the S_1 model the actions will have at most one nonzero component which will be decided at the T_n moments. In between two such consecutive moments the vector defining the action will preserve the same rank for the nonzero component.

Proposition 77 *By considering both the probability constraint from (8.2) and the one from (8.1) we obtain a sequence of time moments $T_1 \leq T_2 \leq ... \leq T_{N-1} \leq T_N = T$, where T_n represents the latest moment in time at which we have to stop working on the design task n, such that with a fixed α probability the team could still have the time to achieve the minimal performance levels for the remaining design tasks $n+1,\ldots,N$. We call from now on the T_n moments the "latest stopping times" of the design tasks.*

Proof. Since the solving times of the design tasks are considered independent random variables, and each of them is a sum of independent identically exponentially distributed random variables (i.e. the solving times of the activities), the $\sum_{n=n(t)+1}^{N}\sum_{l=1}^{l_{\min}(n)} S_n +$

$\sum_{l=l(n,t)+1}^{l_{\min}(n)} S_{n(t)}$ from the inequality (8.1) is distributed Erlang-$\overline{k} := \sum_{n=n(t)+1}^{N} \sum_{l=1}^{l_{\min}(n)} n_a(n) + \sum_{l=l(n,t)+1}^{l_{\min}(n)} n_a(n(t))$, with mean $\frac{\overline{k}}{\mu}$. Thus for any state: x_t the inequality (8.1) is equivalent to $\sum_{j=0}^{\overline{k}-1} \frac{[\mu(T-t)M]^j}{j!} e^{-\mu(T-t)M} \leq 1-\alpha$, which implies a maximal number of possible activities to be done in $(T-t)$ review periods. Since we assumed that for any particular design task the number of activities needed to solve each of its levels is the constant, the number of possible levels to be achieved in $(T-t)$ review periods with the probability α for each design task is constant as well.

Thus, if we want to arrive in $(T-t)$ review periods in the rewarded region (i.e. corresponding to a fully functional product) we have always to choose our actions such that the minimal levels should be always achieved. Thus, the latest stopping time T_n for the design task n can be obtained by subtracting from the deadline the time needed to solve the minimal performance levels on each of the design tasks from $n+1$ to N. Consequently, T_n is the largest time moment verifying $\sum_{q=n(t)+1}^{N} \sum_{l=1}^{l_{\min}(q)} \sum_{j=0}^{n_a(q)-1} \frac{[\mu(T-T_n)M]^j}{j!} e^{-\mu(T-T_n)M} \leq 1-\alpha$. ■

The real "total decision time" available for the decision maker will be obtained after subtracting from the total time horizon (given by the deadline) the latest stopping of the first design task, as well as the time needed to solve it to its minimal performance level with the probability α. For this time we should seek to decide, for each design task n the number (i.e. an integer) of performance levels (all the performance levels of a given design task are equal) to be given to the engineers so as to maximize the total reward without exceeding the deadline.

Consequently, the horizon T has to be initially chosen such that at least all the minimal performance levels can be solved with a given probability. If we denote by T_0 the minimal time horizon, we have to initially have $T \geq T_0$ for a non-trivial control problem.

Remark 78 *The time moments T_n are not necessarily integer positive numbers. However, we may truncate the time moments necessary to achieve the minimal performance levels on each of the design tasks from $n+1$ to N, for each $n \in \{1, \ldots, N-1\}$.*

Lemma 79 *Consider that the design tasks in the given sequence can be performed by the team of engineers one after the the other and they can start only at the beginning of a review period. Then, by restricting the action space of S_0, the stochastic dynamic decision model S_0 may be formulated as a stochastic dynamic decision model S_1, where the actions will have at most one nonzero component which will be decided at the T_n moments. In between two such consecutive moments the vector defining the action will preserve the same rank for the nonzero component.*

Proof. Let T_n for each $n \in \{1, \ldots N-1\}$ be the latest stopping times of the design tasks from the sequence considered in the sequential NPD control problem. Let $T_N = T$. Then we have:

I.a. $t < T_{i_0}$, $x_t(i_0) < l_{\min}(i_0)$:

I.a.i. if $x_t \in \overline{X}(t) - X(t)$ and $\mathrm{a}_{t,\max}(n) + x_t(n) < l_{\min}(n)$, then

$a_t = (0, ..., 0, a_t(i_0) = 0, ..., 0) = 0_{\mathbb{R}^N}$,

I.a.ii. if $x_t \in \overline{X}(t) - X(t)$ and $a_{t,\max}(n) + x_t(n) \geq l_{\min}(n)$, then $a_t = (0, ..., 0, a_t(i_0) \neq 0, ..., 0)$.

I.b. $t < T_{i_0}$, $l_{\min}(i_0) \leq x_t(i_0) < L_{\max}(i_0)$ then $a_t = (0, ..., 0, a_t(i_0) \neq 0, ..., 0)$, or $a_t = (0, ..., 0, a_t(i_0 + 1) \neq 0, ..., 0)$, since we have considered that a design task always starts at the beginning of a review period. I.c. $t < T_{i_0}$, $x_t(i_0) = L_{\max}(i_0)$ then $a_t = (0, ..., 0, a_t(i_0 + 1) \neq 0, ..., 0)$,

II.a. $t = T_{i_0}$, $x_t(i_0) < l_{\min}(i_0)$, then $a_t = (0, ..., 0, a_t(i_0) = 0, ..., 0)$,

II.b $t = T_{i_0}$, $x_t(i_0) \geq l_{\min}(i_0)$, then $a_t = (0, ..., 0, a_{x_t}(i_0 + 1) \neq 0, ..., 0)$,

III.a. $t > T_{i_0}$, $x_t(i_0) < l_{\min}(i_0)$, $a_t = (0, ..., 0, a_t(i_0) = 0, ..., 0) = 0_{\mathbb{R}^N}$

for any $x_t \in X_t, t \in \{1, \dots T\}$, and i_0 with $l(j,t) > 0$ for any $j \leq i_0$, and $l(j,t) = 0$ for any $N \geq j > i_0$, where we denote by $x_t(i) = l(i,t)$ the $i-$th coordinate of the vector x_t. ■

Definition 80 *Let $\{\xi_t\}_{t\in\{0,1,2,...\}}$ a stochastic dynamic decision model with countable state space defined by the tuple $(X, A, D, p, (p_t)_{t\in\{0,1,2,...\}}, (r_t)_{t\in\{0,1,2...\}})$. We say that the transition probabilities $p_t(h_t, a_t, x_{t+1})$ are coupled if there exists the following:*

1) a sequence of independent controlled random variables $\{Y_t^{a_t}\}_{t=0,1,...}$ with values in an arbitrary countable set F, where $a_t \in A(t)$.

2) a sequence of measurable functions $f_t : H_{t-1} \times A(t-1) \times F^{t-1} \to X(t)$ such that

$$\Pr((\xi_t = x_t)|h_{t-1}, a_{t-1}) = \Pr\left\{f_t\left(h_{t-1}, a_{t-1}, \{Y_s^{a_s}\}_{s=0,...,t-1}\right) = x(t)\right\}$$

$$= \Pr\left\{\{y_s^{a_s}\}_{s=0,...,t-1} \mid f_t\left(h_{t-1}, a_{t-1}, \{y_s^{a_s}\}_{s=0,...,t-1}\right) = x(t)\right\}$$

where $h_{t-1} = (x_0, a_0, \dots, x_{t-2}, a_{t-2}, x_{t-1})$, and $y_t^{a_t}$ is an instance (value) of the random variable $Y_t^{a_t}$.

Remark 81 *We may rewrite equivalently this definition as*

$$\xi_t = f_t\left(\{\xi_s\}_{s=0,1,...,t-1}, \{\pi_s\}_{s=0,...,t-1}, \{Y_s^{\pi_s}\}_{s=0,...,t-1}\right)$$

where $\pi = (\pi_t)_{t=0,1,.....}$ is a policy of the process $\{\xi_t\}_{t=0,1,...}$, $\pi_t(x_0, .., x_t) \overset{not}{=} a_t \in A(t)$.

Consequently, the coupling property of the transition probabilities can be written either on any sample path or with random variables. We will study the process properties on sample paths, hence with states and decisions (values and realizations of random variables and policies).

Sequential decision processes with coupled transition probabilities do appear in many applications to queues and related processes. Thus, in (Liu et al., 1995) the mathematical setting used through the paper is the following:

"Let (Ω, F, P) be a probability space, where a generic element in Ω is written in the form $\omega = \left(\omega_1^0, \omega_1^1, \dots, \omega_n^0, \omega_n^1, \dots\right)$. The coordinate processes $(\phi_n^m)_{m,n}$ on Ω is defined as $\phi_n^m(\omega) = \omega_n^m$ for all $n = 1, 2, \dots, m = 0, 1$, so that $\phi = \left(\phi_1^0, \phi_1^1, \dots, \phi_n^0, \phi_n^1, \dots\right)$. Here, ϕ_n^0 and ϕ_n^1 may represent the arrival time and the service requirement of the n-th customer in a queuing system, respectively.

The stochastic processes $(X_n)_n$ considered are of the form

$$X_n = f_n\left(\phi, (X_i)_{i=1}^{n-1}, (U_i)_{i=1}^{n-1}\right) \in S_n, n = 2, 3, \ldots,$$

where the initial state $X_1 \in S_1$ is known, $U_i \in A$ for $i = 1, \ldots, n-1$, and f_n is a measurable mapping from $\Omega \times \prod_{i=1}^{n-1} S_i \times A^{n-1}$ into S_n, respectively. Here, $u = (U_n)_n$ is the *control* and A is the action set."

Also in (Altman and Stidham Jr., 1995) we encounter the same coupling assumption, in the sense that the state after a transition is distributed as a deterministic function of the current state and two random variables, one of which is controllable and the other uncontrollable:

"For any action a let Y^a be random elements of a partially ordered standard Borel space $\mathcal{Y}$. Denote the partial order on $\mathcal{Y}$ by $>$. Let Z be a random element of a standard Borel space $\mathcal{Z}$. We assume that Y^0, Y^1 and Z are mutually independent. There exists a function $g : \mathcal{X} \times \mathcal{Y} \times \mathcal{Z} \to \mathcal{X}$, measurable with respect to the product σ-field on $\mathcal{X} \times Y \times \mathcal{Z}$ and non-decreasing in the first argument, such that

$$q(B : x, a) = P\{X_1 \in B | X_0 = x, A_0 = a\} = \{g(x, Y^a, Z) \in B\}, x \in \mathcal{X}, a \in \mathcal{A}$$

for every Borel subset B of $\mathcal{X}$."

We recall from Section 8.3 our assumption that once the process enters a state which is possible, but not admissible, it remains in that state until the deadline. Thus, in such a non-admissible state the only admissible action until the deadline is $0_{\mathbb{R}^N}$.

Proposition 82 *Consider that the design tasks in the given sequence can be performed by the team of engineers one after the the other and they can start only at the beginning of a review period. Then, the transition probabilities of the stochastic dynamic decision model S_1 are coupled and their system equation is*

$$x_{t+1} = x_t + y_t^{a_t} \tag{8.5}$$

where $x_t = (l(1,t), ..., l(N,t))$ is the state of the system, $a_t = (a_t(1), ..., a_t(N))$ is an action in $A_t(x_t)$, such that $a_t(i) \leq \min\{L_{\max}(i), a_{t,\max}(i)\}$ for $1 \leq i \leq N$, and also $y_t^{a_t} = (w_t(1, a_t(1)), ..., w_t(N, a_t(N)))$ is an instance of a random variable with probability distribution parameterized only by the a_t such that $0 \leq w_t(i, a_t(i)) \leq a_t(i)$, at the time instant t.

Proof. We prove that for any admissible history $(x_0, a_0, \ldots, x_T)$ the system equation 8.5 is respected.

For $t = 0$ we have $X(0) = \{x_0\} = \{0_{\mathbb{R}^N}\}$.

Fix $t \in [1, T]$ and take $x_t \in \overline{X}(t)$, and denote the $i-$th coordinate of x_t by $x_t(i)$. Then there exists an i_0 such that for x_t, $x_t(j) = l(j,t) > 0$ for any $j \leq i_0$, and $x_t(j) = l(j,t) = 0$ for any $i_0 < j \leq N$.

We prove now that if we take an admissible action of the stochastic decision model S_1 at an arbitrary time instant, we can construct a well-defined system equation of the type 8.5 at that time instant. For that, we will consider several cases.

I. $t < T_{i_0}$.

I.a. If $x_t(i_0) < l_{\min}(i_0)$, then the action is always of the form:

I.a.i. if $x_t \in \overline{X}(t) - X(t)$ and $\mathrm{a}_{t,\max}(n) + x_t(n) < l_{\min}(n)$, then $a_t = (0, ..., 0, a_t(i_0) = 0, ..., 0) = 0_{\mathbb{R}^N}$,

I.a.ii. if $x_t \in \overline{X}(t) - X(t)$ and $\mathrm{a}_{t,\max}(n) + x_t(n) \geq l_{\min}(n)$, then $a_t = (0, ..., 0, a_t(i_0) \neq 0, ..., 0)$.

Thus $y_t^{a_t} = (0, ..., 0, w_t(i_0, a_t(i_0)), 0, 0, ..., 0)$, where $0 \leq w_t(i_0, a_t(i_0)) \leq a_t(i_0)$ are the effective worked levels on the design task i_0 during the review period $[t, t+1)$ if we took the action a_t at the time instant t.

I.b. If $l_{\min}(i_0) \leq x_t(i_0) < L_{\max}(i_0)$, then the action is either of the type $a_t = (0, ..., 0, a_{x_t}(i_0) \neq 0, 0, ..., 0)$, with $y_t^{a_t} = (0, ..., 0, w_t(i_0, a_t(i_0)), 0, ..., 0)$, or $a_t = (0, ..., 0, a_{x_t}(i_0 + 1) \neq 0, ..., 0)$, with $y_t^{a_t} = (0, ..., 0, w_t(i_0 + 1, a_t(i_0 + 1)), 0, ..., 0)$, since we have considered that a design task always starts at the beginning of a review period.

I.c. If $x_t(i_0) = L_{\max}(i_0)$, then the action is $a_t = (0, ..., 0, a_{x_t}(i_0 + 1) \neq 0, ..., 0)$, with $y_t^{a_t} = (0, ..., 0, w_t(i_0 + 1, a_t(i_0 + 1)), 0, ..., 0)$

II $t = T_{i_0}$.

II.a. If $x_t(i_0) < l_{\min}(i_0)$, then the action is always of the form $a_t = (0, ..., 0, a_t(i_0) = 0, ..., 0)$. Thus $y_t^{a_t} = 0_{\mathbb{R}^N}$.

II.b. If $x_t(i_0) \geq l_{\min}(i_0)$, then the action is of the form $a_t = (0, ..., 0, a_{x_t}(i_0 + 1) \neq 0, ..., 0)$, with $y_t^{a_t} = (0, ..., 0, w_t(i_0 + 1, a_t(i_0 + 1)), 0, ..., 0)$.

III $t > T_{i_0}$.

III.a. $x_t(i_0) < l_{\min}(i_0)$, then, consistently with the case IIa, we have the same unique action and its corresponding y^{a_t}, both being $0_{\mathbb{R}^N}$.

Thus the action remains admissible for S_1, since on the control intervals included in (T_{i_0}, t), the team of engineers cannot continue working on the design task i_0.

III.b. $x_t(i_0) \geq l_{\min}(i_0)$. This case is impossible, since an admissible policy for S_1 cannot ask the team of engineers to work on the design task i_0 on the intervals $(T_{i_0}, T_{i_0+1}]$, ..., $(T_{n-1}, T]$. An optimal decision rule takes actions with the i_0-th component non-zero only on the intervals $(T_0 - 1, T_0]$, ..., $(T_{i-1}, T_i]$. ■

Remark 83 *The random variables $Y_t^{a_t}$ depend on the pair (h_t, a_t) only through the action a_t (i.e. the action indicates by its nonzero component the current design task to work on, as well as how many levels the team should work on it), and they are independent random variables.*

Corollary 84 *Let $s \in \{0, \ldots, T-1\}$. Then for any $(h_s, a_s) \in H_s \times A_s(x_s)$ we have* $x_{s+1} := \sum_{t=0}^{s} y_t^{a_t}$.

Corollary 85 *For any admissible history h_T of the stochastic dynamic decision model S_1 $(\exists)$ $k \in \{1, \ldots, N\}$ and $0 = t_0 < t_1 < \ldots < t_k \leq T$, with $t_i \in \mathbb{Z}_+$, $t_i \leq T_{i+1}$, $i \in \{1, .., N-1\}$ and $t_N \leq T$ such that for any $t \in \{t_i, \ldots, t_{i+1} - 1\}$, $0 \leq i \leq k-1$ we have $y_t^{a_t} = (0, ..., 0, w_t(i+1, a_t(i+1)) \neq 0, 0, ..., 0)$ and for any $t \in \{t_k, \ldots, T\}$ we have $y_t^{a_t} = 0_{\mathbb{R}^N}$.*

Corollary 86 *For any admissible history h_T of the Markov decision process S_1 if $x_T = (l(1, T), ..., l(N, T))$ then $l(i, T) = \sum_{s=t_{i-1}}^{t_i - 1} w_s(i, a_s(i))$*

For any given history we give in the following Corollary a formal characterization of the realized value of the random number of review periods (i.e. τ_i) on which the team works on the design task i. Thus we will define

- $\sigma_i^1 = \min\left\{t \in \mathbb{Z}_+^* \left| \sum_{s=1}^{t} y_{t_{i-1}+s-1}^{a_{t_{i-1}+s-1}(i)} \geq l_{\min}(i) \right.\right\}$ as the smallest integer time t at which the sum of the levels realized by the engineer can exceed the minimal level. So the engineers should work on the task i until at least time σ_i^1.
- $\sigma_i^2 = \min\left\{t \in \mathbb{Z}_+^* \left| \sum_{s=1}^{t} y_{t_{i-1}+s-1}^{a_{t_{i-1}+s-1}(i)} \geq L_{\max}(i) \right.\right\}$ as the smallest integer time t at which the sum of the levels realized by the engineer can exceed the maximal level. So the engineers should stop working on the task i before time σ_i^2.

Corollary 87 *For any admissible history h_T of the stochastic dynamic decision model S_1 there exists the numbers $D_i \in \{1, \ldots, T_i - t_{i-1}\}$, $i \in \{1, \ldots N\}$ such that $t_i = t_{i-1} + \tau_i$, where $\tau_i = \min\left\{T_i - t_{i-1}, \max\left(\sigma_i^1, D_i\right), \min\left(\sigma_i^2, D_i\right)\right\}$.We recall that T_i was the latest time at which the team should stop working on the design task i such that there is enough time to perform the remaining design task at least at their minimal level, and from the Corollary 85 we have that the team is asked to work on the design task i only for any discrete decision point $t \in \{t_i, \ldots, t_{i+1} - 1\}$.*

Proof. The existence of the sequence $0 \leq t_1 \leq \ldots \leq t_k \leq T$, with $t_i \in \mathbb{Z}_+$, $t_i \leq T_{i+1}$ is due to Corollary 85.

In order to prove the desired equality we consider several cases.

I. If $\sigma_i^1 \geq T_i - t_{i-1}$ then $\tau_i = T_i - t_{i-1}$, and we can consider an arbitrary value for $D_i \in \{1, \ldots, T_i - t_{i-1}\}$.

II. If $\sigma_i^1 < T_i - t_{i-1}$ then

II.a. if $\sigma_i^1 = t_i - t_{i-1}$ then D_i can take any integer value in between 1 and σ_i^1.

II.b. if $t_i - t_{i-1} = \sigma_i^2 \leq T_i - t_{i-1}$ then D_i can take any integer value in between σ_i^2 and $T_i - t_{i-1}$.

II.c. if $\sigma_i^1 \leq t_i - t_{i-1} \leq \sigma_i^2$ then $D_i := t_i - t_{i-1}$. ■

8.5 Eliminating the sample paths with zero final reward

In order to compute the expected value of the total reward for the S_1 model we first construct an equivalent model with reward per period structure. Secondly, we show that S_1 can be obtained from this last model by introducing a series of stopping sets. Finally, we obtain the functional equations of the S_1 model starting from the functional equations of a stochastic dynamic model with stopping sets.

Proposition 88 *Consider the stochastic dynamic decision model S_1. Then there exists the functions: $r_0 : X(0) \to R$, $r_t : X(t-1) \times A(t-1) \times X(t) \to \mathbb{R}$, $t \in \{1, ..., T\}$, such that for any admissible history $(x_0, a_0, \ldots, x_{T-1}, a_{T-1}, x_T)$ with final state x_T, the final reward can be written as $\rho_T(x_T) = r_0(x_0) + \sum_{t=1}^{T} r_t(x_{t-1}, a_{t-1}, x_t)$. (i.e. we can obtain a structure with both final reward r_T and reward per review period r_t, $t \in \{0, T-1\}$ equivalent with the initial structure having only final reward)*

Proof. For the review period $[0,1)$ and $h_0 = x_0$ we define $r_0(x_0) = r_0(x_0, a_0) = 0$ for all $a_0 \in A(0)$.

For the review period $[t, t+1)$, $1 \leq t \leq T-1$, for $x_{t-1} \in X(t-1)$, $x_t \in X(t)$, $x_t = (l(1,t), .., l(i,t), 0, .., 0)$, $a_{t-1} \in A_{t-1}(x_{t-1})$ and $a_t \in A_t(x_t)$ we take

$r_t(x_{t-1}, a_{t-1}, x_t) = r_t(x_{t-1}, a_{t-1}, x_t, a_t) =$

$$= \begin{cases} V(i) \cdot w_{t-1}(i, a_{t-1}(i)) & t < T_i \text{ or } (t = T_i \text{ and } l(i,t) \geq l_{\min}(i)) \\ -\sum_{j=1}^{i} V(j) \cdot l(j,t) & t = T_i \\ & \text{and } l(i,t) < l_{\min}(i) \\ 0 & t > T_i \end{cases}$$

For any $a_{T-1} \in A(T-1)$, $x_{T-1} \in X(T-1)$ and $x_T = (l(1,T), ..., l(N,T))$ we consider the final reward:

$$r_T(x_{T-1}, a_{T-1}, x_T) = \begin{cases} V(N) \cdot w_{T-1}(N, a_{T-1}(N)) & l(N,t) \geq l_{\min}(N)) \\ -\sum_{j=1}^{N} V(j) \cdot l(j,T) & 0 < l(N,t) \text{ and} \\ & l(N,t) < l_{\min}(N) \\ 0 & l(N,T) = 0 \end{cases}$$

Then the desired equality holds by Corollary 86 which says that $l(i,T) = \sum_{s=t_{i-1}}^{t_i - 1} w_s(i, a_s(i))$.

■

In Proposition 88 we constructed a reward structure having for any review period $[t, t+1)$ a reward depending on an admissible history of the process only by the means of the state of the system at the time instant t, x_t, and of both the state, and the action at the time instant $t-1$, x_{t-1}, a_{t-1}. However, this reward does not depend on neither the state of the system, nor the action at the time instant $t+1$.

From now on we will call the new model, *the stochastic decision model S_1 with the nonzero reward structure per review periods* to distinguish it from the initial S_1 which will be called *the stochastic decision model S_1 with the final or terminal reward structure*.

Remark 89 *The stochastic dynamic decision model S_1 can have as stopping sets $\overline{\Phi}_t = \varnothing$ for $t \in \{0, 1, .., T-1\}$ and $\overline{\Phi}_T := \{h_T | \exists\ 1 \leq i_0 \leq N, \quad x_T(i_0) = l(i_0, T) < l_{\min}(i_0)\}$ since stopping the process at its end, or stopping it before are equivalent from the point of view of the final reward (any stopped state during the process development can be retrieved among the states stopped at the end).*

We denote by $\{\xi_t\}_{t=0,...,T} := S_1^{stop}$ the stochastic dynamic decision model obtained from the Markov decision model S_1 with the nonzero reward structure per review periods by considering the stopping sets:

a) $\Phi_t := \emptyset$ for $t \neq T_i$, $t \leq N-1$, $i \in \{1, .., N\}$, where $T_N = T$,

b) $\Phi_{T_i} := \{h_{T_i} \in H_{T_i} | x_{T_i} = (l(1,T_i) \ldots, l(i,T_i) \neq 0, 0, ..., 0),\ l(i,T_i) < l_{\min}(i)\}$, $i \in \{1, .., N\}$ and

c) for $N < t \leq T$: $\Phi_t := \{h_t \in H_t | x_t = (l(1,t) \ldots, l(N,t) \neq 0),\ l(N,t) = L_{\max}(N)\}$

Each design task starts at the beginning of a review period and from the most optimistic perspective can be finished during only one review period. This means that for the $N-$th design task the earliest starting point is $N-1$ and the earliest finishing time is N.

Thus, at the N-th moment it is mandatory to stop the process if the team reached the maximal level of the last design task. (so for each $N < t \leq T$).

Remark 90 *The model S_1^{stop} can be viewed as model with stopping sets, and with a random duration. Its duration equals the smallest value of the time moment t such that $x_t = (l(1,t)\ldots, l(N,t) \neq 0)$, $l(N,t) = L_{\max}(N)$. We denote this duration with t_N and we remark that $t_N \leq T$.*

For the model S_1^{stop} we observe that stopping at any time moment is equivalent with stopping only at the T_n moments because any state stopped between T_{n-1} and T_n will be stopped at the time moment T_n as well.

Proposition 91 *Let $\widetilde{u}_t^*$ be the optimal reward of the process $\{\xi_t\}_{t=0,\ldots,T} := S_1^{stop}$ for the period $[t,T]$ under the total expected reward criterion. Then $\widetilde{u}_t^*$ verifies the functional equation*

$$\widetilde{u}_t^*(x_{t-1}, a_{t-1}, x_t) =$$

$$= \left[r_t(x_{t-1}, a_{t-1}, x_t) + \max_{a_t \in A(t)} \sum_{x_{t+1} \in \overline{X}(t+1)} p_t(x_t, a_t, x_{t+1}) \widetilde{u}_{t+1}^*(x_t, a_t, x_{t+1}) \right] \cdot 1_{B_t}(x_t)$$

where $x_{t-1} \in \overline{X}(t-1), x_t \in \overline{X}(t)$ and $a_{t-1} \in A_{t-1}(x_{t-1})$.

Proof. Let us consider the set B_t of histories $h \in H_t$ which do not result in a stop, i.e.

$$B_t := \{h_t \in H_t : h_\nu \notin \Phi_\nu \text{ for } 1 \leq \nu \leq t\}.$$

We notice that the stopping sets depend on an admissible history of the process, h_t, only through the state x_t Thus, $1_{B_t}(h_t) = 1_{B_t}(x_t)$.

The proof is done by induction for t from T to 0. For $t = T$ we have $\widetilde{u}_T^*(x_{T-1}, a_{T-1}, x_T) = r_T(x_{T-1}, a_{T-1}, x_T) \cdot 1_{B_T}(x_T)$.

The induction step is obvious since according to ((Hinderer, 1970) pp.57) one can describe the stopped process by changing the reward functions r_t to functions $\overline{r}_t(h,a)$. Then we just have to define $\overline{r}_t(h,a) := r_t(h,a) \cdot 1_{B_t}(h)$, $t \in \mathbb{Z}$, $(h,a) \in H_t \times A(t)$. If $h \notin B_t$, then $(h, a_t, \ldots, x_m) \notin B_m$ for all $m > t$ and for all $(h, a_t, \ldots, x_m) \in H_m$, hence $r_m(h, a_t, \ldots, x_m, a_m) = 0$, hence

$$\tilde{u}_t^*(h) = 0, t \in \mathbb{N}, h \notin B_t.$$

Then, the optimality equation becomes:

$$\tilde{u}_t^*(h) = \max_{a \in D_t(h)} \left[r_t(h,a) + \sum_{x_{t+1}} p_t(h,a,x_{t+1}) \tilde{u}_{t+1}^*(h,a,x_{t+1}) \right] \cdot 1_{B_t}(h), t \in \mathbb{N}, h \in H_t$$

where $D_t(h) \overset{not}{=} A_t(h)$ being the decision rule that determines the set of admissible actions at the decision point t.

More precisely for $t \in \{1, .., T-1\}$ the optimality equations are:

$$\widetilde{u}_t^*(x_{t-1}, a_{t-1}, x_t) =$$

$$= \left[r_t(x_{t-1}, a_{t-1}, x_t) + \max_{a_t \in A_t(x_t)} \sum_{x_{t+1} \in \overline{X}(t+1)} p_t(x_t, a_t, x_{t+1}) \, \tilde{u}_{t+1}^*(x_t, a_t, x_{t+1}) \right] \cdot 1_{B_t}(x_t)$$

For $t = 0$ we have $\widetilde{u}_0^*(x_0) = \max_{a_0 \in A_0(x_0)} \sum_{x_1 \in \overline{X}(1)} p_0(x_0, a_0, x_1) \, \tilde{u}_1^*(x_0, a_0, x_1)]$ since $1_{B_0}(x_0) = 1$, while for $t = T$ we have $\widetilde{u}_T^*(x_{T-1}, a_{T-1}, x_T) = r_T(x_{T-1}, a_{T-1}, x_T) \cdot 1_{B_T}(x_T)$. ■

Proposition 92 *S_1^{stop} is a non-stationary Markov decision model with stopping sets.*

Proof. We already noticed that we preserve the initial rewards value while we may consider transition probabilities which do not depend on the actions, and rewards which depend on the actions and are nonzero only for the performance levels values in between the $l_{\min}(\cdot)$ and $L_{\max}(\cdot)$. So, $r_t(x_t, a_t) = \sum_{x_{t+1} \in X(t+1)} p_t(x_t, a_t, x_{t+1}) \, r_t(x_t, a_t, x_{t+1})$ equal $\overline{r}_t(x_t, a_t) = \sum_{x_{t+1} \in \overline{X}(t+1)} p_t(x_t, a_t, x_{t+1}) \, \overline{r}_t(x_t, a_t, x_{t+1})$, where $x_t \in X(t)$, $t \in \{0, 1, .., T-1\}$. But in a Markov model its rewards per period should depend on the state known at the beginning of the review period. For that, in the stopped process, we reduce the cost structure per period. Thus, the reward for the period $[t, t+1)$ equals $r_t(x_t, a_t)$, and we still have the same reward under the maximal expected total reward criterion.

Let $\overline{u}_t(x_t)$ be the optimum of the expected value for the reward during the period $[t, T]$ for the process S_1^{stop} obtained by reducing the reward structure. If $x_t \in X(t)$ is an admissible state for any t (i.e. we eliminate the sample paths which do not belong to $B_t = \{h_t \in H_t : h_\nu \notin \Phi_\nu for 1 \leq \nu \leq t\}$ and thus are at moment stopped) then the optimum of the expected value for the reward during the period $[T-1, T]$ is

$$\overline{u}_{T-1}(x_{T-1}) = \max_{a_{T-1} \in A_{T-1}(x_t)} \sum_{x_T \in X(T)} p_{T-1}(x_{T-1}, a_{T-1}, x_T) \, r_{T-1}(x_{T-1}, a_{T-1}, x_T)$$

and the optimum of the expected value for the reward during the period $[t, T]$, with $t \in \{0, .., T-1\}$ is

$$\overline{u}_t(x_t) = \max_{a_t \in A_t(x_t)} [r_t(x_t, a_t) + \sum_{x_{t+1} \in X(t+1)} p_t(x_t, a_t, x_{t+1}) \, \overline{u}_{t+1}^*(x_{t+1})]$$

Thus, we have for $t = T-1$

$$\begin{aligned} \overline{u}_{T-1}(x_{T-1}) &= \max_{a_{T-1} \in A_{T-1}(x_t)} \sum_{x_T \in X(T)} p_{T-1}(x_{T-1}, a_{T-1}, x_T) \, r_{T-1}(x_{T-1}, a_{T-1}, x_T) \\ &= \max_{a_{T-1} \in A_{T-1}(x_t)} r_{T-1}(x_{T-1}, a_{T-1}) \\ &= \widetilde{u}_{T-1}^*(x_{T-2}, a_{T-2}, x_{T-1}) - r_{T-1}(x_{T-2}, a_{T-2}, x_{T-1}), \end{aligned}$$

for $t \in \{1, .., T-1\}$

$$\overline{u}_t(x_t) = \widetilde{u}_t^*(x_{t-1}, a_{t-1}, x_t) - r_t(x_{t-1}, a_{t-1}, x_t),$$

and for $t = 0$ holds $\overline{u}_0(x_0) = \widetilde{u}_0^*(x_0)$.

We recall that $X(t)$, $t \in \{1, .., T-1\}$ does not contain stopping states and $X(T)$ contains those states x_T with $l_{\min}(i) \leq l(i,T) \leq L_{\max}(i)$ for any $i \in \{1, .., N\}$. So, for $x_t \in X(t)$ we have both $\widetilde{u}_t^*(x_t) > 0$ and $r_t(x_{t-1}, a_{t-1}, x_t) \geq 0$.

Now we can conclude that the actions, the transitions probabilities and the rewards at any moment t depend on an admissible history only by the means of the system state at the decision moment t. Therefore, the process S_1^{stop} is a general or non-stationary Markovian model with stopping sets (see (Hinderer, 1970) pp. 39, pp.57). ■

Corollary 93 *For any admissible history of the decision process S_1^{stop} there exists $t_i \in \mathbb{Z}_+$, with $i \in \{0, .., N\}$, $t_i \leq T_{i+1}$ for $i \in \{0, .., N-1\}$, $T_N = T$, $t_N \leq T$ and $0 = t_0 < t_1 < \ldots < t_N \leq T$ such that:*

1. *for any $t \in \{t_i, \ldots, t_{i+1} - 1\}$, $i \in \{0, .., N-1\}$ we have*

$$y_t^{a_t} = (0, ..., 0, w_t(i+1, a_t(i+1)) \neq 0, 0, ..., 0)$$

2. *for any $t \in \{t_N, \ldots, T\}$ we have $y_t^{a_t} = 0_{\mathbb{R}^N}$.*

We call the moments t_i the transition moments of the process S_1^{stop}.

Proof. The Corollary 93 is a consequence of the Corollary 85, since on all the sample paths of the process with stopping sets $\{\xi_t\}_{t=0,\ldots,T} := S_1^{stop}$ all the design tasks are done, and for all of them the minimal performance level is achieved. For the design task N the team of engineers can finish the design task even before the deadline. ■

8.6 Associating the rewards to the transition moments of S_1^{stop}

In this section we establish the equivalency between the S_1^{stop} model and a model in which the actions are taken at the transition moments of the model S_1^{stop}. The new model reduces the number of decision moments. The purpose of this new reformulation of the initial sequential control model is to simplify the way of computing the expected value of the total reward.

The admissible actions for a design task n on which the team is currently working may vary from zero to $L_{\max}(n)$. They depend on the maximal number of design activities that can be done on this particular design task in between the decision moments t and T_n. In (8.2) there exists a condition of choosing an action which leads at least to the minimal performance level of each design task. It is obvious that an arbitrary decision moment t there exists states from which the minimal performance level of the $n-$th design task is not achievable (i.e with a probability greater than the safety margin α) even if the team will solve $a_{t,\max}(n)$ performance levels (i.e. the maximal number of performance levels that can be decided at the decision moment t for the design task n so that there is time left to solve for the remaining design tasks their minimal performance levels with a probability greater than the safety margin α) from t to T_n Since the reward of a stopped sample path is zero no matter when the stopping took place (i.e. the current sequential NPD control model does not look

as well for a minimization of the loss due to an aborted NPD project) we can assume that the team may work on the design task n until T_n, and the process is stopped only at T_n if the case appears. If from the performance level achieved for the design task n until the decision moment t the team cannot reach its minimal performance level until T_n we can assume that the state is preserved until T_n. In this case for the actions one can assume as admissible action either the zero action from the decision moment t until T (as considered previously when we proved that the S_1 model has coupled transition probabilities), or the action which from the decision moment t until T_n allows the exact achievement of the minimal performance level contradicting the last condition from (8.2). In this way both $X(s)$, $s \in \{t, T_n - 1\}$, and the action sets are enlarged in a controlled way (i.e. instead of the set of admissible states $X(s)$ we consider $\overline{X}(s)$).

It seems more natural, and gives a more intuitive reason for choosing the T_n moments as stopping moments if we require the team of engineers to work on a design task n until T_n in the following way:

1) if there still are admissible actions that lead to at least the minimal performance level, then take one of these actions

2) if there are no more admissible actions that lead to the minimal performance level (the last condition from (8.2) does not hold anymore), then enlarge the action set to allow the work until T_n with the purpose of realizing "the miracle" of achieving the minimal performance level.

However, all the proofs in this chapter (previously done, and following) hold for both ways of enlarging the action space.

Proposition 94 *(Proposition 1. (Kreps, 1977b)) Let a countable state, finite action decision problem $(X_1 \times X_2 \times \ldots, A, (p_t), U)$, where U is an extended real valued utility function defined on the space of complete histories $H := H_0 \times X_1 \times X_2 \times \ldots$. We denote for each policy π, time t, and history $h_t \in H_t := H_{t-1} \times X_t$ the expected total utility using π given h_t as $v_t(\pi, h_t) := E^{\pi}[U(h)|h_t]$. Then*

1) $v_t(\pi, h_t) = E^{\pi}[v_{t+1}(\pi, h_{t+1})|h_t]$, for all policies π, decision points $t \in \mathbb{N}$, and histories $h_t \in H_t$

2) $\sup_{\pi \in \Pi} v_t(\pi, h_t) = \max_{\pi \in \Pi} E^{\pi}\left[\sup_{\pi \in \Pi} v_{t+1}(\pi, h_{t+1}) \middle| h_t\right]$, for all $t \in \mathbb{N}$, and histories $h_t \in H_t$

The equations from Proposition 94 are called optimality equations for a generalized utility function and they are the analogue of the Bellman optimality equations in the case of additive utility.

Proposition 95 *Let $\{\xi_t\}_{t=0,\ldots,T} := S_1^{stop}$ the Markov decision model obtained from the stochastic dynamic model S_1 with the nonzero reward structure per review periods, with the stopping sets Φ_t and let $t_0 = 0 < t_1 < \ldots < t_N \leq T$ the transition moments of the process. We consider the process $\{\tilde{\xi}_t\}_{t=0,\ldots,T}$ defined by $\tilde{\xi}_t := \sum_{n=0}^{N-1} \xi_{t_n} 1_{t_n \leq t < t_{n+1}} + \xi_{t_N} \cdot 1_{t_N \leq t < T}$. In this decision process we consider that the decisions are taken at the transition moments t_n (i.e. at the moment t_n we start the design task $n+1$) and they*

are of the form $\tilde{D}_{n+1} := (D_{n+1}, a^{n+1}_{[t_n,T_{n+1}]})$, *where* $D_{n+1} \in \{1,\dots,T_{n+1}-t_n\}$ *and* $a^{n+1}_{[t_n,T_{n+1}]} = (a_{t_n}(n+1), a_{t_n+1}(n+1),\dots,a_{T_{n+1}-1}(n+1))$.

Then there exists for the process $\{\tilde{\xi}_t\}_{t=0,\dots,T}$ *a reward structure consisting from rewards obtained at the transition moments such that the optimal rewards of the processes* $\{\xi_t\}_{t=0,\dots,T}$ *and* $\{\tilde{\xi}_t\}_{t=0,\dots,T}$ *are equal.*

Proof. This proof starts with the construction of new reward structure. Afterwards, in order to prove the equality of the optimal rewards for the two processes we show that for any policy of the first process, there is a policy of the second process (and conversely) such that their associated rewards are equal. The second part of the proof uses the probability spaces associated with the two decision processes.

Let $r_t(x_{t-1}, a_{t-1}, x_t)$ the reward functions defined in Proposition 88, and also let $\overline{r}_t(x_{t-1}, a_{t-1}, x_t)$ be the rewards of the Markov decision process $\{\xi_t\}_{t=0,\dots,T} := S_1^{stop}$ and let $\overline{r}_{t\pi}(x_{t-1}, x_t)$ be the rewards of the Markov decision process $\{\xi_t\}_{t=0,\dots,T} := S_1^{stop}$ under the policy π.

We reduce the reward structure per period of the process $\{\xi_t\}_{t=0,\dots,T}$ using the first way discussed in Section 8.3. Consequently, for each review period $[t-1,t)$ we receive the reward $\sum_{x_t\in X(t)} \overline{r}_{t\pi}(x_{t-1},x_t)p_{t-1,\pi}(x_{t-1},x_t)$ at the beginning of it (i.e. at the time moment $t-1$).

We define the reward functions $\{\tilde{r}_n\}_{n=0,\dots,N}$ which represent the total rewards for the interval $[t_n,t_{n+1})$ expected at the time moment t_n and the function $\tilde{r}_N$ representing the final reward.

The rewards $\{\tilde{r}_n\}_{n=0,\dots,N}$ will be constructed using the second way (i.e. $r_1 \equiv 0$, $r_t(h,a) = r'_{t-1}(h)$) of reducing the reward structure per period depending on x_t, a_t, and x_{t+1} of a process to the one of a standard model (see Definition 65), applied to a review period determined by two transition moments, namely $[t_n,t_{n+1})$. This enables the process $\{\tilde{\xi}_t\}_{t=0,\dots,T}$ to inherit the stopping sets of the process $\{\xi_t\}_{t=0,\dots,T} := S_1^{stop}$.

Let $h_T = (x_0,a_0,...,x_{T-1},a_{T-1},x_T) \in H_T$ an admissible history of the process $\{\widetilde{\xi}_t\}_{t=0,\dots,T}$, and let $t_0 = 0 < t_1 < \dots < t_N \le T$ the transition moments associated with h_T. If the state at the time moment t_n is $x_{t_n} = (l(1,t),..,l(n,t),0,..,0)$, if the sample path of the process $\{\widetilde{\xi}_t\}_{t=0,\dots,T}$ is h_T, then on the interval $[t_n,t_{n+1}), n \in \{1,..,N-1\}$ the sum of the rewards equals $V(n)\sum_{s=t_{n-1}}^{t_n-1} w(n,a_s(n))$, where $\sum_{s=t_{n-1}}^{t_n-1} w(n,a_s(n)) = l(n,t)$.

The n is running until $N-1$ because according to the second way of reducing the reward structure we consider the reward realized in the interval $[t_{n-1},t_n)$ for the interval $[t_n,t_{n+1}), n \in \{1,..,N-1\}$ and the sample path $h_T = (x_0,a_0,...,x_{T-1},a_{T-1},x_T) \in H_T$

The final reward is $V(N)\sum_{s=t_{N-1}}^{t_N-1} w(N,a_s(N))$ and it corresponds to the t_N moment.

Let $n \in \{1,\dots,N-1\}$. If the $n-1-$th transition moment is t_{n-1}, with the corresponding action $\tilde{D}_n$ (i.e. $\widetilde{\pi}_{n-1}(t_{n-1}) = \tilde{D}_n$), if the $n-$th transition moment is t_n and if at moment t_n the process is in state x_{t_n}, we define as the reward at moment t_n for the interval $[t_n,t_{n+1})$ the expression $\tilde{r}_n\left(t_{n-1},t_n,\tilde{D}_n,x_{t_n}\right) = \sum V(n)\sum_{s=t_{n-1}}^{t_n-1} w(n,a_s(n))$. The sum is

taken over all the sample paths of the process determined by the time moments t_{n-1} and t_n such that $\sum_{s=t_{n-1}}^{t_n-1} w\left(n, a_s\left(n\right)\right) = l\left(n,t\right)$. Let $W\left(t_{n-1}, t_n, x_{t_n}\right)$ the set of all these sample paths.

Similarly we define the final reward $\tilde{r}_N\left(t_{N-1}, t_N, \tilde{D}_N, x_{t_N}\right)$.

For emphasizing the dependence of the rewards $\tilde{r}_n$ on the policy considered we denote them by $\tilde{r}_{n,\widetilde{\pi}}\left(t_{n-1}, t_n, x_{t_n}\right)$.

The expression of $\tilde{r}_n$ depends on the history of the process only through t_{n-1}, t_n, $\tilde{D}_n$, and through the state x_{t_n}(more precisely only through $x_{t_n}(n)$,i.e. the last nonzero component of the vector x_{t_n}).

For $n = 0$ (i.e. for the period $[0, t_1)$), the reward expected at moment $t_0 = 0$ is zero.

Let π a policy for the process $\{\xi_t\}_{t=0,...,T} := S_1^{stop}$. Using the Corollaries 85, 86, 87 we can associate to the policy π, a policy $\widetilde{\pi} = \{\widetilde{\pi}_n\}_{n\in\{0,1,..,N-1\}}$, $\widetilde{\pi}_n(x_{t_n}, t_n)$=$\widetilde{\pi}_n(t_n)$ =$\tilde{D}_{n+1}$ for the process $\{\tilde{\xi}_t\}_{t=0,...,T}$ by choosing an admissible sequence $(D_1, ..., D_N)$ (see Corollary 87), and by completing for any interval $[t_n, T_{n+1})$ the vector of admissible actions $\left(a_{t_n}(n+1), ..., a_{t_{n+1}-1}(n+1)\right)$ with arbitrary admissible choices in between the time moments $t_{n+1} - 1$ and $T_{n+1} - 1$.

Conversely, using the necessary number of components of the vector $a_{[t_n,T_{n+1}]}^{n+1} = (a_{t_n}(n+1), a_{t_n+1}(n+1), \ldots, a_{T_{n+1}-1}(n+1))$ we can obtain from a policy $\widetilde{\pi}$ a policy π. We say that the policies π and $\widetilde{\pi}$ obtained as discussed above constitute a pair of policies.

The key-observation in writing the expected value of the total reward for the process S_1^{stop} is that the set of all sample paths $(x_0, ..., x_T)$ can be written as a union taken over all the sets of transition moments $(t_1, .., t_N)$, and all the sets of the form $(y_0^{\pi_0}, ..., y_{T-1}^{\pi_{T-1}})$ such that for any $n \in \{1, .., N\}$ holds $l_{\min}(n) \leq \left(\sum_{t=t_{n-1}}^{t_n-1} y_t^{\pi_t}\right)(n) \leq L_{\max}(n)$. This remark is a consequence of the existence in the process S_1^{stop} of the sequence $(t_1, .., t_N)$ such that $x_T = \sum_{t=0}^{T-1} y_t^{\pi_t} = \sum_{n=1}^{N} \sum_{t=t_{n-1}}^{t_n-1} y_t^{\pi_t}$ and for any $t \in \{t_{n-1}, \ldots, t_n - 1\}$ we have $y_t^{\pi_t} = (0, ..., 0, w\left(n, \pi_t(n)\right), 0, .., 0)$.

Let π a policy for the process $\{\xi_t\}_{t=0,...,T} := S_1^{stop}$, and $\widetilde{\pi}$ a policy for the process $\{\tilde{\xi}_t\}_{t=0,...,T}$, both with their associated probability space. We want to prove that under the total reward criterion the policies are equivalent. For this we need to show that

$$\sum_{(x_0,...,x_T)} \sum_{t=0}^{T} \overline{r}_{t\pi}(x_{t-1}, x_t) P_\pi(x_0, ..., x_T)$$

equals the expression

$$= \sum_{\left(x_{t_1},...,x_{t_N},t_1,...,t_N\right)} \sum_{n=1}^{N} \widetilde{r}_{n\widetilde{\pi}}(t_{n-1}, t_n, x_{t_n}) P_{\widetilde{\pi}}(x_{t_1}, ..., x_{t_N}, t_1, ..., t_N)$$

Now, we will further re-write both members of the desired equality. Thus,

$$\sum_{(x_0,...,x_T)}\sum_{t=0}^{T}\overline{r}_{t\pi}(x_{t-1},x_t)P_\pi(x_0,...,x_T)=\sum_{(x_0,...,x_T)}\sum_{t=1}^{T}\overline{r}_{t\pi}(x_{t-1},x_t)P_\pi(x_0,...,x_T)$$

$$=\sum_{(x_1,...,x_T)}\sum_{t=1}^{T}\overline{r}_{t\pi}(x_{t-1},x_t)P_\pi(x_0,x_1,...,x_T)$$

$$=\sum_{(x_1,...,x_T)}\sum_{n=1}^{N}\sum_{t=t_{n-1}+1}^{t_n}\overline{r}_{t\pi}(x_{t-1},x_t)p_{t_0,x_0,\pi}\left(x_{t-1},x_t\right)$$

$$=\sum_{(x_1,...,x_T)}\sum_{n=1}^{N}\sum_{t=t_{n-1}+1}^{t_n}\overline{r}_{t\pi}(x_{t-1},x_t)\times p_{0,\pi}(x_0,x_1)...p_{t_1-1,\pi}(x_{t_1-1},x_{t_1})\times$$
$$\times p_{t_1,\pi}(x_{t_1},x_{t_1+1})...p_{t_2-1,\pi}(x_{t_2-1},x_{t_2})\times...$$
$$\times p_{t_{n-1},\pi}(x_{t_{n-1}},x_{t_{n-1}+1})...p_{t_n-1,\pi}(x_{t_n-1},x_{t_n})$$

where we denoted by $p_{t_0,x_0,\pi}\left(x_{t-1},x_t\right)$ the probability that at time moments $t-1$, and respectively t the system is in the state x_{t-1}, respectively x_t, being given the state and initial time moment and assuming that the policy π is the one being used. We notice that $p_{t;\pi}(x_t,x_{t+1})=p_{t;\pi}(y_t^{\pi_t})$.

Thus, we obtain

$$\sum_{(x_0,...,x_T)}\sum_{t=1}^{T}\overline{r}_{t\pi}(x_{t-1},x_t)P_\pi(x_0,...,x_T)=$$

$$\sum_{(t_1,...,t_N)}\sum_{(y_0^{\pi_0},...,y_{t_N-1}^{\pi_{t_N-1}})\text{ s.t.}\left(\sum_{t=t_{n-1}}^{t_n-1}y_t^{\pi_t}\right)(n)\le L_{\max}(n)),n\in\{1,...,N\}}^{}\sum_{t=1}^{T}\overline{r}_{t\pi}(y_t^{\pi_t})P_\pi(t_1,...,t_N,y_0^{\pi_0},..y_{t_{N-1}}^{\pi_{t_N-1}})=$$

$$\sum_{(t_1,...,t_N)}\sum_{(y_0^{\pi_0},...,y_{t_N-1}^{\pi_{t_N-1}})\text{ s.t.}\left(\sum_{t=t_{n-1}}^{t_n-1}y_t^{\pi_t}\right)(n)\le L_{\max}(n)),n\in\{1,...,N\}}^{}\sum_{t=1}^{T}\overline{r}_{t\pi}(y_t^{\pi_t})P_\pi(t_1,...,t_N)p(y_0^{\pi_0})...p(y_{t_{N-1}}^{\pi_{t_N-1}})=$$

$$\sum_{(t_1,...,t_N)}\sum_{(y_0^{\pi_0},...,y_{t_N-1}^{\pi_{t_N-1}})\text{ s.t.}\left(\sum_{t=t_{n-1}}^{t_n-1}y_t^{\pi_t}\right)(n)\le L_{\max}(n)),n\in\{1,...,N\}}^{}\sum_{n=1}^{N}\sum_{t=t_{n-1}+1}^{t_n}\overline{r}_{t\pi}(y_t^{\pi_t})\times P_{t_0\pi}(t_1,...,t_N)p(y_0^{\pi_0})...p(y_{t_{N-1}}^{\pi_{t_N-1}})=$$

$$\sum_{n=1}^{N}\sum_{(t_1,...,t_N)}\sum_{(y_0^{\pi_0},...,y_{t_N-1}^{\pi_{t_N-1}})\text{ s.t.}\left(\sum_{t=t_{n-1}}^{t_n-1}y_t^{\pi_t}\right)(n)\le L_{\max}(n)),n\in\{1,...,N\}}^{}\sum_{t=t_{n-1}+1}^{t_n}\overline{r}_{t\pi}(y_t^{\pi_t})\times P_{t_0\pi}(t_1,...,t_N)p(y_0^{\pi_0})...p(y_{t_{N-1}}^{\pi_{t_N-1}})=$$

$$=\sum_{(t_0,t_1)}\sum_{(y_0^{\pi_1},...,y_{t_1-1}^{\pi_{t_1-1}})\in U(t_0,t_1)}\left(\sum_{t=0}^{t_1-1}\overline{r}_{t\pi}(y_t^{\pi_t})\right)\times p(y_0^{\pi_0})...p(y_{t_1-1}^{\pi_{t_1-1}})p_{t_0,\pi}(t_0,t_1)+$$

$$+\sum_{(t_1,t_2)}\sum_{(y_{t_1},...,y_{t_2-1})\in U(t_1,t_2)}\left(\sum_{t=t_1}^{t_2-1}\overline{r}_{t\pi}(y_t^{\pi_t})\right)\times p(y_{t_1}^{\pi_{t_1}})..p(y_{t_2-1}^{\pi_{t_2}-1})p_{t_0,\pi}(t_1,t_2)+....$$

$$+\sum_{(t_{N-1},t_N)}\sum_{(y_{t_{N-1}}^{t_N-1},...,y_{t_N-1}^{\pi_{t_n}-1})\in U(t_{N-1},t_N)}\left(\sum_{t=t_{N-1}+1}^{t_N}\overline{r}_{t\pi}(y_t^{\pi_t})\right)\times p(y_{t_{N-1}}^{\pi_{N-1}})...p(y_{t_N-1}^{\pi_{t_N-1}})\times$$

$$\times p_{t_0,\pi}(t_{N-1},t_N)$$

where $U(t_{n-1},t_n)=\{(y_{t_{n-1}}^{\pi_{t_{n-1}}},...,y_{t_n-1}^{\pi_{t_n-1}})\mid l_{\min}(n)\leq\left(y_{t_{n-1}}^{\pi_{t_{n-1}}}+...+y_{t_n-1}^{\pi_{t_n-1}}\right)(n)\leq L_{\max}(n)\}$ and $P_\pi(t_1,...,t_N,y_0^{\pi_0},..y_{t_{N-1}}^{\pi_{t_N-1}})$=$P_\pi(x_0,...,x_T)$.

We used the fact that the random variables $Y_t^{a_t}$ are independent, and we parameterized them only as a function of the action (respectively policy).

We recall that if the state at the time moment t_n is $x_{t_n}=(l(1,t),..,l(n,t),0,..,0)$, if the sample path of the process $\{\widetilde{\xi}_t\}_{t=0,...,T}$ is h_T, then on the interval $[t_{n-1},t_n),n\in\{1,..,N-1\}$ the sum of the rewards equals $V(n)\sum_{s=t_{n-1}}^{t_n-1}w(n,\pi_t(n))$, where $l(n,t)=\sum_{s=t_{n-1}}^{t_n-1}w(n,\pi_t(n))$. We can further rewrite the previous expression, because it depends on the random variables $Y_t^{a_t}$ only through their nonzero components, which are at the time moment $t\in[t_n,t_{n+1})$ of the form $w(n+1,\pi_t(n+1))$.

As a consequence:

$$\sum_{t=1}^{T}\overline{r}_{t\pi}(x_{t-1},x_t)P_\pi(x_0,...,x_T)=$$

$$=\sum_{n=1}^{N}V(n)\sum_{(t_{n-1},t_n)}\sum_{[w(n,\pi_{t_{n-1}}(n)),...,w(n,\pi_{t_n-1}(n))]\in W(t_{n-1},t_n)}\left(\sum_{t=t_{n-1}}^{t_n-1}w(n,\pi_t(n))\right)\times$$

$$\times P\left(w\left(n,\pi_{t_{n-1}}(n)\right),..,w\left(n,\pi_{t_n-1}(n)\right)\right)p_{t_0,\pi}(t_{n-1},t_n)=$$

$$=\sum_{n=1}^{N}V(n)\sum_{(t_{n-1},t_n)}\sum_{k=l_{\min}(n)}^{L_{\max}(n)}kP(w(n,\pi_{t_n}(n))+...$$

$$+w\left(n,\pi_{t_{n+1}-1}(n)\right)=k)p_{t_0,\pi}(t_{n-1},t_n)$$

where $W(t_{n-1},t_n)=\{[w(n,\pi_{t_{n-1}}(n)),...,w(n,\pi_{t_n-1}(n))]\mid 0\leq w(n,\pi_s(n))\leq\pi_s(n),s\in\{t_n,...,t_{n+1}-1\},l_{\min}(n)\leq\sum_{s=t_{n-1}}^{t_n-1}w(n,\pi_s(n))\leq L_{\max}(n)\}$ and $\pi_s(n)$ is the n–th component of the vector π_s.

We denote by $p_{t_0,\pi}(t_{n-1},t_n)$ the probability that the $n-1-$th and the $n-$ th transition moments equal t_{n-1} respectively t_n. We remark that the random variables $w(n,\pi_t(n))$ do not depend on the moments t_{n-1} and t_n. They depend only on the policy π.

Now consider $\widetilde{\pi}$ a policy for the process $\{\widetilde{\xi}_t\}_{t=0,...,T}$, and let the probability space associated to it.

Let $W(t_{n-1},t_n,x_{t_n})=\{[w(n,\pi_{t_{n-1}}(n)),...,w(n,\pi_{t_n-1}(n))]\mid 0\leq w(n,\pi_s(n))\leq\pi_s(n),s\in\{t_n,...,t_{n+1}-1\},l_{\min}(n)\leq\sum_{s=t_{n-1}}^{t_n-1}w(n,\pi_s(n))=x_{t_n}(n)\leq L_{\max}(n)\}$.

Since $P_{\widetilde{\pi}}(x_{t_0},...,x_{t_N},t_0,...,t_N)=P_{\widetilde{\pi}}(x_{t_1},...,x_{t_N},t_1,...,t_N)=$

$$=p_{t_0,x_{t_0},\widetilde{\pi}}(t_1,x_{t_1})\times...\times p_{t_{N-1},x_{t_{N-1}},\widetilde{\pi}}(t_N,x_{t_N})=$$

$$=p_{t_0,x_{t_0},\widetilde{\pi}}(t_1)p_{t_0,x_{t_0}t_1,\widetilde{\pi}}(x_{t_1})....p_{t_{N-1},x_{t_{N-1}};\widetilde{\pi}}(t_N)p_{t_{N-1},x_{t_{N-1}},t_N,\widetilde{\pi}}(x_{t_N})=$$

$=p_{t_0,\widetilde{\pi}}(t_1)p_{t_0,t_1,\widetilde{\pi}}(x_{t_1})....p_{t_{N-1},\widetilde{\pi}}(t_N)p_{t_{N-1},t_N,\widetilde{\pi}}(x_{t_N})$
where
$p_{t_{n-1},t_n,\widetilde{\pi}}(x_{t_n}) = \sum P((y_s^{\pi_s})_{t\in\{t_{n-1},..,t_n-1\}}) = \sum P((w(s,\pi_s(n))_{t\in\{t_{n-1},..,t_n-1\}})=$

$$= \sum_{[w(n,\pi_{t_{n-1}}(n)),...,w(n,\pi_{t_{n-1}}(n))]\in W(t_{n-1},t_n,x_{t_n})} \prod_{t=t_{n-1}}^{t_n-1} P(w(t,\pi_t(n)).$$

It results that

$$\sum_{(x_{t_1},...,x_{t_N},t_1,...,t_N)} \sum_{n=1}^{N} \widetilde{r}_{n\widetilde{\pi}}(t_{n-1},t_n,x_{t_n})P_{\widetilde{\pi}}(x_{t_1},...,x_{t_N},t_1,...,t_N) =$$

$$= \sum_{(x_{t_1},...,x_{t_N},t_1,...,t_N)} \sum_{n=1}^{N} \widetilde{r}_{n\widetilde{\pi}}(t_{n-1},t_n,x_{t_n})\, p_{t_0,\widetilde{\pi}}(t_1)p_{t_0,t_1,\widetilde{\pi}}(x_{t_1})....p_{t_{n-1},\widetilde{\pi}}(t_n)p_{t_{n-1},t_n,\widetilde{\pi}}(x_{t_n}) =$$

$$= \sum_{n=1}^{N} V(n) \sum_{(t_{n-1},t_n,x_{t_n})} \sum_{[w(n,\pi_{t_{n-1}}(n)),...,w(n,\pi_{t_n-1}(n))]\in W(t_{n-1},t_n,x_{t_n})} \left(\sum_{t=t_{n-1}}^{t_n-1} w(t,\pi_t(n))\right) \times$$

$$\times \left(\prod_{t=t_{n-1}}^{t_n-1} P(w(t,\pi_t(n))\right) p_{t_0,\widetilde{\pi}}(t_{n-1},t_n) =$$

$$= \sum_{n=1}^{N} V(n) \sum_{(t_{n-1},t_n)} \sum_{k=l_{\min}(n)}^{L_{\max}(n)} kP\left(w\left(n,\pi_{t_n}(n)\right)+\ldots+w\left(n,\pi_{t_{n+1}-1}(n)\right)=k\right) p_{t_0,\widetilde{\pi}}(t_{n-1},t_n)$$

where $P_{t_{n-1},x_{t_{n-1}},t_n,\widetilde{\pi}}(x_{t_n}) = P_{t_{n-1},t_n,\widetilde{\pi}}(x_{t_n})$ is the probability that the state at the moment t_n is x_{t_n} knowing that the $n-$th and $n-1-$th transition moments were t_n, respectively t_{n-1} and that the state at the t_{n-1} moment was $x_{t_{n-1}}$; $p_{t_{n-1};x_{t_{n-1}},\widetilde{\pi}}(t_n) = p_{t_{n-1};\widetilde{\pi}}(t_n)$ is the probability that the $n-$th transition moment is t_n knowing that the previous transition moment was t_{n-1}, and t_{n-1}corresponding state was $x_{t_{n-1}}$, and $p_{t_0,\widetilde{\pi}}(t_{n-1},t_n)$ is the probability that the $n-$th and $n-1-$th transition moments are t_n, respectively t_{n-1}.

Thus, we obtain the equality of the total expected rewards of the pair of policies π and $\widetilde{\pi}$. Moreover, one can obtain the optimality equations of the process $\{\tilde{\xi}_t\}_{t=0,...,T}$.

Let $\tilde{u}_n^*\left(t_{n-1},t_n,\tilde{D}_n,x_{t_n}\right)$ for $n\in\{1,\ldots,N-1\}$ be the maximum for the total expected reward from the $n-$th transition moment up to the end of the process, knowing that the $n-$th transition moment is t_n, the state of the process at that moment is x_{t_n}, the previous transition has taken place at moment t_{n-1} and the decision at moment t_{n-1} was $\tilde{D}_n$. Also let $\tilde{u}_N^*\left(t_{N-1},t_N,\tilde{D}_N,x_{t_N}\right) = \tilde{r}_N\left(t_{N-1},t_N,\tilde{D}_N,x_{t_N}\right)\cdot 1_{B_{t_N}}(x_{t_N})$.

By iteratively applying the optimality equations from Proposition 94 and taking into account that there is no reward at any moment $t\neq t_n$ for $n\in\{1,\ldots,N\}$ and that the random variables $Y_t^{a_t}$ are independent, we obtain the following equations

$$\tilde{u}_{n-1}^*\left(t_{n-2},t_{n-1},\tilde{D}_{n-1},x_{t_{n-1}}\right) = [\tilde{r}_{n-1}\left(t_{n-2},t_{n-1},\tilde{D}_{n-1},x_{t_{n-1}}\right)$$

$$+\max_{\tilde{D}_n}\sum_{t_n}\sum_{\times_{t_n}\in\overline{X}(t_n)} p_{t_{n-1},\tilde{D}_n}(t_n)\, p_{t_{n-1},t_n,\tilde{D}_n}(x_{t_n})\,\tilde{u}_n^*\left(t_{n-1},t_n,\tilde{D}_n,x_{t_n}\right)]\cdot 1_{B_{t_n}}(x_{t_{n-1}})$$

where $n\in\{2,...,N-1\}$ and

$$\tilde{u}_0(x_0=0) = \max_{\tilde{D}_1}\sum_{t_1}\sum_{\times_{t_1}\in\overline{X}(t_1)} p_{t_0,\tilde{D}_1}(t_1)\, p_{t_0,t_1,\tilde{D}_1}(x_{t_1})\cdot\tilde{u}_1^*\left(t_0,t_1,\tilde{D}_1,x_{t_1}\right)$$

where $p_{t_{n-1},\tilde{D}_n}(t_n)$ is the probability that the $n-$th transition moment is t_n if at the moment t_{n-1} was taken the action $\tilde{D}_n$ and $p_{t_{n-1},t_n,\tilde{D}_n}(x_{t_n})$ is the probability that the state corresponding to the $n-$th transition moment is x_{t_n} knowing that the $n-$th and $n-1-$th transition moments were t_n, respectively t_{n-1}.

For any admissible sample path $(x_0, x_{t_1}, .., x_{t_N}, .., x_T)$ of the process $\{\tilde{\xi}_t\}_{t=0,...,T}$, $x_{t_n} \in X(t_n)$, $n \in \{1, .., N\}$, $x_T \in X(T)$ we may write

$$\tilde{u}_N^*\left(t_{N-1}, t_N, \tilde{D}_N, x_{t_N}\right) = \tilde{r}_N\left(t_{N-1}, t_N, \tilde{D}_N, x_{t_N}\right),$$

$$\tilde{u}_{n-1}^*\left(t_{n-2}, t_{n-1}, \tilde{D}_{n-1}, x_{t_{n-1}}\right) = [\tilde{r}_{n-1}\left(t_{n-2}, t_{n-1}, \tilde{D}_{n-1}, x_{t_{n-1}}\right) +$$

$$\max_{\tilde{D}_n} \sum_{t_n} \sum_{\times_{t_n} \in X(t_n)} p_{t_{n-1},\tilde{D}_n}(t_n)\, p_{t_{n-1},t_n,\tilde{D}_n}(x_{t_n})\, \tilde{u}_n^*\left(t_{n-1}, t_n, \tilde{D}_n, x_{t_n}\right)],$$

$$\tilde{u}_0(x_0 = 0) = \max_{\tilde{D}_1} \sum_{t_1} \sum_{\times_{t_1 \in X(t_1)}} p_{t_0,\tilde{D}_1}(t_1)\, p_{t_0,t_1,\tilde{D}_1}(x_{t_1}) \cdot \tilde{u}_1^*\left(t_0, t_1, \tilde{D}_1, x_{t_1}\right)$$

■

For the process $\{\tilde{\xi}_t\}_{t=0,...,T}$ we may consider both ways of reducing the reward structure in between two transition moments. Using the first way we may define a reward of the type $V(n+1) \sum_{k=l_{\min}(n+1)}^{L_{\max}(n+1)} k \Pr\left[\sum_{s=t_n}^{t_{n+1}-1} w(n+1, a_s(n+1)) = k\right]$ for the review period $[t_n, t_{n+1})$.

We denote by $\overline{u}_n^*(t_n) = \overline{u}_n^*(t_n, x_{t_n})$, $n \in \{0, ., N-1\}$ the optimal reward during $[t_n, T]$ (which is the same as the reward for the period $[t_n, t_N]$) constructed for the process $\{\tilde{\xi}_t\}_{t=0,...,T}$ using the first way of reducing the reward structure, knowing that the $n-$th transition moment is t_n. Then the following functional equations are verified

$$\overline{u}_{N-1}^*(t_{N-1}) = \max_{\tilde{D}_N} \sum_{(t_N, x_{t_N})} \tilde{r}_N\left(t_{N-1}, t_N, \tilde{D}_N, x_{t_N}\right) p_{t_{N-1},\tilde{D}_N}(t_N)\, p_{t_{N-1},t_N,\tilde{D}_N}(x_{t_N}),$$

and for $n \in \{1, .., N-2\}$

$$\overline{u}_{n-1}^*(t_{n-1}) = \max_{\tilde{D}_n} [\overline{r}_n\left(t_{n-1}, \tilde{D}_n\right) +$$

$$+ \sum_{t_n} \sum_{\times_{t_n} \in X(t_n)} p_{t_{n-1},\tilde{D}_n}(t_n)\, p_{t_{n-1},t_n,\tilde{D}_n}(x_{t_n})\, \overline{u}_n^*(t_n)],$$

where $\overline{r}_n\left(t_{n-1}, \tilde{D}_n\right) = \sum_{(t_n, x_{t_n})} \tilde{r}_n\left(t_{n-1}, t_n, \tilde{D}_n, x_{t_n}\right) p_{t_{n-1},\tilde{D}_n}(t_n)\, p_{t_{n-1},t_n,\tilde{D}_n}(x_{t_n})$ is the reward corresponding to the period $[t_{n-1}, t_n)$ according to the first way of reducing the reward structure.

8.7 Sufficient statistic — Decide only on how many review periods the team of engineers should work

In this section we show that there exists a sequence of sufficient statistics in the sense of (Hinderer, 1970), and (Dynkin, 1965) so that the solutions of a reduced unidimensional optimality equation are optimal solutions for the model introduced in the previous section. This leads to the construction of a last unidimensional model which is equivalent with all the previous ones and which has a reduced number of decision moments, as well as smaller set of admissible states and actions.

Definition 96 *(Sufficient statistics, from (Hinderer, 1970)) Let F_n be an arbitrary non-empty set and $\lambda_n : H_n \to F_n$ an arbitrary surjective map. The sequence $(\lambda_n)_n$ is called a sufficient statistic of the stochastic dynamic decision process $(X, A, D, (p_n), (r_n))$ if there exists the functions D'_n, p'_n, r'_nsuch that:*

$$\left.\begin{array}{c} D_n(h) = D'_n(\lambda_n(h)), \\ p_n(h,a,x) = p'_n(\lambda_n(h), a, x), \\ r_n(h,a) = r'_n(\lambda_n(h), a), \end{array}\right\}, n \in \mathbb{N}, (h,a,x) \in H_{n+1}$$

and if $n \in \mathbb{N}$, $h \in H_n$, $h' \in H_n$, then $\lambda_n(h) = \lambda_n(h')$ implies $\lambda_{n+1}(h,a,x) = \lambda_{n+1}(h',a,x)$ for all $a \in D_n(h)$ and $x \in X$.

Theorem 97 *(see (Dynkin, 1965), Theorem 6.0 (Hinderer, 1970)) Let $(\lambda_n)_n$be a sufficient statistic of the stochastic dynamic decision model $(X, A, D, p, (p_n)_n, (r_n)_n)$. Then the following statements hold:*

1) There exists for any $n \in \mathbb{N}$ a unique map $\tilde{u}_n^* : F \to \bar{\mathbb{R}}$ such that $u_n^* = \tilde{u}_n^* \circ \lambda_n$

2) $(\tilde{u}_n^*)_n$ is a solution of the reduced optimality equation

Proposition 98 *For the process $\{\tilde{\xi}_t\}_{t=0,T}$ there exists a process $\{Z_t\}_{t=0,T}$ which is equivalent in the sense of (Hinderer, 1970) (i.e. there exists a sequence of sufficient statistics).*

Proof. Let $t_0 = 0 < t_1 < \ldots < t_N \leq T$ the transition moments of the process $\{\tilde{\xi}_t\}_{t=0,\ldots,T}$. Let $Z_t = \sum_{n=0}^{N-1} t_n \cdot 1_{t_n \leq t < t_{n+1}} + t_N \cdot 1_{t_N \leq t \leq T}$, that is, a process which at the transition moments t_n, takes the values t_n themselves, and the value t_n is kept for any t such that $t_n \leq t < t_{n+1}$. Thus its set of states is $\{0, \ldots, T\}$.

At the moment t_n, for $n \in \{0, \ldots, N-1\}$ we take the decision $\tilde{D}_{n+1}$. We obtain the sojourn time $\tau_{n+1} = \tau_{n+1}(t_n, \widetilde{D}_{n+1})$, and that the next transition time $t_{n+1} = t_n + \tau_{n+1}$. We associate the reward

$$V(n+1) \sum_{k=l_{\min}(n+1)}^{L_{\max}(n+1)} k \Pr\left[\sum_{s=t_n}^{t_{n+1}-1} w(n+1, a_s(n+1)) = k\right]$$ for the review period $[t_n, t_{n+1})$.

For the period $[t_N, T]$ we define the reward associated as being zero. This definition is equivalent with saying that the process $\{Z_t\}_{t=0,T}$ stops at a random time moment, more precisely at the random moment t_N.

For $0 \leq n \leq N$, let $h_n = \{x_{t_0} = 0_{R^N}, \tilde{D}_1, x_{t_1} = (l(1,t_1), 0, \ldots, 0), \tilde{D}_2, x_{t_2} = (l(1,t_1), l(2,t_2), 0, \ldots, 0),$

$\ldots, x_{t_n} = (l(1,t_1), l(2,t_2), \ldots, l(n,t_n), 0, \ldots, 0)\}$ an admissible history of the process $\{\widetilde{\xi}_{t_n}\}_{n=0,\ldots,N}$.

Let $\lambda_n(h_n) = \left(t_0 = 0, \tilde{D}_1, t_1, \ldots, \tilde{D}_n, t_n\right)$. We can prove that the functions $(\lambda_n)_n$ form a sequence of sufficient statistics for the process $\{\tilde{\xi}_t\}_{t=0,T}$ because $D_n(h_n) = D_n(\lambda_n(h_n))$, i.e. the decision taken at the transition moment t_n depends only on t_n, because we know that at the transition moment t_n the engineers start working on the design task number n. They may continue working until at most the time moment T_{n+1}.

We also have $\overline{r}_{n+1}\left(h_n, \tilde{D}_{n+1}\right) = \overline{r}_{n+1}\left(t_n, \tilde{D}_{n+1}\right) = \overline{r}_n\left(\lambda_n(h_n), \widetilde{D}_{n+1}\right)$.

According to Theorem 97 (Theorem 6.0 pp. 37 from (Hinderer, 1970)), we have that the processes $\{\tilde{\xi}_t\}_{t=0,\ldots,T}$ and $\{Z_t\}_{t=0,\ldots,T}$ are equivalent. ■

All the previous results in this chapter were designed to reformulate the sequential NPD control problem in a way that helps us to characterize the "optimal behavior" of the process. This is done in the next result, where we prove that the policies induced by the actions $\tilde{a}_s(n+1) = \min(L_{\max}(n+1), a_{s,\max}(n+1))$ exhibit an optimal behavior.

Proposition 99 *Denote by* Π *the set of policies for the process* $\{\tilde{\xi}_t\}_{t=0,\ldots,T}$*, and by*

$$\hat{\Pi} = \left\{\hat{\pi} = (\hat{\pi}_n)_{n=0,\ldots,N-1} \,|\, \hat{\pi}_n(x_{t_n}, t_n) = \hat{\pi}_n(t_n) = \left(D_{n+1}, \tilde{a}^{n+1}_{[t_n, T_{n+1}]}\right)\right\}$$

where $\tilde{a}^{n+1}_{[t_n,T_{n+1}]} = \left(\tilde{a}_{t_n}(n+1), \tilde{a}_{t_n+1}(n+1), \ldots, \tilde{a}_{T_{n+1}-1}(n+1)\right)$*, with* $\tilde{a}_s(n+1) = \min(L_{\max}(n+1), a_{s,\max}(n+1))$ *and* $x_{t_n} = (l(1,t_n), l(2,t_n), \ldots, l(n,t_n), 0, \ldots, 0)$.

Then, for any policy $\pi \in \Pi$ *of the process* $\{\tilde{\xi}_t\}_{t=0,\ldots,T}$*, there exists* $\hat{\pi} \in \hat{\Pi}$ *with the property that the total expected reward of* $\hat{\pi}$ *is greater or equal than the total expected reward of* π.

Proof. Denote by $\tilde{D}^{\max}_{n+1} = \left(D_{n+1}, \tilde{a}^n_{[t_n,T_{n+1}]}\right)$, by $\tilde{D}_{n+1} = \left(D_{n+1}, a^n_{[t_n,T_{n+1}]}\right)$, where $a^{n+1}_{[t_n,T_{n+1}]} = \left(a_{t_n}(n+1), a_{t_n+1}(n+1), \ldots, a_{T_{n+1}-1}(n+1)\right)$, with $a_s(n+1)$ between 0 and $\tilde{a}_s(n+1)$.

Let $\pi \in \Pi$, $\pi = \{\pi_n\}_{n\in\{0,1,..N-1\}}$, $\pi_n(x_{t_n}, t_n) = \pi_n(t_n) = \tilde{D}_{n+1} = \left(D_{n+1}, a^{n+1}_{[t_n,T_{n+1}]}\right)$, and $\widehat{\pi} \in \hat{\Pi}$, $\widehat{\pi} = \{\widehat{\pi}_n\}_{n\in\{0,1,..N-1\}}$, $\widehat{\pi}_n(x_{t_n}, t_n) = \widehat{\pi}_n(t_n) = \tilde{D}_{n+1} = \left(D_{n+1}, \tilde{a}^{n+1}_{[t_n,T_{n+1}]}\right)$ two policies having for any sample paths of the process $\{\tilde{\xi}_t\}_{t=0,\ldots,T}$ the same sequence $(D_1, \ldots, D_N)$.

We prove, by using forward induction for n, that for any pair $\pi \in \Pi$, $\hat{\pi} \in \hat{\Pi}$.with the above property the total expected reward of $\hat{\pi}$ is greater or equal than the total expected reward of π.

In this proof we will use the second way of reducing the reward structure. The choice of how to reduce the reward structure is given by how easily one can write the expected values defining the rewards.

Let $\pi \in \Pi$. The verification step of the induction holds for $n = 0$, because the reward of the first review period is zero no matter the decision taken at the transition moment t_0.

Let $\tilde{R}_1$ be the reward for the first two review periods. We have that

$\tilde{R}_1\left(0,t_1,\tilde{D}_1,x_{t_1}\right)=\tilde{r}_1\left(0,t_1,\tilde{D}_1,x_{t_1}\right)=\tilde{r}_1\left(0,t_1,\pi_0(t_0,x_0),x_{t_1}\right)\overset{not}{=}\widehat{r}_{1\pi}\left(x_0,t_1,x_{t_1}\right)$, where $\tilde{D}_1=\pi_0(t_0,x_0)$. Let $p_{t_0,x_0,\pi}(t_1)$ the probability that first transition moment is t_1, if we use the policy π, and $p_{x_0,t_0,t_1\pi}(x_{t_1})$ the probability that the state at the first transition moment is x_{t_1}, knowing that the first transition moment was t_1, that the state at the previous moment was $x_0=0_{\mathbb{R}^N}$, and that we use the policy π.

We notice that $p_{t_0x_0,\pi}(t_1,x_{t_1})=p_{t_0,x_0,\pi}(t_1)\cdot p_{t_0,x_0,t_1,\pi}(x_{t_1})$ where $p_{t_0x_0,\pi}(t_1,x_{t_1})$ is the probability that given the policy π the first transition moment equals t_1 and that the state of the system at the first transition moment is x_{t_1}.

Then the expected value of $\tilde{R}_1$, let it $R_{1\pi}(0)$ is

$$R_{1\pi}(0)=\sum_{(t_1,x_{t_1})}\widetilde{r}_{1\pi}\left(x_0,t_1,x_{t_1}\right)p_{t_0x_0,\pi}(t_1,x_{t_1})=$$

$$=\sum_{t_1}\sum_{x_{t_1}\in X(t_1)}p_{t_0,x_0,\pi}(t_1)\cdot p_{x_0,t_0,t_1\pi}(x_{t_1})\widetilde{r}_{1\pi}\left(x_0,t_1,x_{t_1}\right)=$$

$$=V(1)\sum_{t_1}p_{t_0,x_0,\pi}(t_1)\cdot\sum_{k=l_{\min}(1)}^{L_{\max}(1)}k\cdot\Pr\left(\sum_{s=0}^{t_1-1}w\left(1,a_s\left(1\right)\right)=k\right)$$

where $0\leq w\left(1,a_s\left(1\right)\right)\leq\min\left\{a_s\left(1\right),L_{\max}\left(1\right)\right\}$.and $k=l\left(1,t_1\right)=x_{t_1}(1)$.

We observe that $p_{x_0,t_0,t_1\pi}(x_{t_1})=p_{t_1}(x_0,\widehat{D}_1,x_{t_1})=$

$$=\Pr\left(\sum_{s=0}^{t_1-1}w\left(1,a_s\left(1\right)\right)=l\left(1,t_1\right)=x_{t_1}(1)\right)\leq\Pr\left(\sum_{s=0}^{t_1-1}w\left(1,a_{s,\max}\left(1\right)\right)=l\left(1,t_1\right)\right)=$$

$=p_{t_1}(x_0,\widehat{D}_1^{\max},x_{t_1})=p_{x_0,t_0,t_1\widetilde{\pi}}(x_{t_1})$ because the number of possible combinations of elements w having the sum equal to $l\left(1,t_1\right)=x_{t_1}(1)$ is greater for $a_{\max,s}\left(1\right)$ and all possible combinations for $a_s\left(1\right)$ are also among the ones for $a_{s,\max}\left(1\right)$.

We remark that for any $t_1$value

$p_{t_0,x_0,\pi}(t_1)=p_{t_0}\left(x_0,t_1,\tilde{D}_1\right)\leq p_{t_0}\left(x_0,t_1,\tilde{D}_1^{\max}\right)=p_{t_0,x_0,\widetilde{\pi}}(t_1)$, because the probability that the sojourn time for the state x_0 (i.e. the time while the engineers are working for the design task 1) be equal to a given value is the probability of all combinations of elements w which lead to the realization of the level $l_{\min}\left(1\right)$ or of the level $L_{\max}\left(1\right)$.

We are now ready to prove the induction step. Suppose the property is true for the first n review periods, and let us prove this is also true for $n+1$. Consider a policy $\pi\in\Pi$.

Let $R_{n,\pi}\left(0\right)$ be the expected value of the reward for the first $n+1$ review periods, and let $\tilde{R}_{n,\pi}\left(0,t_n,x_{t_n}\right)$ be the expected value of the reward for the first $n+1$ review periods if the $n-$th transition moment is t_n, the state at that moment is x_{t_n}, and the policy π is used. We observe that $R_{n,\pi}\left(0\right)$ is the expected value of $\tilde{R}_{n,\pi}\left(0,t_n,x_{t_n}\right)$ with respect to the variables t_n and x_{t_n}. Let $\tilde{R}_{n,\pi}\left(0,t_{n-1},x_{t_{n-1}},t_n,x_{t_n}\right)$ be the expected value of the reward for the first $n+1$ review periods if the $n-1-$th and the $n-$th transition moments are t_{n-1} and t_n, the states at those moments are $x_{t_{n-1}}$ and x_{t_n}, and the policy π is used. Hence

$$\tilde{R}_{n,\pi}\left(0,t_n,x_{t_n}\right)=\sum_{\left(t_{n-1},x_{t_{n-1}}\right)}\tilde{R}_{n,\pi}\left(0,t_{n-1},x_{t_{n-1}},t_n,x_{t_n}\right)p_{t_{n-1},x_{t_{n-1}}\pi}(t_n,x_{t_n})=$$

$$=\sum_{t_{n-1}}\sum_{x_{t_{n-1}}}\tilde{R}_{n,\pi}\left(0,t_{n-1},x_{t_{n-1}},t_n,x_{t_n}\right)p_{t_{n-1},x_{t_{n-1}},\pi}(t_n)p_{t_n,x_{t_{n-1}},\pi}(x_{t_n})=$$

$$=\sum_{t_{n-1}}\sum_{x_{t_{n-1}}}[\tilde{R}_{n-1,\pi}\left(0,t_{n-1},x_{t_{n-1}}\right)+\tilde{r}_{n\pi}\left(t_{n-1},t_n,x_{t_n}\right)].p_{t_{n-1},x_{t_{n-1}},\pi}(t_n)p_{t_n,x_{t_{n-1}},\pi}(x_{t_n}).$$

Also, $R_{n,\pi}(0) = \sum_{(t_n, x_{t_n})} \tilde{R}_{n,\pi}(0, t_n, x_{t_n})\, p_{x_0,t_0,\pi}(t_n, x_{t_n}) =$

$= \sum_{(t_{n-1}, x_{t_{n-1}})} \tilde{R}_{n-1,\pi}\left(0, t_{n-1}, x_{t_{n-1}}\right) p_{x_0,t_0,\pi}(t_{n-1}, x_{t_{n-1}}) +$

$+ \sum_{(t_{n-1}, t_n, x_{t_n})} \tilde{r}_{n\pi}(t_{n-1}, t_n, x_{t_n}) \cdot p_{x_0,t_0,\pi}(t_{n-1}, t_n, x_{t_n}) =$

$= R_{n-1,\pi}(0) +$

$$+V(n)\sum_{t_{n-1}}\sum_{t_n}\sum_{k=l_{\min}(n)}^{L_{\max}(n)} k \cdot \Pr\left(\sum_{s=t_{n-1}}^{t_n-1} w(n, a_s(n)) = k\right) p_{x_0,t_0,\pi}(t_{n-1}) p_{t_{n-1},\pi}(t_n) =$$

$=R_{n-1,\pi}(0) +$

$$+V(n)\sum_{t_{n-1}}\sum_{\tau_n}\sum_{k=l_{\min}(n)}^{L_{\max}(n)} k \cdot \Pr\left(\sum_{s=t_{n-1}}^{t_n-1} w(n, a_s(n)) = k\right) p_{x_0,t_0,\pi}(t_{n-1}) p_{t_{n-1},\pi}(\tau_n) =$$

where $p_{t_{n-1},x_{t_{n-1},}\pi}(t_n, x_{t_n})$ is the probability that the $n-$th transition moment is t_n.and that the state corresponding to the $n-$th transition moment is x_{t_n} knowing that the $n-1-$th transition moment was t_{n-1} and that the state corresponding to the $n-1-$th transition moment was $x_{t_{n-1}}$; $p_{x_0,t_0,\pi}(t_{n-1}, x_{t_{n-1}})$ is the probability that $n-1-$th transition moment is t_{n-1} and that the state corresponding to the $n-1-$th transition moment is $x_{t_{n-1}}$;.$p_{x_0,t_0,\pi}(t_{n-1}, t_n, x_{t_n})$.is the probability that the $n-$th and the $n-1-th$ transition moments are t_n and respectively t_{n-1}.and that the state corresponding to the $n-$th transition moment is x_{t_n}; $p_{x_0,t_0,\pi}(t_{n-1})$ is the probability that $n-1-$th transition moment is t_{n-1}; and.$k = l(n, t_n) = x_{t_n}(n)$.

We remark that $p_{t_{n-1},t_n,\pi}(x_{t_n}) = \Pr\left(\sum_{s=t_{n-1}}^{t_n-1} w(n, a_s(n)) = k = x_{t_n}(n)\right)$.

Together the induction hypothesis for n and the verification step (for $n = 1$; n being the number of periods for which the property is proven) give us the desired property, concluding the proof. ■

Corollary 100 *Consider that the design tasks in the given sequence can be performed by the team of engineers one after the the other and they can start only at the beginning of a review period. Then, the optimal policies of the Markov decision process* $\{\tilde{\xi}_t\}_{t=0,\ldots,T}$*,belonging to* $\hat{\Pi}$ *(i.e. the optimal policies which decide only on how many review periods the team of engineers should work on each design task, instead of deciding also the levels up to which the team has to work on each of the design tasks) are optimal policies of the Markov sequential decision problem* S_0.

8.8 Conclusions

By imposing a probabilistic constraint on the sample paths of the process of sequentially completing design tasks through the completion of a minimal number of design activities, it can be shown that the optimal strategy belongs to a class of strategies determined by a set of stopping times. At each point in time, given the current state of the system, it is decided how long to proceed with the current design task. As this expires, one can decide

to stop the project with no pay-off or to proceed with either the same design task or the next one. In the mean time at each review point in time these stopping times are recalculated.

The final model required the definition of the following elements:

1. $T_1 \leq T_2 \leq ... \leq T_{N-1} \leq T_N = T$, latest stopping times. T_n represents the latest moment in time at which we have to stop working on the design task n, such that with the probability α we can still achieve the minimal performance levels of the design tasks $n+1,..,N$. The existence of the T_n moments allows us to reduce the set of actions (the S_1 model) actions which have at most one non-empty component, which will be decided at the T_n moments. In between two such moments the rank of the non-empty component remains the same.

2. $t_0 = 0 < t_1 < \ldots < t_N \leq T$, the transition moments. They can be determined since the transition probabilities of the stochastic dynamic decision model S_1 are coupled and their system equation is $x_{t+1} = x_t + y_t^{a_t}$ where $x_t = (l(1,t), ..., l(N,t))$ is the state of the system, $a_t = (a_t(1), ..., a_t(N))$ is an action in $A_t(x_t)$, such that $a_t(i) \leq \min\{L_{\max}(i), a_{t,\max}(i)\}$ and $y_t^{a_t} = (w_t(1, a_t(1)), ..., w_t(N, a_t(N)))$ is a random object with probability distribution parametrized only by the a_t such that $0 \leq w_t(i, a_t(i)) \leq a_t(i)$, at the time instant t.

   We proved that for any admissible history h_T of the stochastic dynamic decision model S_1 $(\exists)\ k \in \{1, \ldots, N\}$ and $0 = t_0 < t_1 < \ldots < t_k \leq T$, with $t_i \in \mathbb{Z}_+$, $t_i \leq T_{i+1}$, $i \in \{1, .., N-1\}$ and $t_N \leq T$ such that for any $t \in \{t_i, \ldots, t_{i+1} - 1\}$, $0 \leq i \leq k-1$ we have $y_t^{a_t} = (0, ..., 0, w_t(i+1, a_t(i+1)) \neq 0, 0, ..., 0)$ and for any $t \in \{t_k, \ldots, T\}$ we have $y_t^{a_t} = 0_{\mathbb{R}^N}$.

3. $D_i \in \{1, \ldots, T_i - t_{i-1}\}$, $i \in \{1, \ldots N\}$ such that being given an admissible history h_T of the stochastic dynamic decision model S_1we have $t_i = t_{i-1} + \tau_i$, where $\tau_i = \min\left\{T_i - t_{i-1}, \max\left(\sigma_i^1, D_i\right), \min\left(\sigma_i^2, D_i\right)\right\}$, with

   $$\sigma_i^1 = \min\left\{t \in \mathbb{Z}_+^* \,\middle|\, \sum_{s=1}^{t} y_{t_{i-1}+s-1}^{a_{t_{i-1}+s-1}(i)} \geq l_{\min}(i)\right\} \text{ and}$$

   $$\sigma_i^2 = \min\left\{t \in \mathbb{Z}_+^* \,\middle|\, \sum_{s=1}^{t} y_{t_{i-1}+s-1}^{a_{t_{i-1}+s-1}(i)} \geq L_{\max}(i)\right\}.$$

4. $\widehat{D}_{n+1} := \left(D_{n+1}, \tilde{a}_{[t_n, T_{n+1}]}^{n+1}\right)$, where $D_{n+1} \in \{1, \ldots, T_{n+1} - t_n\}$ and

   $\tilde{a}_{[t_n, T_{n+1}]}^{n+1} = \left(\tilde{a}_{t_n}(n+1), \tilde{a}_{t_n+1}(n+1), \ldots, \tilde{a}_{T_{n+1}-1}(n+1)\right)$, with

   $\tilde{a}_s(n+1) = \min\left(L_{\max}(n+1), a_{s,\max}(n+1)\right)$

With these elements we define $Z_t = \sum_{n=0}^{N-1} t_n \cdot 1_{t_n \leq t < t_{n+1}} + t_N \cdot 1_{t_N \leq t \leq T}$, that is, a process which at the transition moments t_n, takes the values t_n themselves, the value t_n being kept during the interval $t_n \leq t < t_{n+1}$ Thus, its set of states is $\{0, \ldots, T\}$. At the moment t_n, for $n \in \{0, \ldots, N-1\}$ we take the decision $\widehat{D}_{n+1}$.

The process $\{Z_t\}_{t=0,T}$ is a process with a random end, namely t_N. The transition mechanism is (i.e. the state of the process at transition time t_{n+1}) is $t_{n+1} = t_n + \tau_{n+1}$ where

$\tau_{n+1} = \tau_{n+1}(t_n, \widehat{D}_{n+1})$ is the sojourn time. Thus the transition probabilities are determined by the sojourn times.

We associate the reward $V(n+1) \sum_{k=l_{\min}(n+1)}^{L_{\max}(n+1)} k \Pr\left[\sum_{s=t_n}^{t_{n+1}-1} w\left(n+1, a_s\left(n+1\right)\right) = k\right]$ for the review period $[t_n, t_{n+1})$. For the review period $[t_N, T]$ the reward will be zero.

The first implication of this results is that the we reduced a multidimensional Markov decision process to an unidimensional one, by partially identifying the optimal policy. Moreover, this characteristic of the optimal strategies can be described in a relatively easy way and it is likely that given the structure of these optimal strategies, that these policies can be computed numerically in a more efficient way than by applying some generic DP algorithm.

Chapter 9

Conclusions and Further Research

9.1 Overview

In this dissertation we have studied planning and control methods for a new type of projects called time-constrained New Product Development (NPD) projects with high technological product or process uncertainty. The engineers in the development team have to quickly design a new product, starting from loose specifications, which evolve until the project deadline. The product specifications can be broken down via a specification tree into design tasks, and these design tasks have to be solved by the team. The design tasks are subject to precedence relationships. Later in time, design reviews are needed and more design tasks may emerge.

These projects have gained increased momentum in recent years, as the economical and technological environment of many firms underwent significant changes, calling for shorter development times, and more flexible organizing guidelines and methods. Although the modern research in general planning and control is at least half-a-century old, stemming from NASA's Program Evaluation and Review Technique, new methods have to be devised. Recent empirical research shows that one of the main characteristics significantly differentiating this type of NPD projects from more classical ones is the large amount of uncertainty, both market and technology related, which requires to be intimately taken into account in planning and control procedures.

In the extensive literature that we have surveyed, the NPD models, which allow the product specifications to be gradually reconsidered, were dealing with the market uncertainty and not with the technological uncertainty appearing inside the firm as the result of its own innovation process. One of the contributions of this dissertation is to incorporate technological uncertainty as well, through a very general framework. This setting consists of formal definitions and axioms which we propose, based on numerous surveys and empirical studies closely monitoring the design planning and control in many firms. This model thus brings measurable variables into play in order to accurately account for the uncertainties and the quality control. Moreover, our model also integrates some human aspects, through the probabilistic dependency of engineers' behavior of their perceived time-to-the-due-date pressure, as well as through their ability to take concurrency and relative urgency of different design tasks into account. Since, on the one hand, the product specifications play a central role by the nature of the problem, and on the other hand, for practical purposes, managers would like

to see a dependency on expected market value of the new product, we have connected these two notions in the model. Finally, the model can also serve as a basis for further addition of constraints, for close tune-up to different situations encountered in real-life.

The second part of the dissertation builds upon the modelling stage, presenting concrete analytical problems and solutions as theorems and algorithms with provable properties. These formulations and solutions are, to the best of our bibliographical knowledge, new, and they focus on concurrent (i.e. without precedence constraints) NPD's, as well as on sequential NPD's. We use discrete-time finite horizon non-stationary Markov decision processes, and give results concerning their optimal policies. Several of these results also extend what is called the monotonicity theory (of these optimal policies) and also the partial order programming techniques to bounded subsets without holes of integer vector lattices.

Finally, we have used computer simulations to further gain insight in the interrelationship of various parameters, and to pave the way for devising heuristic optimal control policies.

In the remaining of this chapter we shall, in a more in-depth manner, review the characteristics of the model and thereafter the details of the analytical problems, tools and solutions, closing with a few directions which could be pursued in future studies.

9.2 Modelling proposals

For these NPD projects we distinguish three *overlapping* phases: the *system design/concept development* (performing a first work breakdown from customer needs into product specifications, and from product specifications into design tasks), the *detailed design phase* **(**consisting of solving the design tasks**)**, and the *system level test* (integrating the solved design tasks result into a complete system and tested). Since the first and last phases are too problem dependent to be modelled in a general mathematical framework, we focus on *the management of the time and resources* in the control of the evolution of the new product specifications, specifically during the *detailed design phase*. The overlap of the three phases translates the very influence of uncertainty during the design process, and *by allowing the update of the product specifications at regular intervals*, our model allows the product definition to evolve after the beginning of the detailed design phase.

Among given inputs we have the project delivery time, as well as the available resources (a finite number of engineers). The NPD project can be described at any time moment by *a network of design tasks. Each design task* has a number of *increasing performance levels* giving the quality of its execution. Each performance level requires the execution of a *list of planned activities*. For each design task a *minimal performance level* has to be achieved in order to obtain a fully functional new product. We use these performance levels as decision variables. Thus, the new product is dynamically redefined until the deadline We integrate in the model both the market requirement uncertainties and the technological uncertainties, and make explicit the trade-offs in the new product definition in terms of which design tasks, and up to which performance level should be performed until the NPD deadline.

The proposed control model of the NPD project is a discrete time one. The project will be reviewed at equidistant points in time until the deadline, T. The hierarchical structure proposed for each review period corresponds to a decision-scheduling-execution cycle, and consists of three levels: *aggregate decision level, detailed planning level* (rescheduling decision, scheduling), and *execution/engineering level*. At the aggregate decision level the

performance levels of the design tasks are controlled, while at the detailed planning level a nonpreemptive schedule of design tasks is obtained for a relatively short planning horizon. After updating the design task network structure, the project management may decide to decrease/increase the performance levels of some of the already scheduled design tasks, but not finished yet, due to the addition/deletion of design tasks and the limited capacity available versus their stochastic solving time. Then the design tasks and/or the number of their planned activities is changed. Since a feasible schedule is obtained by means of stochastic ordering, re-scheduling may occur.

The defined model elements are rich enough to incorporate the available knowledge from the relevant fields such as new product management and systems engineering, and still allow at the detailed and aggregate decision level for the mathematical analysis of the process. Some of the problems constructed were new: the aggregate decision problem and the problem of allocating concurrent NPD design tasks to the engineers, at the detailed planning level. The dissertation provides analytical solutions for them.

9.3 Analytical results

We now move onto the mathematical formulation and solving of the new problems. We concentrate first on the design task allocation problem, and then on optimal policies for the aggregate decision problem.

9.3.1 Design task allocation

We state in a mathematical way the allocation problem and prove that it can be formulated as a discrete deterministic dynamic-programming (DP) problem, linked to multiple choice knapsack problems.

The DP formulation of our allocation problem ensures the existence of an optimal solution; also, through a graph structure of the problem space, it also allows the finding of an optimal solution without an exhaustive search of the entire problem space. This graph structure can actually be searched efficiently with heuristic search algorithms. Using aggregate information (from the input data) on the design tasks set, a heuristic evaluation function of A* type is constructed to perform such a search. Based on established results from heuristic search algorithms, we propose an A* type algorithm for solving the problem, with two possible implementations: standard best-first search (open and closed priority lists, and RBFS implementation (Korf, 1993). It is proven that owing to the properties of the proposed heuristic, our algorithm is guaranteed to find an optimal allocation of the design tasks to engineers. For the implementations there is a complexity trade-off: the RBFS implementation runs in linear space in the branching factor and the depth of the tree, and in linear time in the number of generated nodes, at the expense of revisiting some nodes. The standard best-first search implementation on the other hand is proven to expand only a minimal number of nodes (with respect to any other algorithm from its class), yet the spatial complexity may be exponential: the branching factor elevated to the depth. Experimental evidence of the tests shows however that the RBFS solution revisits very few more nodes (having thus a small supplemental running time with respect to the standard best-first), and the total number of visited nodes is very small compared to the cardinality of the search space (second kind Stirling numbers).

9.3.2 Aggregate planning decision overview

Based on lower levels (i.e. the detailed planning and the engineering level) mathematical description, the model review periods were linked into a multi-period aggregate decision problem. We first introduced simple heuristics for both the engineering and the detailed planning processes, and we afterwards constructed a queueing model to estimate the solving time distribution of design tasks in NPD projects. The shape of the distribution function obtained via this model is similar to the one from data collected in the recent empirical research on the time-to-repair distribution functions in manufacturing systems, which suggest a long tailed, skewed, multi-modal distribution function. For the case of only one engineer, we tested with a goodness-of-fit test if the eight data sets collected in a very recent empirical research from an advanced micro-lithography systems firm can be derived from our model cumulative distribution function. Based on the results of the Kolmogorov-Smirnov goodness-of-fit test, the model showed no statistically significant difference with respect to data sets considered.

Later, the use of this queueing model allows the computation of transition probabilities for a non-stationary Markovian decision process model of the aggregate decision problem in a multi-period setting. This process focuses on the NPD technological uncertainty (i.e we take into account only the market payoff function values available at the deadline T of the NPD project) and supports the dynamic achievement of the new product definition taking into account a high technological uncertainty that affects the content of the project design tasks. The type of information used is in line with the design approach to hierarchical production control systems: at the highest decision level we obtain only an aggregate plan based on the estimation of the time and capacity needed, leaving the lower planning levels to elaborate detailed plans for subnetworks of the entire project, and only for a review period horizon. This is more efficient, because due to the high technological uncertainties in the time-constrained new product development, a detailed plan for the entire NPD horizon (i.e. from $t = 0$ to T) might be completely changed at each review period.

The principal drawback of a Markov model is the tendency for the underlying state space to grow explosively with respect to the size of the system. The final goal was to enable a more efficient computation of optimal policies in the aggregate decision process, as well as the characterization of its optimal policies. Thus, in Chapter 7 and Chapter 8 we investigated the structural properties of the optimal policies in two particular cases of the general aggregate decision problem. The first case considers an NPD project without precedence constraints, and with stable market conditions. The second considers an NPD project consisting of a sequence of design tasks, and with stable market conditions as well.

9.3.3 Concurrent NPD aggregate decision planning overview

In Chapter 7 we focus on a concurrent NPD (i.e. without precedence constraints), consisting of N concurrent design tasks and described by a discrete-time, finite horizon non-stationary Markov decision process. For enabling a more efficient computation of optimal policies in Markov models of sequential decision processes, one is often interested in finding structured policies (monotonic, convex, etc.). When monotonic policies can be found, a monotonic backwards induction algorithm can be used. It is known that in general dynamic programming problems often the solutions are too complicated to be used for deriving insights about the given problem. Moreover, in general, in higher dimensions the monotonicity of the decision rules does not imply a stable structure of the optimal paths (i.e. an increase in

an exogenous parameter will imply a uniform increase of the optimal path). In the existing literature it is proven that the existence of monotonic policies leads to robust optimal paths, under several assumptions including that the action spaces are sublattices of $\mathbb{R}^m$ for any state. This lattice type of action set is required also by the monotonic backwards induction algorithm. However, we show in Chapter 6 that such a strong condition (of having lattices) can in real-life eliminate all the possible computational gains that can be obtained from the existence of monotonic optimal policies. Thus we have extended the existing monotonicity theory to take into account this fact.

9.3.4 Extension of monotonicity theory

In Chapter 6 we start by assuming no lattice constraint. Because due to the requirements of the lattice notion, the derived backwards induction algorithms become completely infeasible we tried to relax this notion in such a way that we can still take advantage of the fact that the Markov processes under study had partially ordered state and action spaces. When reviewing the well-established theory in the literature, we only found results dealing with lattices. We thus extended the setting, to namely deal with fragments of lattices: bounded subsets without holes of integer lattices. We derived the general conditions of obtaining instead of monotonic optimal policies, weakly monotonic optimal policies. This has lead to a new weakly-monotonic backwards induction algorithm, and we also have been able to prove, in this chapter, that our results exhibit this sought-after property of robustness.

These results are general and they might be applied to other types of problems, especially since partially ordered structures have been recognized as being important in many fields from mathematics to biology, economics and also physics. In economics, the partial orderings, doubled by the lattice programming techniques of (Topkis, 1998), encompass many applications in many production planning models ((Hopenhayn and Prescott, 1992; Garcia and Smith, 2000) for discrete-time production planning with stochastic demand, (Athey, 2002), and (Athey and Schmutzler, 1995) for the analysis of several attributes of a firm's short-run innovative activity). In physics, as pointed out in (Landsberg and Friedman, 1996), the presence of partial orderings restrains significantly the behavior, yet it allows for interesting trajectories and even chaotic ones. However, due to special consequences, the chaotic behavior is unstable and not present for most initial conditions when physically observing the system in the long term. This makes partially ordered dynamical systems interesting and rich of possibilities. Moreover, since these studies emphasize the passive aspect, that is observing the system and predicting its behavior, now, with results such as the ones presented in this chapter, the presence of partial orderings can also benefit to the control of such systems.

9.3.5 Concurrent NPD aggregate decision planning theoretical results

The NPD related results are the optimality of monotonic (in the partial order on the state space) policies in the case of a simplified workload constraint, and respectively weakly monotonic otherwise. To achieve that we formulate our control problem in a dynamic programming setting, and we try to establish the supermodularity of the objective function. In microeconomics, and in theories of production and consumer choice, supermodularity of an utility function is equivalent to products being *substitutes*. The substitute notion appears very intuitive in the case of a concurrent NPD situation, where after we finish all the design

tasks at their minimal performance levels, having done more levels of performance for one design task compensates for doing less performance levels of a different one.

The existence of the first type of monotonic nondecreasing optimal policies confirms an intuitive "greedy" feature of the optimal control policy "The performance requirement of a controller increases with the number of performance levels already achieved by the engineers". The existence of the weakly nondecreasing optimal control policies refines the early heuristic. According to them, the NPD aggregate controller may take actions which are not directly comparable (neither increasing, nor decreasing) from the point of view of performance levels required. The actions are only comparable from the potential cost/reward point of view, even if the states in which these decisions are taken are comparable from the performance levels achieved point of view. The second type of optimal policy leads to significant improvements in the computational efficiency as one can see from the weakly monotonic backward induction algorithm.

Finally, we prove that the weakly monotonic optimal paths are robust to the variation of the safety margin for achieving the new product at the deadline.

9.3.6 Concurrent NPD aggregate decision planning simulation results

We used simulation studies to investigate the robustness of the weakly monotonic optimal paths with respect to the variation of the solving rate of design activities. The experiments were performed using synthetic data. For small variations of the solving rate of the design activities we noticed a relatively stable structure of the optimal paths (i.e. an increase in the solving rate leads to a uniform increase of the optimal path).

We also used simulation studies to investigate the variation of the optimal value as a function of the degree of specification of the characteristics of the new product at the beginning of the NPD. The experiments were performed using synthetic data. We observed a rather surprising effect. It seems that independently of the value of the solving rate of a design activity, we have an increase of the optimal value if the product is less specified at the beginning. However, this effect is less significant when the solving rate of one activity increases, i.e. the team of engineers is overall more performant.

9.3.7 Sequential NPD aggregate decision planning overview

In Chapter 8, we focus on a NPD project with precedence constraints, consisting of $N \leq T$ sequential design tasks, described by a a discrete-time, finite horizon non-stationary Markov decision process. We assume that the design tasks in the given sequence can be performed by the team one after the other, and they can only start at the beginning of a review period. The first design task may start in the first review period. No arrival of new design tasks takes place in the case of the sequential NPD project, and the final reward form has to be given by a linear (weighted additive) function. The methodology used to deal with the complex Markovian structure of the sequential NPD control process is the one of the sample path analysis.

9.3.8 Sequential NPD aggregate decision planning theoretical results

The sample path analysis technique aims at comparing sample path by sample path stochastic processes defined on a common probability space so that characteristics of the "op-

timal behavior" or even the "optimal behavior" can be identified. An important issue with respect to the sample path techniques is that not all the ways of describing the stochastic evolution of a system allow for a comparison between two given policies. Thus, the problem formulation should be carefully done, and particular care was taken in choosing an appropriate state description in Chapter 5. However this is not enough. To perform the sample path analysis we have to define in a very formal way the underlying probability space, the class of admissible actions at a certain time moment under a given history, and the reward function. Thus, in the rigorous framework of non-stationary models of (Hinderer, 1970) we identify our sequential NPD control problem as being a nonstationary stochastic dynamic decision model with stopping sets in the sense of (Hinderer, 1970).

Due to an implicit probabilistic constraint on the sample paths of the process of sequentially completing design tasks through the completion of a minimal number of design activities, it can be shown that the optimal strategy belongs to a class of strategies determined by a set of "latest stopping times" for the design tasks. By restricting the action space, eliminating the sample paths with zero final reward, splitting the final reward into rewards to the transition moments, and finally by restricting simultaneously the number of decision moments, decision sets and the state space we obtain a sequence of four reformulations of the initial sequential NPD control problem. Using the sample path analysis we prove that the optimal policies of all those reformulations are optimal policies for the initial problem. The last reformulation reduces the initial multidimensional control problem to a unidimensional one in both state and action space. In the new control problem the optimal policy will decide only on how many review periods the team of engineers should work on each design task, being optimal to always choose, while working on a design task, as decision the maximal performance level.

9.4 Generalizations and future directions suggestions

9.4.1 From the multidimensional setting to the unidimensional one

First of all, let us notice that is it possible to use the characterization of the optimal policies in those particular cases to reduce the general aggregation decision problem from a multidimensional one to a unidimensional one.

9.4.2 Guidelines for deriving heuristic optimal policies

Our goal here is to advance further in the direction of methods for finding optimal control policies, by devising and justifying as much as possible heuristics, based on the findings exposed before.

As discussed in Chapter 2, we assure that an NPD problem can be formulated for a short period of time as a general task network (i.e. an acyclic digraph with a source and a sink). Since we have devised techniques for dealing with parallel and respectively serial task networks, we can think along the line leading to simplifying an arbitrary acyclic task network in these terms. In the sequel, we shall describe a general method of breaking down the initial arbitrary network in subsets of tasks to be separately dealt with.

Recall that the engineers working in the development team are all assumed to solve any activity of any task with the same average solving rate (see Chapter 2, Section 2.3).

We will suppose as well that any engineer can handle any task. What we need is a way to transform the partial ordering of the tasks given by the initial network into a total and non-strict ordering: that is, the initial network allows for uncomparable tasks (but has no cycles), and we need to complete it such as any two tasks become comparable, one preceding the other ("less than") or both being executable at the same time ("equal"). Once we have achieved this, we separately take care of each group of "equal" tasks, in "increasing order". The equal tasks are dealt with by the parallel case method, and the sequence of these groups is dealt with by the sequential case method. This completion of the partial ordering can be achieved by any exact or heuristic method, and for this reason that method is a parameter of our procedure.

We outline below the details of the steps of the general procedure we propose here.

1. consider all the tasks having a solving time given by their maximum performance level
2. use a heuristic priority-rule R which proposes a list schedule for the engineers
3. group each subset of tasks having the same priority in the list into a so called "aggregated" task; if there is only one task with a certain priority value, then it forms an "aggregated" task by itself
4. derive an optimal policy for the sequential NPD control problem defined by the "aggregated" design tasks. Following the results on optimal control policies for the sequential NPD case from Chapter 8, there exist optimal policies which ask the team of engineers to perform all the design tasks at the maximal level and afterwards set for each "aggregated" design task an optimal number of periods on which the team should work on it. We call this number the optimal time horizon of the "aggregated" design task.
5. derive an optimal/near-optimal policy for the control of each set of concurrent design tasks which constitute an "aggregated" design task. The time horizon in which each of these reduced concurrent NPD problems has to be solved is given by the optimal time horizon of its corresponding "aggregated" design task.

The priority-rule R can have different objectives when building the list schedule. For example, it can minimize $E(C_{\max})$, or $\max E(C)$, or $E(\sum C)$. As (Neumann and Zimmerman, 1998) mention, these rules are based on several general insights: to minimize $E(C_{\max})$, somehow corresponds to minimizing idle time; to minimize $\max E(C)$, tasks with large expected completion times (caused by the precedence constraints) should be scheduled as early as possible; finally, to minimize $E(\sum C)$, tasks with small expected durations should be scheduled first.

We claim that the most appropriate objective for an NPD problem is the minimization of $E(C_{\max})$, because this is linked to the time-to-market deadline constraint, given the economic setting as explained in Chapter 2.

We can give two examples of possible priority-rules to use with our general procedure. Other rules can be found for example in (Neumann and Zimmerman, 1998).

The first example makes use of the distance d_n to the sink for any given task n: this distance simply gives the longest path (in terms of number of nodes) from the task to the sink. Formally,

$$d_n = \begin{cases} 0 & \text{if } n = \text{sink} \\ 1 + \max\limits_{j \in \text{succ}(n)} d_j & \text{otherwise.} \end{cases}$$

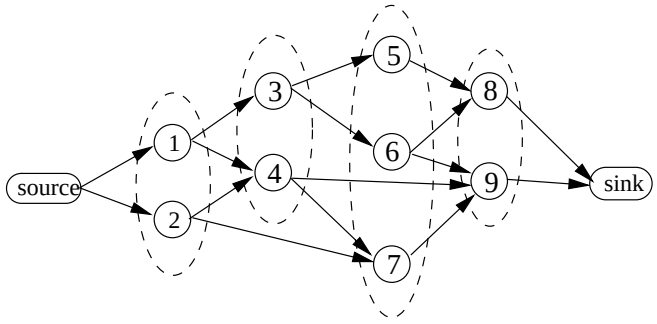

Figure 9.1. Example for priority-rule R_d

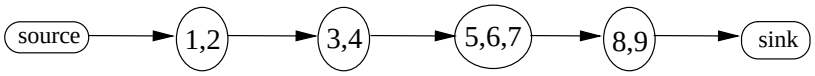

Figure 9.2. "Aggregated" task sets giving the sequential part for the example for priority-rule R_d started in Figure 9.1

The priority-rule R_d based on this distance orders the tasks according to their decreasing distance to the sink. Two tasks fall into the same "aggregated" task set if they have the same distance to the sink. Clearly two tasks such that one is successor of the other one cannot be found in the same "aggregated" task set. This ensures the correctness of the method.

A simple example illustrating the application of this rule can be given by the network of Figure 9.1.

Applying R_d we obtain the sequential problem having as "aggregated" tasks the following sets.

Each of these "aggregated" task sets then forms a parallel problem to be solved individually, as shown in Figure 9.2.

A further refinement to this rule can be made in a analogous way to the one of (Neumann and Zimmerman, 1998). First of all, the distance to the sink is changed so that it also takes into account the expected duration of each task (recall we have required the maximum performance level), formally giving

$$\overline{d}_n = \begin{cases} 0 & \text{if } n = \text{sink} \\ \max\limits_{j \in \text{succ}(n)} (\overline{d}_j + S_n(T, L_{\max}(n,T))) & \text{otherwise.} \end{cases}$$

We clearly also have here that two tasks such that one is the successor of another one are not being put into the same "aggregated" task set. In order to see why this is so, suppose the contrary. Let tasks n and p be such that $p \in \text{succ}(n)$ and that $\overline{d}_n = \overline{d}_p$. Then, according to the definition of $\overline{d}$, we have $\overline{d}_n \geq \overline{d}_p + S_n(T, L_{\max}(n,T))$, which amounts to $0 \geq S_n(T, L_{\max}(n,T))$, which would lead to void tasks: a clear contradiction.

The second part of the refinement separates tasks with the same $\overline{d}$, in decreasing order of their outdegrees $|\succ(.)|$. This rule, which further breaks down the "aggregated"

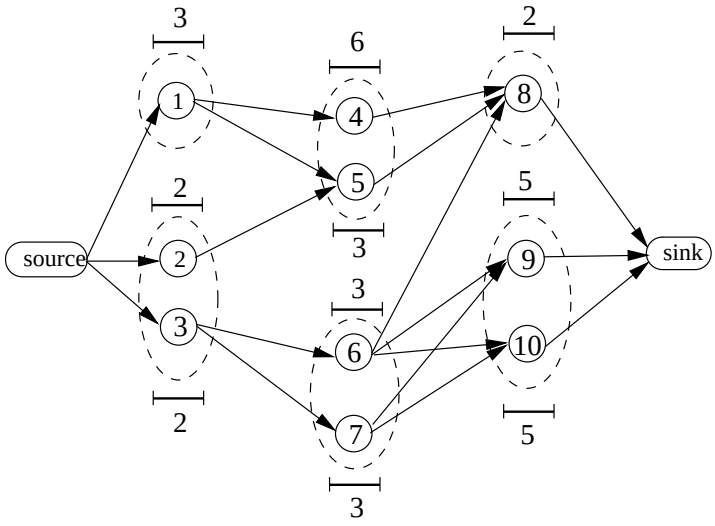

Figure 9.3. Example for priority-rule $R_{\overline{d}}$, with expected durations $S_n(T, L_{\max}(n,T))$ for each task

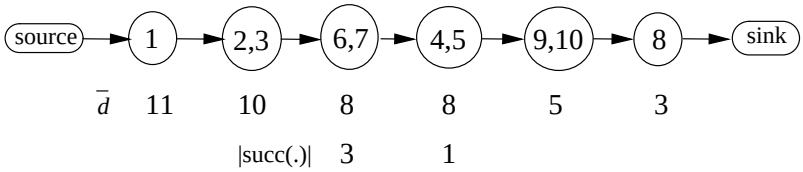

Figure 9.4. "Aggregated" task sets giving the sequential part for the example for priority-rule $R_{\overline{d}}$ started in Figure 9.3

tasks into smaller ones which are serialized, is to be applied only when the size of the initial "aggregated" task is greater than the number M of engineers.

Another example, for $M = 2$, illustrating the use of this improved rule is given by the network of Figure 9.3. We give also the expected durations $S_n(T, L_{\max}(n,T))$ near each task, to show how to compute the distance $\overline{d}$.

Applying $R_{\overline{d}}$ we obtain the sequential problem having as "aggregated" tasks the sets of Figure 9.4. Note that tasks 6, 7, 4 and 5 all have $\overline{d}$ equal to 8, but the first two have a greater outdegree $|\succ(.)|$, and thus are separated and come first. However, even though the tasks 2 and 3, which have the same $\overline{d}$ equal to 10, have different outdegrees $|\succ(.)|$, they still stay in the same "aggregated" task, since its cardinality is less than or equal to $M = 2$.

As for the previous example, each of these "aggregated" task sets then forms a parallel problem to be solved individually.

The next step in improving our general procedure is finding heuristics for near optimal policies for the particular cases of parallel respectively sequential orderings.

In the parallel case, following the results in Chapter 7, we see that the behavior of the optimal policy does indeed seem to have some greedy features. In general, the proven

existence of monotonic optimal policies usually leads to the optimality of such a greedy behavior. The reason is that greedy algorithms are less ambitious than non-greedy ones, in that they continuously look for a better solution, instead of looking for the best one, while searching for a global optimum. In such a step-by-step approach, the monotonic properties are usually useful. Thus, more study can be done along this path, in several directions: more simulations can give more confidence in the near optimality of such greedy myopic policies, and more theoretical investigation can lead, maybe with a few more model restrictions, to a formal proof of the optimality (or quantification of the near optimality) of the greedy myopic policies.

In the sequential case, the landscape looks more complicated. We have shown that the general multidimensional problem can be reduced to a uni-dimensional one; the next step would then be to construct a way of computing the transition probabilities in the uni-dimensional case corresponding to any given multidimensional sequential problem. Then simulation studies can be performed in order to give more insight about the structure of the optimal policies, and appropriate heuristics can be tested against the theoretical optimal solution. Thereafter more analytical studies might help consolidate these heuristics, assessing in a more formal way their performance characteristics.

9.4.3 Open analytical problems

Further developments of the concepts of weak monotonicity can include their application to other problems than the ones from the NPD framework. We namely suggest the characterization of other families of Operational Research problems, which on the one hand do not fulfill the necessary conditions to have optimal policies more structured than the weakly-monotonic ones, but for which on the other hand the existence of weakly-monotonic optimal policies could be a first step in devising or justifying the usage of appropriate greedy-type algorithms.

Specifically concerning the NPD problems, it might be interesting to try to assess in an analytical way whether the weakly monotonic optimal paths are robust to the variation of the solving rate of the design activities, and which are the relations between the optimal value and for instance the degree of specification of the characteristics of the new product at the beginning of the NPD. These studies could be helped perhaps by adding more real-life related constraints to the model, thus narrowing the domain of possible dependency families.

Samenvatting (Dutch Summary)

In dit proefschrift leveren we een aantal bijdragen aan het vinden van analytische oplossingen voor wiskundige geformuleerde problemen die onderdeel zijn van een hiërarchische model van de beheersing van productontwikkelingsprojecten (NPD-projecten). De NPD-projecten kenmerken zich door een hoge technologische onzekerheid en een korte, harde levertijd. Gebruikmakend van de resultaten van recent empirisch onderzoek naar ontwikkelprojecten stellen we eerst een hiërarchisch dynamisch model op van de beheersing van capaciteittoewijzing aan ontwerptaken: ontwerptaken hebben betrekking op het omzetten van functionele productspecificaties in technische productspecificaties. Het ontwikkelproject bestaat uit een netwerk van ontwerptaken. Functionele productkenmerken worden gerealiseerd door het uitvoeren van een of meer ontwerptaken. Een ontwerptaak bestaat uit een aantal ontwerpactiviteiten; naarmate de eisen gesteld aan het ontwerp stijgen, stijgt het aantal ontwerpactiviteiten van een ontwerptaak. De levertijd, T, en de beschikbare ontwikkelcapaciteit, bestaande uit een team van M engineers, worden als gegeven beschouwd. De beheersing reageert op zowel de stochastische voortgang van het ontwikkelproject zelf, als de stochastische ontwikkeling van de markteisen, door aanpassing van de functionele productspecificaties gedurende het ontwikkelproject. Hierbij wordt een expliciete afweging gemaakt tussen de verschillende dimensies van de functionele productspecificaties, in het licht van de marktwaarde van de functionele productkenmerken, en van de nog beschikbare capaciteit tot de project deadline. Het ontwikkelde beheersingsmodel is geformuleerd in discrete tijdsperiodes. De projectstatus wordt met vaste tijdsintervallen beoordeeld tot aan de deadline T. Na elk tijdsinterval wordt een hiërarchische beslissingscyclus doorlopen, bestaande uit twee niveaus: het aggregaatbeslisniveau, en het detailbeslisniveau, en gedurende elk tijdsinterval worden ontwerptaken uitgevoerd. Op aggregaatniveau worden de ontwerpeisen gesteld aan de ontwerptaken beheerst; op detailniveau wordt een non-preemptive schedule van ontwerptaken opgesteld voor een relatief korte planningshorizon. Uitgaande van het netwerk van ontwerptaken en uitgaande van en de geboekte voortgang tot moment t, kan de projectmanager besluiten de ontwerpeisen gesteld aan geschedulde ontwerptaken te verhogen of te verlagen; dit met het oog op de nog beschikbare capaciteit tot de deadline, en de onzekerheid m.b.t. het aantal nog uit te voeren ontwerptaken en de benodigde capaciteit per ontwerptaak. Dit kan leiden tot een nieuw detailplan, aangezien detailplanning is gebaseerd op stochastische ordening van ontwerptaken. In het hiërarchische model is een aantal elementen uit de productmanagementliteratuur en de systems engineering literatuur verwerkt, en dat maakt het mogelijk om de beslisproblemen op aggregaat- en detailniveau wiskundig te analyseren. Sommige van deze problemen zijn nieuw.

Het detailplanningsprobleem wordt opgevat als een allocatieprobleem. We laten zien dat het kan worden geformuleerd als een deterministisch dynamisch programmeringsprob-

leem. Het allocatieprobleem is ook gerelateerd aan het multiple-choice knapsack-probleem. De DP formulering van het allocatieprobleem garandeert het bestaan van een optimale oplossing: het maakt het ook mogelijk om een optimale oplossing te vinden zonder de gehele oplossingsruimte af te zoeken. Gebruikmakend van geaggregeerde informatie over de verzameling ontwerptaken construeren we een heuristische evaluatiefunctie van het A* type. Uitgaande van bekende resultaten op het gebied van de heuristische zoekalgoritmen, stellen we een A* type algoritme voor, met twee mogelijke implementaties: standaard best-first search (open en gesloten prioriteitslijsten) en RBFS implementatie. We bewijzen dat ons algoritme een optimale allocatie van ontwerptaken aan engineers vindt. Met betrekking tot de implementaties bestaat er een complexe afweging: de RBFS implementatie heeft een runtijd lineair met de vertakkingsfactor en met de diepte van de zoekboom, en lineair met het aantal gegenereerde knopen; echter de kans bestaat dat sommige knopen opnieuw geëvalueerd worden. Van de standaard best-first implementatie is bewezen dat het een minimaal aantal knopen ontwikkelt (relatief t.o.v. enig ander algoritme binnen haar klasse); echter de runtijd kan exponentieel toenemen (de vertakkingsfactor verheven tot een macht gelijk aan de diepte). Experimenteel onderzoek laat echter zien dat de RBFS implementatie slechts enkele knopen extra bezoekt (leidend tot een kleine extra runtijd vergeleken met standaard best-first) en dat het totale aantal bezochte knopen zeer klein is in vergelijking tot de kardinaliteit van de zoekruimte (Stirling getallen van de tweede soort).

De achtereenvolgende tijdsintervallen in het hiërarchische model zijn met elkaar verbonden via benaderende oplossingen voor het allocatieprobleem en voor de engineeringactiviteiten. De verbinding is gebaseerd op een eenvoudig wachtrijmodel waarmee de distributiefunctie geschat wordt van de tijdsduur die gemoeid is met het uitvoeren van ontwerptaken. De vorm van de distributiefunctie die uit dit model volgt vertoont overeenkomsten met de data die verkregen zijn uit recent empirisch onderzoek naar de time-to-repair distributiefunctie in productieprocessen; een scheve, multi modale distributiefunctie met een lange staart. Daarnaast hebben we ons model getest door vergelijking met acht verzamelingen van gegevens over de doorlooptijd van ontwerptaken, verzameld bij een ontwikkelbedrijf van geavanceerde lithografische systemen. Vergelijking van ons model met de data uit de acht gegevensverzamelingen op basis van de Kolmogorov-Smirnov goodness-of-fit test laat geen significante verschillen zien. Het verbindingsmechanisme tussen de achtereenvolgende tijdsintervallen maakt het mogelijk om de overgangskansen te berekenen voor het niet-stationaire Markov beslissingsmodel van het aggregaatprobleem. Dit levert een zeer algemeen raamwerk op dat het mogelijk maakt om on line op de voortgang te reageren, daarbij rekening houdend met de grote technologische onzekerheid zowel aan de marktkant als aan de ontwikkelkant. Het aldus verkregen raamwerk vertoont grote gelijkenis met bekende hiërarchische raamwerken voor productiebeheersing: op het hoogste niveau nemen we aggregaatbeslissingen, gebaseerd op schattingen van benodigde tijd en capaciteit, en het wordt aan de lagere niveaus overgelaten om over een korte horizon detailplannen uit te werken voor delen van het netwerk.

We zijn er slechts ten dele in geslaagd theoretische uitspraken te doen over de optimale oplossing van het aggregaatprobleem. De primaire oorzaak hiervan is gelegen in het feit dat de toestandsruimte voor het probleem explosief toeneemt als functie van de grootte van het probleem. Ons einddoel was om zowel een karakterisering te kunnen geven van de optimale policies, als om efficinte algoritmen te ontwikkelen. In de hoofdstukken 7 en 8 onderzoeken we de structurele eigenschappen van de optimale policies voor twee specifieke situaties. De eerste situatie bestaat uit een NPD project met ontwerptaken zonder onderlinge volgorde relaties (zuivere concurrency), in een stabiele markt. De tweede situatie bestaat

uit een NPD project met ontwerptaken die zuiver serieel moeten worden uitgevoerd (geen concurrency), ook in een stabiele markt. In hoofdstuk 7 beschouwen we een zuiver concurrent NPD project bestaande uit N ontwerptaken, gemodelleerd als een niet-stationair Markov beslisprobleem met een eindige horizon in discrete tijd. Voor de efficiënte bepaling van optimale policies in Markov-modellen van sequentiële beslisproblemen is men vaak geinteresseerd in het vinden van structurele eigenschappen (monotoniciteit, convexiteit, etc.). Voor monotone policies kunnen monotonic backward induction algoritmes worden gebruikt. Het is bekend dat de oplossingen van dynamische programmeringproblemen vaak zodanig complex zijn dat daaruit geen inzichten kunnen worden gedestilleerd m.b.t. de aard van het gegeven probleem. Bovendien geldt dat in hogere dimensies monotoniciteit van de beslissingspolicies niet inhoudt dat er een stabiele structuur is van de optimale paden (d.w.z. een verhoging van een exogene parameter houdt niet in dat er een uniforme verhoging is van het optimale pad). In de literatuur is bewezen dat onder bepaalde voorwaarden, waaronder de voorwaarde dat de actieruimtes voor elke toestand sublattices zijn van $\mathbb{R}^m$, het bestaan van monotone policies leidt tot robuuste optimale paden. Actieverzamelingen van dit lattice-type zijn ook vereist voor het gebruik van monotonic backward induction algoritmen. In hoofdstuk 6 wordt aangetoond dat deze sterke voorwaarde alle voordelen teniet kan doen die gepaard gaan met het bestaan van monotone optimale policies. Daarom laten we de lattice-voorwaarde vallen, en ontwikkelen we de algemene voorwaarden waaronder er zwak monotone policies bestaan. Dit leidt tot een nieuw weakly monotonic backward induction algoritme. We bewijzen dat onze resultaten leiden tot de gezochte robuustheids-eigenschappen. We tonen verder de optimaliteit aan van monotone policies onder de voorwaarde van een eenvoudige werklastbeperking (eerste type), en de optimaliteit van weakly monotonic policies in andere gevallen (twede type). We verkrijgen dit resultaat door het probleem te formuleren in een Dynamische Programmering context en door de supermodulariteit van de objectfunctie aan te tonen. In de micro-economie, in de productietheorie en, in de theorie rond consumentengedrag is supermodulariteit van een nutsfunctie equivalent met substitueerbaarheid van producten. Het idee van substitueerbaarheid heeft intuitief betekenis in een concurrent NPD project, waar, nadat alle ontwerptaken zijn gerealiseerd op het minimale vereiste niveau, het minder bereiken in de ene ontwerptaak (ten dele) gecompenseerd kan worden door meer te bereiken in andere ontwerptaken. Het bestaan van het eerste type monotoon niet-dalende optimale policies bevestigt de intuitieve notie dat de optimale policy ”greedy” is: de eisen die de beslisser stelt aan het ontwerpresultaat stijgen naarmate er meer bereikt is. Het bestaan van weakly nondecreasing optimal control verfijnt ons heuristiek inzicht. Volgens de weakly-monotoon intutieve notie kan het voorkomen dat er op aggregaatniveau acties mogelijk zijn die onderling niet direct vergelijkbaar zijn in termen van het geëiste ontwerpresultaat. De acties zijn alleen vergelijkbaar met betrekking tot de potentiële kosten en opbrengsten, zelfs als de toestanden van waaruit deze acties genomen kunnen worden wel onderling vergelijkbaar zijn met betrekking tot het al bereikte ontwerpresultaat. Zoals blijkt uit het weakly monotonic backward induction algoritme leidt het tweede type optimale policy tot substantile verbeteringen van de rekentijd. We hebben via simulatiestudies onderzocht hoe de optimale beslissing afhangt van de mate waarin de functionele productkenmerken aan het begin van het ontwerpproject gespecificeerd zijn. De productkenmerken noemen we onder (over) gespecificeerd als er relatief weinig (veel) ontwerpactiviteiten aan het begin van het project bekend zijn, en er relatief veel (weinig) ontwerpactiviteiten gedurende het ontwerpproject bijkomen. De simulatiestudie wees uit dat, onafhankelijk van de tijd nodig voor het uitvoeren van ontwerpactiviteiten, de optimale waarde van het vereiste ontwerpresultaat stijgt naarmate er meer

nieuwe activiteiten ”ontstaan” gedurende het project (en minder al activiteiten bekend zijn aan het begin van het project). Echter, dit verband zelf is zwakker naarmate de tijd nodig voor het realiseren van een ontwerpactiviteit kleiner is. In hoofdstuk 7 bestuderen we een zuiver serieel NPD project, bestaande uit $N \leq T$ sequentiële ontwerptaken, dat gemodelleerd wordt als eindige horizon, niet-stationair Markov beslissingsproces in discrete tijd. We nemen aan dat met de volgende ontwerptaak gestart kan worden, zodra de voorafgaande ontwerptaak klaar is. Aan de eerste ontwerptaak kan onmiddellijk bij aanvang van het project gestart worden, en er komen geen nieuwe ontwerptaken bij gedurende het project. De waarde van het ontwerpresultaat aan het einde van het project (tijdstip T) is een lineaire additieve functie van de bereikte resultaten per ontwerptaak. We maken gebruik van sample path analyse voor dit probleem. Sample path analyse vergelijkt de sample paden van stochastische processen die gedefinieerd zijn op een gemeenschappelijke kansruimte, teneinde kenmerken van het optimale gedrag, of het optimale gedrag zelf, te achterhalen. Een belangrijk probleem bij het gebruik van sample path analyse is dat de mogelijkheid om twee gegeven policies te kunnen vergelijken, afhangt van de manier waarop de stochastische ontwikkeling van het systeem beschreven wordt. De keuze voor de toestandbeschrijving van het ontwerpproject zoals weergegeven in hoofdstuk 8 is gebaseerd op deze overweging. Echter de toestandbeschrijving zelf is nog niet voldoende. Teneinde een sample path analyse te kunnen uitvoeren moeten de onderliggende kansruimte, de klasse van toegestane acties als functie van de tijd en de historie tot op dat moment, en de waardefunctie op formele wijze gedefinieerd worden. Gebruikmakend van het formele raamwerk voor niet-stationaire modellen, kenmerkt ons sequentieel NPD probleem zich als een niet-stationair dynamisch beslisprobleem met ”stopping sets”. De sample paden van het proces van achtereenvolgende uitgevoerde ontwerptaken met een minimaal aantal ontwerpactiviteiten per ontwerptaak zijn stochastisch begrensd. Op grond hiervan kan worden aangetoond dat de optimale strategie behoort tot de klasse van strategieën die gekenmerkt worden door een verzameling ”laatste stop tijdstippen” voor de ontwerptaken. Door vervolgens de actieruimte in te perken, sample paden die tot een eindwaarde van nul leiden te elimineren, de eindwaarde te relateren aan waardes op de beslismomenten, en gelijktijdig het aantal beslismomenten, de beslismogelijkheden en de toestandruimte verder in te perken, verkrijgen we een viertal herformuleringen van het initiële sequentiële NPD probleem. Gebruikmakend van sample path analyse bewijzen we dat de optimale policies voor elk van deze herformuleringen optimaal zijn voor het initiële probleem. De laatste van de vier herformuleringen reduceert het initiële multidimensionale beslisprobleem tot een eendimensionaal beslisprobleem, in zowel de toestandruimte als in de actieruimte. In het nieuwe beslisprobleem hoeft er enkel beslist te worden hoeveel achtereenvolgende perioden het team van engineers werkt aan elk van de ontwerptaken, waarbij ze, zolang aan een ontwerptaak werken, ernaar streven het maximale ontwerpresultaat te bereiken. In hoofdstuk 6 combineren we de resultaten uit de zuivere parallelle situatie en de zuivere sequentiële situatie teneinde het algemene aggregaatbeslisprobleem terug te brengen van een multi-dimensionaal probleem tot een eendimensionaal probleem. Verder geven we richtlijnen voor het ontwikkelen van goede heuristische beslispolicies voor het algemene aggregaatprobleem.

Appendix

Let $\Theta(n,t,\delta)$ be the *current maximal contribution of each design task n, in achieving the customer need δ*. As described in Subsection 2.3.3 *a time dependent design task contribution function in achieving the customer need* δ is given by $f(n,l(n,t),t) := \theta(n,\delta,t)\frac{l(n,t)}{L_{max}(n,t)}$.

At the detailed planning level, the criterion used in the allocation step is the one of maximizing the value of the concurrent allocated design tasks. Now, a good choice for the design task value functions, $V(\cdot,t)$ is one which leads to the maximization of the market payoff function, while maximizing the sum of design task values for the performance levels, $\hat{l}(\cdot,t)$, decided at the aggregate decision level.

We show below that in the case of the linear market payoff of (Askin and Dawson, 2000) one can find a formula describing $V(\cdot,t)$ such that maximizing the market payoff function (at the aggregate decision level) is exactly the same as maximizing the sum of design tasks value functions (at the detailed planning level). So in the linear case the coherence of objectives at the two control levels is exemplary. If (see (Askin and Dawson, 2000))

$$M\left[\left(\Theta(t,\cdot,\delta)_{1\leq\delta\leq\Delta}, \hat{l}(\cdot,t)\right)_{1\leq n\leq N}\right] \stackrel{def}{:=} \sum_{\delta=1}^{\Delta} w_\delta(t)\left[\sum_{n=1,..,N}\Theta(n,t,\delta)\cdot\frac{\hat{l}(n,t)}{L_{max}(n,t)}\right] \text{ then}$$

$$\max_{\hat{l}(n,t),\ n\in\{1,...,N\}} M\left[\left(\Theta(t,\cdot,\delta)_{1\leq\delta\leq\Delta}, l(\cdot,t)\right)_{1\leq n\leq N}\right] =$$

$$\max_{\hat{l}(n,t),\ n\in\{1,...,N\}} \sum_{n=1,..,N}\frac{\hat{l}(n,t)}{L_{max}(n,t)}\cdot\left[\sum_{\delta=1}^{\Delta} w_\delta(t)\cdot\Theta(n,t,\delta)\right]$$

Thus, an obvious choice for design task value functions is

$$V(n,t) := \frac{\hat{l}(n,t)}{L_{max}(n,t)}\cdot\left[\sum_{\delta=1}^{\Delta} w_\delta(t)\cdot\Theta(n,t,\delta)\right], \forall n=1,...,n\in J(t)\cup Y(t),$$

where $\hat{l}(n,t)$ is the $n-$th design task target performance level.

In the more complicated case of the market payoff introduced by (Yoshimura, 1996) the coherence of objectives between the aggregate decision level and the detailed planning level is not complete. However, we introduce below a formula describing $V(\cdot,t)$ such that maximizing the sum of design tasks value functions (at the detailed planning level) provides at least a lower bound on maximizing the market payoff function (at the aggregate decision level). Thus, the criterion used in the allocation step of the detailed planning level will partially reflect the intention of the aggregate decision maker.

If $M\left[\left(\Theta\left(t,\cdot,\delta\right)_{1\leq\delta\leq\Delta},\hat{l}\left(\cdot,t\right)\right)_{1\leq n\leq N}\right] \overset{def}{:=} \prod_{\delta=1}^{\Delta}\left[\overline{S}_{\delta}\left(t,\hat{l}\left(\cdot,t\right),\delta\right)\right]^{w_{\delta}(t)}$ (see (Yoshimura, 1996)) we have no direct decomposition indexed by the design tasks numbers.

Since

$$\max_{\hat{l}(n,t),\ n\in\{1,...,N\}} M\left[\left(\Theta\left(t,\cdot,\delta\right)_{1\leq\delta\leq\Delta},\hat{l}\left(\cdot,t\right)\right)_{1\leq n\leq N}\right]$$
$$\approx \max_{\hat{l}(n,t),\ n\in\{1,...,N\}} \log\left\{M\left[\left(\Theta\left(t,\cdot,\delta\right)_{1\leq\delta\leq\Delta},\hat{l}\left(\cdot,t\right)\right)_{1\leq n\leq N}\right]\right\}$$

we define $V\left(n,t\right):=\log\left\{\prod_{\delta=1}^{\Delta}\left[\overline{S}_{\delta}\left(t,\hat{l}\left(n,t\right)\right)\right]^{w_{\delta}(t)}\right\}-V_{\min}+\varepsilon\geq\varepsilon>0$ to obtain positive design task value functions, where

$$\exists V_{\min}=\inf_{n\in J(t)\cup Y(t),\forall\delta=1,...,\Delta}\log\left\{\prod_{\delta=1}^{\Delta}\left[\overline{S}_{\delta}\left(t,\hat{l}\left(n,t\right)\right)\right]^{w_{\delta}(t)}\right\}\in(-\infty,0)$$

since $\overline{S}_{\delta}\left(t,\hat{l}\left(n,t\right)\right)\in(0,1),\forall n\in J(t)\cup Y(t),\forall\delta=1,...,\Delta$ represent a finite number of finite values.

If

$$LHS\overset{notation}{:=}\sum_{n\in J(t)\cup Y(t)}V\left(n,t\right)$$
$$\leq\log\left\{M\left[\left(\Theta\left(t,\cdot,\delta\right)_{1\leq\delta\leq\Delta},\hat{l}\left(n,t\right)\right)_{n\in J(t)\cup Y(t)}\right]\right\}\overset{notation}{:=}RHS$$

holds, then the detailed level allocation solution (which maximizes the value of the allocated design tasks) provides an heuristic lower bound for the aggregate decision targets.

$$LHS=\sum_{n\in J(t)\cup Y(t)}\sum_{\delta=1}^{\Delta}w_{\delta}\left(t\right)\log\overline{S}_{\delta}\left(t,\hat{l}\left(n,t\right)\right)$$
$$=\sum_{\delta=1}^{\Delta}w_{\delta}\left(t\right)\sum_{n\in J(t)\cup Y(t)}\log\overline{S}_{\delta}\left(t,\hat{l}\left(n,t\right)\right)$$
$$=\sum_{\delta=1}^{\Delta}w_{\delta}\left(t\right)\log\prod_{n\in J(t)\cup Y(t)}\left[\overline{S}_{\delta}\left(t,\hat{l}\left(n,t\right)\right)\right]$$

Since $\overline{S}_{\delta}\left(t,l\left(\cdot,t\right)\right)\in(0,1),\forall\delta=1,...,\Delta$ are normalized $S-$functions $LHS\leq\sum_{\delta=1}^{\Delta}w_{\delta}\left(t\right)\log\left[\overline{S}_{\delta}\left(t,\hat{l}\left(n_{0},t\right)\right)\right]$. Since $log\left(\cdot\right)$ and $\overline{S}_{\delta}\left(t,l\left(\cdot,t\right)\right)$ (being $S-$ functions of the $\sum_{n=1,..,N}\Theta\left(n,t,\delta\right)\cdot\frac{\hat{l}(n,t)}{L_{max}(n,t)}$) are increasing functions of the levels:

$$LHS\leq\sum_{\delta=1}^{\Delta}w_{\delta}\left(t\right)\log\left\{\sum_{n\in J(t)\cup Y(t)}\left[\overline{S}_{\delta}\left(t,\hat{l}\left(n,t\right)\right)\right]\right\}\leq RHS$$

References

Abate, J. and Whitt, W. (1995). Numerical Inversion of Laplace Transforms of Probability Distributions. *ORSA Journal on Computing*, 7:36–43.

Abboud, N., Inuiguchi, M., Sakawa, M., and Uemura, Y. (1998). Manpower allocation using genetic annealing. *European Journal of Operational Research*, 111:405–420.

Adlakha, V. G. and Kulkarni, V. G. (1989). A Classified Bibliography of Research on Stochastic PERT Networks: 1966-1987. *INFOR*, 27(3):272–296.

Altman, E., Gaujal, B., and Hordijk, A. (2000). Multimodularity, Convexity and Optimization Properties. *Mathematics of Operations Research*, 25:324–347.

Altman, E., Konstantopoulos, P., and Liu, Z. (1992). Stability, Monotonicity and Invariant Quantities in General Polling Systems. *Queuing Systems*, 11:35–57.

Altman, E. and Koole, G. (1998). On submodular value functions and complex dynamic programming. *Stochastic Models*, 14(5):1051–1072.

Altman, E. and Stidham Jr., S. (1995). Optimality of monotone policies for two-action markovian decision processes, with applications to control of queues with delayed information. *Queueing Systems*, 21:267–291.

Askin, R. G. and Dawson, D. W. (2000). Maximizing Customer Satisfaction by Optimal Specification of Engineering Characteristics. *IIE Transactions*, 32:9–20.

Aslasken, E. and Belcher, R. (1992). *Systems Engineering*. Prentice Hall, Inc., Englewood Cliffs, New Jersey, USA.

Atan, T. S. and Pandit, R. (1996). Auxiliary Tool Allocation in Flexible Manufacturing Systems. *European Journal of Operational Research*, 89(3):642–659.

Athey, S. and Schmutzler, A. (1995). Product and Process Flexibility in an Innovative Environment. *RAND Journal of Economics*, 26:557–574.

Athey, S. (2002). Monotone Comparative Statistics under Uncertainty. *Quarterly Journal of Economics*, CXVII(1):187–223.

Bensoussan, A., Crouhy, M., and Proth, J. M. (1985). *Mathematical Theory of Production Planning*. North Holland, New York.

Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, USA.

Best, J. B. (1995). *Cognitive Psychology*. West Publishing Company, St. Paul, Minneapolis.

Bhattacharya, S., Krishnan, V., and Mahjan, V. (1998). Managing New Product Definition in Highly Dynamic Environments. *Management Science*, 44:50–64.

Bossert, W. (1998). Welfarism and rationalizability in allocation problems with indivisibilities. *Mathematical Social Sciences*, 35(2):133–150.

Boudarel, R., Delmas, J., and Guichet, P. (1968). *Commande optimale des processus. Tome 3: programmation dynamique et ses applications*. Dunod, Paris.

Bourgeois, L. J. and Eisenhard, K. (1988). Strategic decision processes in high velocity environments: four cases in the microcomputer industry. *Management Science*, 34:816–835.

Bowers, C. A., Braun, C. C., and Morgan Jr., B. B. (1997). Team Workload: its meaning and measurement. In Brannick, M. T., Salas, E., and Prince, C., editors, *Team Performance Assessment and Measurement: theory, research, and applications*, pages 85–108. Lawrence Erlbaum Associates, Inc., New Jersey.

Brown, S. L. and Eisenhardt, K. M. (1995). Product Development: Past Research, Present Findings, and Future Directions. *Academy of Management Review*, 20:343–378.

Buzacott, J. A. and Shanthikumar, J. G. (1980). Models for understanding flexible manufacturing systems. *AIIE Transactions*, 12(3):339–349.

Charalambous, C., Tahmassebi, T., and Hindi, K. (2000). Modelling multi-stage manufacturing systems for efficient scheduling. *European Journal of Operational Research*, 122:329–338.

Choi, S. H. and Lee, J. S. L. (2001). Computational algorithms for modeling unreliable manufacturing systems based on markovian property. *European Journal of Operational Research*, 133:667–684.

Colajanni, M., Lo Presti, F., and Tucci, S. (2000). A Hierarchical Approach for Bounding the Completion Time Distribution of Stochastic Task Graphs. *Performance Evaluation*, 41:1–22.

Committee, P. M. I. S. (2000). *A guide to the Project Management Body of Knowledge*. Project Management Institute, Upper Darby, Pennsylvania.

Dawson, C. W. and Dawson, R. J. (1995). Generalised Activity-on-Node Networks for Managing Uncertainties in Project. *International Journal of Project Management*, 13:353–362.

Dawson, R. J. and Dawson, C. W. (1998). Practical Proposals for Managing Uncertainty and Risk in Project Planning. *International Journal of Project Management*, 16(5):299–310.

Dechter, R. and Pearl, J. (1985). Generalised best-first search strategies and the optimality of A*. *Journal of the Association for Computing Machinery*, 32(3):505–536.

Dempster, M. A. H., Fisher, M. L., Jansen, L., Lageweg, B. J., Lenstra, J. K., and Rinnooy Kan, H. G. (1981). Analytical evaluation of hierarchical planning systems. *Operations Research*, 29:707–716.

Denekere, R. and Pelikan, S. (1986). Competitive Chaos. *Journal of Economical Theory*, 40:13–25.

Diks, E. B. and Kok, A. G. d. (1998). Optimal control of a divergent multi-echelon inventory system. *European Journal of Operational Research*, 111(1):75–97.

Dilworth, R. P. (1950). A decomposition Theorem for Partially Ordered Sets. *Annals of Mathematics*, 51:161–166.

Dixit, A. and Pindyck, R. (1996). *Investment Under Uncertainty*. Princeton University Press, NJ, USA, 2nd edition.

DOD/NASA (1962). *PERT/Cost Guide*. US Government Printing Office, Washington, D.C., USA.

Doumeingts, G., Girard, P., and Eynard, B. (1996). GRAI Approach to Product Development. In Huang, G. Q., editor, *Design for X*, pages 153–172. Chapman & Hall.

Dragut, A. B. and Bertrand, J. W. M. (2002a). A General Framework for Controlling Time Constrained NPD Projects. under review.

Dragut, A. B. and Bertrand, J. W. M. (2002b). Mathematical Modelling of a Hierarchical Framework for Controlling NPD Projects Under a Hard Time Constraint. under review.

Dragut, A. B. (2002). Heuristic allocation based on a dynamic programming state-space representation. *Journal of Computational and Applied Mathematics*, 140:257–273.

Dragut, A. B. (2003a). A Weakly Monotonic Backward Induction Algorithm on Finite Bounded Subsets of Vector Lattices. under review.

Dragut, A. B. (2003b). Monotonic Robust Optimal Control Policies for the Time-Quality Trade-Offs in Concurrent Product Development. under review.

Dyer, M. E., Riha, W. O., and Walker, J. (1995). A hybrid dynamic programming/branch-and-bound algorithm for the multiple-choice knapsack problem. *Journal of Computational and Applied Mathematics*, 58:43–54.

Dynkin, E. B. (1965). *Markov Processes. Vol. I,II*. Springer, Berlin.

Eisenhardt, K. and Tabrizi, B. (1995). Accelerating adaptive processes: product innovation in the global computer industry. *Administrative Science Quarterly*, 40:84–110.

Elmaghraby, S. E. and Kamburovski, J. (1992). The Analysis of Activity Networks under Generalized Precedence Relations (GPRs). *Management Science*, 38:1245–1263.

Elmaghraby, S. E. (1995). Activity Nets: A Guided Tour through Some Recent Developments. *European Journal of Operational Research*, 82:383–408.

Eppinger, S. D., Whitney, D. E., Smith, R. P., and Gebala, D. A. (1994). A Model-Based Method for Organizing Tasks in Product Development. *Research in Engineering Design*, 6:1–13.

Eppstein, D., Ghalil, Z., and Giancarlo, R. (1988). Speeding up dynamic programming. In *Proceedings of the 29th IEEE Symposium on Foundations of Computer Science*, pages 488–496.

Forman, E. H. and Gass, S. I. (2001). The Analytic Hierarchy Process – An Exposition. *Operations Research*, 49(4):469–486.

Forman, E. H. and Selly, M. A. (1999). *Decision by objectives*. World Scientific Publishing, Singapore, http://www.mdm.gwu.edu/forman.

Foulds, L. R., Hoffman, D., and Neumann, K. (1991). Stochastic Identical Parallel Machine Scheduling with OR Precedence Constraints. *Asia-Pacific Journal of Operational Research*, 8:1–25.

Friedman, E. J. and Johnson, S. (1997). Dynamic Monotonicity and Comparative Statics for Real Options. *Journal of Economic Theory*, 75(1):104–121.

Garcia, A. and Smith, R. L. (2000). Solving Nonstationary Infinite Horizon Stochastic Production Planning Problems. *Operations Research Letters*, 27(3):135–141.

Garcia, F. and Ndiaye, S. (1998). A Learning Rate Analysis of Reinforcement Learning Algorithms in Finite-Horizon. In Shavlik, editor, *Proceeedings of the 15-th International Conference on Machine Learning*, pages 215–223. Morgan Kaufmann, San Francisco, CA, USA.

Ghalil, Z. and Park, K. (1992). Dynamic Programming with Convexity, Concavity and Sparsity. *Theoretical Computer Science*, 92:49–76.

Glasserman, P. and Yao, D. D. (1994). *Monotone Structure in Discrete Event Systems*. John Wiley and Sons.

Golenko-Ginzburg, D. and Gonik, A. (1998). High Performance Heuristic Algorithm for Controlling Stochastic Network Projects. *International Journal of Production Economics*, 54:235–245.

Grama, A. Y. and Kumar, V. (1995). Parallel Processing of Combinatorial Optimization Problems. *ORSA Journal of Computing*, 7(4):365–385.

Grossman, G. M. and Shapiro, C. (1986). Optimal Dynamic R&D Programs. *RAND Journal of Economics*, 17(4):581–594.

Hackman, S. T. and Leachman, R. C. (1989). A General Framework for Modeling Production. *Management Science*, 35:478–495.

Haluk, E. I. (2000). Consistency in House Allocation Problems. *Journal of Mathematical Economics*, 34(1):77–97.

Haque, B., Pawar, K. S., and Barson, R. J. (2000). Analysing Organisational Issues in Concurrent New Product Development. *International Journal of Production Economics*, 67:169–182.

Herroelen, W., De Reyck, B., and Demeulemeester, E. (1998). Resource-Constrained Project Scheduling: A Survey of Recent Developments. *Computers and Operations Research*, 25(4):279–302.

Heyman, S. P. and Sobel, M. J. (1984). *Stochastic Models in Operations Research*, volume II. McGraw-Hill, New York.

Hinderer, K. F. and Stieglitz, M. (1994). On polychotomous search problems. *European Journal of Operations Research*, 73(1):279–294.

Hinderer, K. F. (1984). On the structure of solutions of stochastic dynamic programs. In *Proceedings of the 7th Conference on Probability Theory*, pages 173–182, Brasov, Romania.

Hinderer, K. (1967). *Zur Theorie stochastischer Entscheidungmodelle*. Habilitationsschrift, Univ. Stuttgart.

Hinderer, K. (1970). *Foundations of Nonstationary Dynamic Programming with Discrete-Time Parameter*. Springer-Verlag, New York.

Hopenhayn, H. and Prescott, E. (1992). Stochastic Monotonicity and Stationary Distributions for Dynamic Economies. *Econometrica*, 60:1307–1406.

Huchzermeier, A. and Loch, C. H. (2001). Project Management Under Risk: Using the Real Options Approach to Evaluate Flexibility in R&D. *Management Science*, 47:85–101.

Ibaraki, T., Hasegawa, T., Teranaka, K., and Iwase, J. (1978). The Multiple-choice Knapsack Problem. *Journal of Operations Research Society Japan*, 21:59–95.

Innam, R. R. (1999). Empirical Evaluation of Exponential and Independence assumptions in queueing models of manufacturing systems. *Production and Operations Managemen*, 5:409–432.

Kalin, D. (1978). A Note on Monotone Optimal Policies for Markov Decision Processes. *Mathematical Programming*, 15:220–222.

Klamroth, K. and Wiecek, M. M. (2000). Dynamic Programming Approaches to the Multiple Criteria Knapsack Problem. *Naval Research Logistics*, 47:57–76.

Kleinrock, L. (1975). *Queuing Systems*, volume 1, Theory. John Wiley and Sons.

Korf, R. E. (1993). Linear-space Best-first Search. *Artificial Intelligence*, 62:41–78.

Korf, R. E. (1995). Space-Efficient Search Algorithms. *Computing Surveys*, 22(3):337–339.

Kreps, D. M. (1977a). Decision Problems with Expected Utility Criteria, II: Stationarity. *Mathematical of Operations Research*, 2(3):266–274.

Kreps, D. M. (1977b). Decision Problems with Expected Utility Criteria, I: Upper and Lower Convergent Utility. *Mathematical of Operations Research*, 2(1):45–53.

Krishnan, V., Eppinger, S. D., and Whitney, D. E. (1997). A Model Based Framework to Overlap Product Development Activities. *Management Science*, 43:437–451.

Krishnan, V. and Ulrich, K. T. (2001). Product Development Decisions: A Review of the Literature. *Management Science*, 47:1–21.

Kulkarni, V. G. and Adlakha, V. G. (1986). Markov and Markov-Regenerative PERT Networks. *Operations Research*, 34(5):769–781.

Kumar, V. and Kanal, L. N. (1988). The CDP: A Unifying Formulation for Heuristic Search, Dynamic Programming and Branch-and-bound. In Kanal, L. N. and Kumar, V., editors, *Search in Artificial Intelligence*, pages 1–27. Springer Verlag, New York.

Kusiak, A. (1995). *Engineering Design. Products, Processes, and Systems*. Academic Press, London, U.K.

Landsberg, A. S. and Friedman, E. J. (1996). Dynamical Effects of Partial Orderings in Physical Systems. *Physical Review E*, 54(4):3135–3141.

Larkin, J. H. and Raif, F. (1979). Understanding and teaching problem solving in physics. *European Journal of Science Education*, 1:191–203.

Liu, Z., Nain, P., and Towsley, D. (1995). Sample path methods in the control of queues. *QUESTA*, 21:293–336.

Loehle, C. (1994). A critical path analysis of scientific productivity. *Journal of Creative Behavior*, 28:33–47.

Martello, S., Pisinger, D., and Toth, P. (2000). New Trends in Exact Algorithms for the 0-1 Knapsack Problem. *European Journal of Operational Research*, 123:325–332.

McCormack, A., Verganti, R., and Iansiti, M. (2001). Developing Products on "Internet Time": The Anatomy of a Flexible Development Process. *Management Science*, 47:133–150.

McDermott, C. (1999). Managing Radical Product Development in Large Manufacturing Firms: A Longitudinal Study. *Journal of Operations Management*, 17:631–644.

Mirsky, L. (1971). A Dual of Dilworth decomposition theorem. *American Mathematical Monthly*, 78:876–877.

Moscarini, G. and Smith, L. (2001). The Optimal Level of Experimentation. *Econometrica*, 11:1629–1644.

Naor, P. (1969). On the regulation of queueing size by levying tolls. *Econometrica*, 37:15–24.

Neumann, K. and Steinhardt, U. (1979). *GERT Networks and Time-Oriented Evaluation of Projects*. Springer Verlag, Berlin, Germany.

Neumann, K. and Zimmerman, J. (1998). Heuristic Procedures for Parallel Machine Scheduling Problems with Stochastic Precedence Constraints. *Annals of Operations Research*, 83:115–136.

Neverman, P. and I., R. (1985). Holes in ordered sets. *Graphs and Combinatorics*, pages 339–350.

Nilsson, N. J. (1982). *Principles of Artificial Intelligence*. Springer Verlag, Berlin.

Oorschot, K. E. v., Bertrand, J. W. M., and Rutte, C. G. (2002). Am empirical study into the causes of the lateness of new product development projects. *Beta working paper WP-100, Technische Universiteit Eindhoven*.

Oorschot, K. E. v. (2001). *Analysing Radical NPD Projects from an Operational Control Perspective*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands.

Pich, M. T., Loch, C. H., and De Meyer, A. (2002). On uncertainty, Ambiguity, and Complexity in Project Management. *Management Science*, 48:1008–1023.

Pontradolfo, P. (2000). Project Duration in Stochastic Networks by the PERT-Path Technique. *International Journal of Project Management*, 18(3):215–222.

Puterman, M. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons.

Reed, S. K. (1988). *Cognition, Theory and Applications*. Brooks/Cole Publishing Company, Pacific Grove, California.

Reeves, G. R. and Reid, R. C. (1999). A Military Reserve Manpower Planning Model. *Computers And Operations Research*, 26(12):1231–124?

Reibman, A. (1990). Modeling the effect of reliability on performance. *IEEE Transactions on Reliability*, 39(3):314–320.

Reppening, N. R. (2000). A Dynamic Model of Resource Allocation in Multi-project Research and Development Systems. *System Dynamics Review*, 16:173–212.

Romeijn, H. E. and Morales, R. D. (2000). A Class of Greedy Algorithms for the Generalized Assignment Problem. *Discrete Applied Mathematics*, 103:209–235.

Russell, S. J. and Norvig, P. (1995). *Artificial Intelligence. A modern approach*. Prentice Hall, New Jersey.

Serfozo, R. F. (1976). Monotone Optimal Policies for Markov Decision Processes. *Mathematical Programming Study*, 6:202–215.

Shtub, A., Bard, J. F., and Globerson, S. (1994). *Project Management. Engineering, Technology, and Implementation*. Prentice Hall. Inc., Englewood Cliffs, New Jersey.

Sieger, D. B., Badiru, A. B., and Milatovic, M. (2000). A Metric for Agility Measurement in Product Development. *IIE Transactions*, 32:637–645.

Sobek, D. K., Ward, A. C., and Liker, J. K. (1999). Toyota's principles of set-based concurrent engineering. *Sloan Management Revue*, 40:637–83.

Tatikonda, V. M. and Rosenthal, S. R. (2000a). Technology Novelty, Project Complexity, and Product Development Project Execution Success: A Deeper Look at Task Uncertainty. *IEEE Transactions on Engineering Management*, 47(1):74–86.

Tatikonda, V. and Rosenthal, S. (2000b). Succesuful execution of product development projects: Balancing firmness and flexibility in the innovation process. *Journal of Operations Management*, 18:401–425.

Tavares, L. V. (2002). A review of the contribution of Operational Research to Project Management. *European Journal of Operational Research*, 136:1–18.

Taylor, B. W. and Moore, L. J. (1980). R&D Project Planning with Q-GERT Network Modeling and Simulation. *Management Science*, 26:44–59.

Thomke, S. and Reinerstein, D. (1998). Agile Product Development: Managing Development Flexibility in Uncertain Environments. *California Management Review*, 41(1):8–30.

Topkis, D. M. (1976). Minimizing a Subadditive Function on a Lattice. *Operations Research*, 26:305–321.

Topkis, D. M. (1998). *Supermodularity and Complementarity*. Frontiers of Economic Research series. Princeton University Press.

Turner, J. R. and Cochrane, R. A. (1993). Goals-and-methods Matrix: Coping with Projects with Ill Defined Goals And/or Methods of Achieving Them. *International Journal of Project Management*, 11(2):93–102.

Ulrich, K. and Eppinger, S. (2000). *Product Design and Development*. McGraw-Hill Inc., New York.

Weber, R. R. and Stidham, Jr., S. (1987). Optimal Control of service rates in networks of queues. *Advances of Applied Probability*, 19:202–218.

Weiss, G. (1995). A tutorial in stochastic scheduling. In Chretienne, P., G., C. J. E., Lenstra, J. K., and Liu, Z., editors, *Scheduling Theory and its Applications*, pages 33–64. John Wiley & Sons, New York.

White, C. C. (1980). The Optimality of Isotone Strategies for Markov Decision Problems with Utility Criterion. In Hartley, R., Thomas, L. C., and White, D. J., editors, *Recent Developments in Markov Decision Processes*, pages 261–276. Academic Press.

White, D. J. (1969). *Dynamic Programming*. Oliver and Boyd, London.

Wickens, C. D. and Hollands, J. G. (1999). *Engineering psychology and human performance*. Harper Collins, New York.

Wideman, R. M. (2000). *Project and Program Risk Management*. Project Management Institute, Newton Square, Pennsylvania.

Williams, H. P. (1978). *Model Building in Mathematical Programming*. Wiley, Chichester.

Williams, T. M. . (1999). The need for new paradigms for complex projects. *International Journal of Project Management*, 17:269–273.

Yao, F. F. (1980). Efficient Dynamic Programming Using Quadrangle Inqualities. In *Proceedings of 12th ACM Symposium on Theory of Computing*, pages 429–435.

Yoder, B. and Mason, D. (1995). Evaluating qfd relationships through the use of regression analysis. In *Proceedings of the Seventh Symposium on Quality Function Deployment*, pages 35–59. ASI&GOAL/QPC, American Supplier Institute, Livonia, Michigan.

Yoshimura, M. (1996). Design optimisation for product life cycle. In Huang, G. Q., editor, *Design for X*, pages 424–440. Chapman & Hall, London.

Curriculum Vitae

Andreea Bogdana Dragut was born in Bucharest, Romania, on May 6th, 1974. After completing her pre-university education at the Computer Science High School of Bucharest, she started in the same year to study at the University of Bucharest. She graduated the first from the Department of Applied Mathematics, of the Faculty of Mathematics.

After graduation, she worked as a research assistant at the Center for Mathematical Statistics of the Romanian Academy of Sciences, where, in December 1999, she received her PhD title in Mathematics with the thesis entitled ”Contributions to optimal classification methods”.

From September 1999, she worked as a trainee research assistant on an interdisciplinary project in the Operations Planning and Control group of the Faculty of Technology Management, at the Technische Universiteit Eindhoven. The research on mathematical models for the control of new product development projects, carried out in the period September 1999 — May 2003, led to this dissertation.

After the defense, which is to take place in November 2003, Andreea B. Dragut is planning to proceed as a researcher in the same Operations Planning and Control group.