Integrated and Dynamic Vehicle and Crew Scheduling

ISBN 90 5170 764 9
Cover design: Crasborn Graphic Designers bno, Valkenburg a.d. Geul

This book is no. 325 of the Tinbergen Institute Research Series, established through cooperation between Thela Thesis and the Tinbergen Institute. A list of books which already appeared in the series can be found in the back.

Integrated and Dynamic Vehicle and Crew Scheduling

Geïntegreerde en dynamische
voertuig- en personeelsplanning

Proefschrift

ter verkrijging van de graad van doctor
aan de Erasmus Universiteit Rotterdam
op gezag van de Rector Magnificus
Prof.dr. S.W.J. Lamberts
en volgens besluit van het College voor Promoties.

De openbare verdediging zal plaatsvinden op
vrijdag 20 februari 2004 om 13.30 uur
door

Dennis Huisman

geboren te Dordrecht.

Promotiecommissie

Promotoren:	Prof.dr. A.P.M. Wagelmans
	Prof.dr.ir. R. Dekker
Overige leden:	Prof.dr. L.G. Kroon
	Dr. G. Desaulniers
	Prof.dr. S.L. van de Velde

Acknowledgements

Thinking back on the last four-and-a-half years, immediately Richard Freling comes in my mind. I met him about five years ago for the first time, when I started to write my master thesis, which was an extension of Richard's research on integrated vehicle and crew scheduling. Later on, after I became a PhD student, we had a lot of discussions about our joint research, but also about life in general. For me, it was therefore a shock to hear that he was seriously ill. After a fight of five months, he passed away in January 2002. However, he is still and will always be in my mind as an extremely kind and helpful person. At this place, I would therefore like to thank him for his support, since without him this thesis would never have been written!

Furthermore, I would like to thank my promotors Albert Wagelmans and Rommert Dekker. Albert, I met you for the first time, which you probably do not remember anymore, at an information day on Econometrics in March 1995. Later on, you were my teacher in Combinatorial Optimization, where my interest in this area was raised. Afterwards, we had some talks about following several PhD-courses and becoming a student assistant. After a successful collaboration, I decided to start a PhD under your (and Richard's) supervision. During these four-and-a-half years we had a lot of interesting discussions about the research and you gave a lot of valuable comments, which significantly improved the papers we wrote and this thesis as well. Moreover, we always had a good personal relation, which in my opinion is at least as important as a good professional relation. Rommert, we always had a lot of discussion during lunch time about the field of Operations Research, money and other issues related to science in general. I always enjoyed these discussions.

Some valuable comments to improve this thesis were given by Leo Kroon. I appreciate it very much that you read an earlier draft of this thesis very carefully and gave some helpful suggestions. Furthermore, I would like to thank all other members of the PhD committee: Guy Desaulniers, Steef van de Velde, Raymond Kwan and Hans Frenk. Moreover, I would like to thank Guy and Raymond for their time to be here during the defense and for the talk they gave at the workshop prior to this defense.

This thesis would also not have been possible without the help of several companies, who provided me with data and other support. Especially, my acknowledgements go to Martin Gerritsen (Connexxion), Cees Boogaard (RET) and Paul van Mulekom (Stadsbus

Maastricht).

During my PhD period, I gave some courses and supervised several Master students. Hereby, I thank all of them for listening to my lectures and/or working on the case studies. In particular, I enjoyed supervising the theses of Albert Varekamp, Ildikó Fodor Birtalan, Pieter-Jan Fioole, Kees van de Wagt and Sebastiaan de Groot. Several parts of Sebastiaan's research are also described in this thesis and make a valuable contribution to some practical aspects of integrated vehicle and crew scheduling. Furthermore, I joined the Rintel-project after Richard's illness. In this project I learned a lot about working in a great team and about the problems of the Dutch Railways. The resulting paper of this project was awarded as the best (PhD-)student paper in railways. Therefore, I thank the other contributors of this paper (Richard Freling, Leo Kroon and Ramon Lentink) for the successful corporation and several student assistants (Pieter-Jan, Wilco, Arnoud, Cor and Ildikó) for the valuable contributions in the implementation. The dinner and movie we had together, was a nice consequence of this award.

Of course, one of the nicest aspects in a PhD-project is the opportunity to attend conferences abroad. Some of those conferences were in beautiful places such that it was possible to combine it with a holiday. To most of these conferences, I went together with Ramon Lentink, which always guaranteed a great time. In particular, the four weeks in the USA were very nice. Hereby, I thank you (and w.r.t. the time in the States also your girl-friend Bearene) for those great times abroad, the successful corporation in several projects and for being a good friend!

During these years, I had several roommates. Most time, I spent with Nanda Piersma and Marisa de Brito. Nanda, thanks for the support in the first year of my PhD and Marisa, thanks for being my roommate for three years, which at least further improved my English and my knowledge about other cultures (including the strange Dutch habits).

Furthermore, I am grateful to all other (ex-)colleagues of the Econometric Institute, which I did not mention before. In particular: Arjan, Gonda and Peter (for the lunches), Csaba, Emöke and Gabriella (for the Hungarian touch), Elli (for her support with organizing COPT), Eric and Robin (for being neighbors), Kevin and Ovidiu (for the LNMB exercises), Patrick (for the fruitful discussions), Ruud (for the positive atmosphere in the MABES group), Kitty and Tineke (for administrative support), and Willie (for the personal interest).

Finally, I would like to thank the most important people in my life, my parents and brother, in Dutch. *Pa en Ma hartstikke bedankt! Zonder jullie zou dit proefschrift nooit tot stand zijn gekomen en zou ik nu niet de persoon geweest zijn die ik nu ben. Jeroen - ik zal jou deze keer niet overslaan(!) - bedankt voor alle gezellige dagen, wedstrijdjes en discussies.*

Dennis Huisman

Rotterdam, February 2004

Contents

Chapter 1

Introduction

This thesis deals with vehicle and crew scheduling. These are two (of the most) important planning problems in public transport scheduling, since vehicles and crews are the two main resources for the public transport companies to provide services to the passengers. In the context of this thesis, the terms scheduling and planning are interchangeable, since almost all planning problems at a public transport company are actually scheduling problems.

During recent years, there has been an increased attention to computerized solution approaches for vehicle and crew scheduling. There are several reasons for this. First of all, there has been much more focus on cost reductions due to the increased competition in the public transport market and the pressure on governments to cut expenses. Secondly, planning problems have become more and more complex due to an increased size of the problems and an increased complexity of labor rules. This makes it more difficult to solve them manually, while on the other hand the power of computers and algorithms has increased tremendously. The third and final reason we mention here, is the fact that many young planners are used to work with computers and some of them do not have the skills anymore to solve those scheduling problems manually.

However, our focus in this thesis is not only on cost reductions but also on an increased service to the customers. In particular, we also consider the aspect of delays. Due to delays and too tight schedules, a domino effect can occur, i.e. one delay at a certain moment leads to many other delays during the rest of the day. This has become an important aspect, since public transport companies have to fulfill certain requirements by governments with respect to the number of delays. If there are too many delays, a company sometimes has to pay a certain penalty.

In the remainder of this chapter, we describe the general planning process of a public transport company and we discuss the differences between the planning problems of a bus company and those of a railway or airline company (Section 1.1). The reason for this discussion is that our main focus in this thesis is on bus and bus driver scheduling. However, a lot of ideas can also be used for other modes of transport. The opposite is

of course also true, e.g. a lot of ideas in bus driver scheduling are initially developed for airline crew scheduling. A motivation for the research in this thesis and its purpose is given in Section 1.2. Subsequently, we discuss in Sections 1.3 and 1.4 some well-known combinatorial optimization problems and techniques, respectively. These problems occur as subproblems in the remainder of this thesis, while the techniques are applied to solve several problems. Finally, we give an overview of the remainder of this thesis in Section 1.5.

1.1 Planning Process of a Public Transport Company

In Figure 1.1 we show the relation between the four operational planning problems at a public transport company, which can be either a bus, tram, metro, train or airline operator.

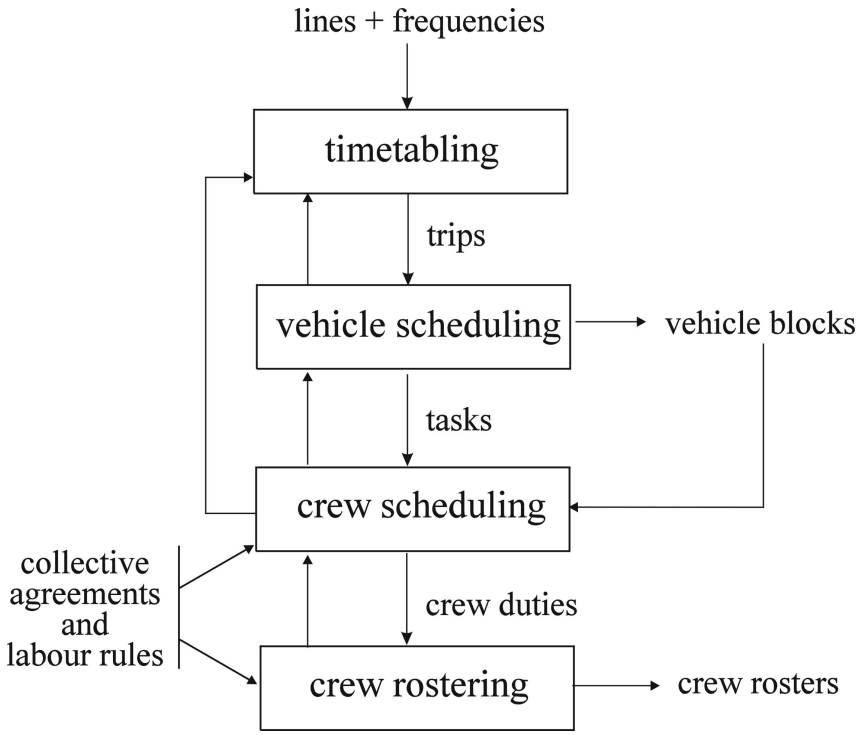

Figure 1.1: Planning process at a public transport company

Decisions about which routes or *lines* to operate and how frequently, are input for the operational planning process. They can be either given by the marketing department of the company or determined by (local, regional or national) authorities. Furthermore, the travel times between various points on the route are assumed to be known. Based on the lines and frequencies, timetables are determined resulting in *trips* with corresponding start and end locations and times. The second planning process is vehicle scheduling,

which consists of assigning vehicles to trips, resulting in a schedule for each vehicle. The vehicles, which are not in use for some time are parked in a *depot*. A schedule for a vehicle is split into several *vehicle blocks*, where a new vehicle block starts at each departure from the depot. On such a block a sequence of *tasks* can be defined, where each task needs to be assigned to a working period for one crew (a *crew duty* or *duty)* in the crew scheduling process. The feasibility of a duty is dependent on a set of collective agreements and labor rules, that refer to sufficient rest time, etcetera. Crew scheduling is short term crew planning (one day), i.e. assigning crew duties to tasks on each specific day, while the crew rostering process is long term crew planning (e.g. half a year) for constructing rosters from the crew duties.

In Subsection 1.1.1 we consider some details of this planning process for a bus company. Differences in the planning problems of other modes of public transport are discussed in Subsection 1.1.2.

1.1.1 Planning Process of a Bus Company

Although the planning process for each bus company is quite similar, we have to make some distinctions between those companies. Some companies consider each bus line separately, while others consider a part of the network or even the whole network at once. Another important characteristic is the number of depots and vehicle types that is considered in the planning process. Later on in this thesis, it will be explained that for some of those planning problems there is an important difference from a computational point of view between problems with one single depot and one vehicle type and those with multiple depots and/or more vehicle types.

In the Netherlands, for example, the RET (local public transport company in Rotterdam; see Section 3.7 for an application on vehicle and crew scheduling) considers each line separately and a single depot and vehicle type has already been assigned to each line by the management. Other companies like Connexxion (largest extra-urban and regional bus company in the Netherlands; see Chapters 4 and 5) consider a part of their network at once. In most cases this part is defined by a geographical area or a "licence" area defined by a (regional) government where different companies tender for providing bus transport. Furthermore, they take all depots in this area into account. However, some lines have already been assigned to a certain depot or a certain subset of the depots. Small companies such as Stadsbus Maastricht (local bus company in Maastricht; see Section 2.4 for an application on crew scheduling) consider their whole network at once and have only one depot/vehicle type.

Before we consider some details of the planning process for bus transport, we would like to mention that the actual operations always differs from the planning. This happens for instance if drivers are ill or delays occur. Such changes are not part of the planning

process itself. Therefore, they will not be considered in the remainder of this section. However, in the last chapter of the thesis we consider a dynamic planning process, where delays are taken into account.

Timetabling is especially interesting if a large network of bus lines is considered, since connections for passengers should be taken into account at some intersections. For example, in regional bus transport a lot of bus lines have a kind of "hub-and-spoke" network, where several lines arrive at (depart from) the hub at the same time such that passengers can easily change from one line to another one. If only one bus line is considered the timetabling problem by itself is not very interesting. However, in this case small changes in the timetable can be used to save vehicles and/or crews later on.

In the vehicle scheduling problem, bus schedules are determined. That is all trips are assigned to a certain depot and vehicle type, and a schedule for each bus is obtained. Such a schedule starts and finishes at a depot and performs a sequence of trips. In between, a bus is allowed to return to a depot for some time. Therefore, we define a *vehicle block* as a part of the schedule for one vehicle from a depot to a depot. One vehicle can thus perform several vehicle blocks. Furthermore, there are movements necessary in a vehicle block in time and/or space between two trips, from a depot to the first trip and from the last trip to a depot. Such a movement is called a *deadhead*.

In Figure 1.2 we give an example of such a vehicle schedule for one vehicle consisting of two blocks and seven trips. The vehicle starts in the depot at 5:00 and moves (empty) to location A. In A, it starts with the first trip to B (5:30-6:30). After this trip, the vehicle stays in B for an half hour, which is an example of a deadhead between two trips without moving from one location to another. After the second trip (7:00-8:00 from B to A), the vehicle moves empty from A to C. Afterwards, it performs three other trips before moving back from location D to the depot, where the first block finishes. After three hours, the same vehicle starts with a second sequence of trips and moves to location B. Then it performs two trips and it returns to the depot.

On each vehicle block certain *relief points* can be defined. These are points in time (and space), where one driver could be relieved by another one either to take his break or to finish his duty. The sequence of deadheads and trips between two relief points is called a *task*. Therefore, a task is the smallest part on a vehicle block which has to be assigned to one crew member. Consider again Figure 1.2 and assume that a driver can be relieved at locations A and D. Then the first block consists of seven tasks, namely from 5:00 (depot) to 5:30 (A), 5:30 (A) to 8:00 (A), 8:00 (A) to 9:30 (D), 9:30 (D) to 11:00 (A), 11:00 (A) to 11:30 (A), 11:30 (A) to 12:30 (D) and from 12:30 (D) to 13:30 (depot). In a similar way, three tasks can be defined on the second block.

In the crew scheduling problem those tasks should be assigned to crew duties such that all tasks are performed and each duty is feasible. Sometimes we need to make a distinction between cases where *changeovers*, i.e. a driver changes from one bus to another one during

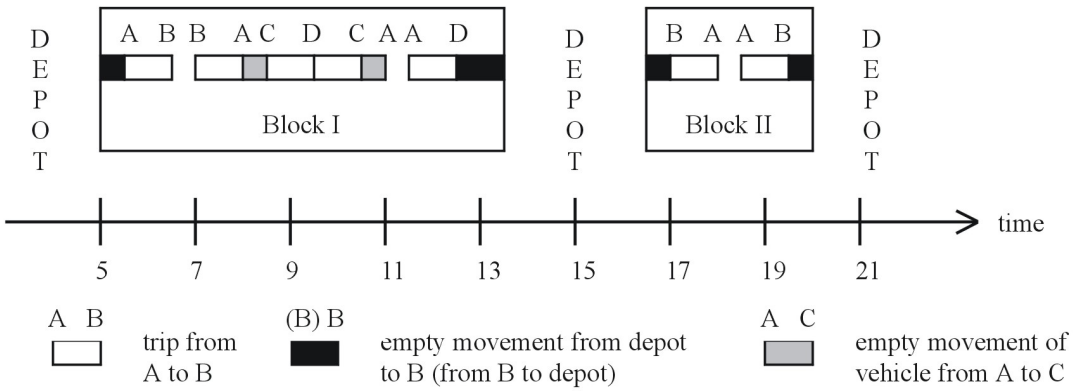

Figure 1.2: Schedule of one vehicle consisting of two blocks and seven trips

his break, are allowed and where they are not. In the last case, drivers stay the whole duty on one bus, which means that if the driver has a break the bus should wait. In general, a duty consists of several (mostly two) *pieces of work*, where a piece of work (in short *piece*) is defined as a sequence of tasks on one vehicle without breaks. An example of such a piece in Figure 1.2 is the sequence of tasks that starts at 5:00 in the depot and ends at 9:30 in D. Such a piece has a length of 4 1/2 hours. During the crew scheduling problem a lot of different constraints have to be taken into account. These constraints follow from laws, collective labor agreements and company rules.

The first three planning problems have a time horizon of one day, i.e. a problem should be solved on a Monday, a Tuesday and so on. Sometimes, the problems for several days are similar like, for example, on a Monday and a Tuesday. In general, the problems do not change for a whole timetable period, which is mostly half a year or one year. However, sometimes the timetable and schedules need to be changed more often. This occurs, for instance, if construction work on a major road or bus lane leads to different travel times.

In the last planning step the crew rosters are determined and the results of the planning processes for different days are coupled. Most companies use cyclic rosters on a weekly basis, i.e. driver 1 is assigned to duty A on Monday, duty B on Tuesday, is free on Wednesday and Thursday and is assigned to duties C, D and E on Friday, Saturday and Sunday, respectively, in week 1 and driver 2 performs the same order of duties and free days in week 2.

1.1.2 Differences with Railway and Airline Planning

There are several differences between railway planning and the described planning process of a bus company in the previous subsection. These differences are summarized below.

- Timetabling in railway planning is much more complicated than in bus transport, since the infrastructure of a railway system has to be taken into account. For example, between each pair of trains on the same track there should be a minimum headway due to safety regulations. Furthermore, differences in speed should be taken into account: a fast Intercity service cannot start just after a slow train in the same direction. Similar remarks can be made about the railway stations, where for instance cross-platforms connections between certain trains are preferred. In a bus system all those complicating problems do not occur.

- The vehicle scheduling problem is also slightly different, since different train units can be combined to form one larger train. This is not possible with buses. The order of the units in a train is another complicating factor. Furthermore, other problems that do not exist in the bus context, like shunting at railway stations should be taken into account. Finally, the movement of empty trains is often restricted, which makes the problem in this aspect somewhat easier than bus scheduling, since there are much less possible solutions.

- The crew scheduling and rostering problem are very similar. There is only one major difference: in general, a train needs more crew members (driver and guards) than a bus (only a driver). However, this does not make the problem much more complicated, since drivers and guards can either be scheduled independently or they are scheduled as teams consisting of a driver and one or two guards.

From these differences we can conclude that, in general, railway planning is more complicated than bus planning. The differences between bus and airline planning are much smaller. They are summarized below.

- Since most airlines have a hub-and-spoke network, the timetabling step can be compared with the one in regional bus transport where connections should be ensured on the hubs. However, the number of slots (capacity) at the airports is limited for each airline.

- Vehicle scheduling is divided into two steps for most airlines: fleet assignment and aircraft routing. In the fleet assignment problem, types of planes are assigned to each flight, while in the aircraft routing problem the actual schedules are determined. The first problem has a planning horizon of a season and the second one of a week. This is the major difference with bus scheduling where the planning horizon is one day. On the other side, the trips in airline planning are much longer than in bus scheduling.

- Crew scheduling (and rostering) differ in the number of crew members that has to be scheduled. However, pilots, copilots, cabin personnel can be scheduled independently of each other, in general. A more important difference is that the crew

related processes are mostly divided into three steps: duty scheduling, crew pairing and crew rostering. A pairing is defined here as a sequence of duties without a (long) rest period. There are also some differences from a computational point of view. Since the trips are much longer, the number of trips per duty is much smaller than for bus driver scheduling. Furthermore, crew is much more restricted to certain types of airplanes. Therefore, large problems can be split up easily.

If we compare the differences above, it seems that airline and bus planning have the same degree of difficulty. This in contradiction with railway planning, which is much more complicated than bus and airline planning.

1.2 Motivation and Purpose of the Thesis

In this thesis, we do not consider vehicle and crew scheduling problems as separate problems, which are solved sequentially, but we solve them as one integrated problem. The main reason for this is that several authors obtained significant savings with such an integrated approach, since crew costs dominate vehicle costs and the feasibility of a crew schedule is much more restricted than that of a vehicle schedule. These observations are made for the case with a single depot. The savings are, in general, much higher in the case of multiple depots than in the case of only a single depot or if all trips have to be assigned to a certain depot. We will show in the following, simple example of 3 trips that one duty can be saved by using an integrated approach without using more vehicles.

Example 1.1 *Consider the following three trips: trip 1 starting in A at 8:00 and ending in C at 13:30; trip 2 from C to B (14:00-15:00) and trip 3 from B to A (15:30-21:30). There are two depots and the travel times from depot 1 to A, B and C are 20, 50 and 30 minutes respectively, while these times are 50, 10 and 30 minutes for depot 2. Furthermore, consider the following crew rules: a duty is feasible if the duration is at most 8 hours; each duty starts and ends in its own depot or in case it does not start/end there, a walking time (equal to the travel time between the start/end location and the depot) should be added to the length of a duty. Finally, we have the following cost structure. A fixed cost of 1,000 per vehicle and 1,000 per duty and variable vehicle costs of 1 per minute time that a vehicle is without passengers.*

Then, the optimal vehicle schedule consists of 1 vehicle from depot 1 with costs 1,100 (1,000 + 20 + 30 + 30 + 20). Given this vehicle schedule the optimal crew schedule has 3 duties with costs of 3,000. So the overall costs are equal to 4,100. The explanation is as follows (see also Figure 1.3). A crew schedule with 2 duties is not possible, since the first duty starts at 7:40 and therefore has to finish before 15:40. However, the walking time from B to depot 1 is 50 minutes. So the first driver should be relieved by another one

at 14:00, which means that the second duty starts at 13:30. It is obvious that this duty cannot do the remainder of the trips, since the vehicle returns in the depot at 21:50.

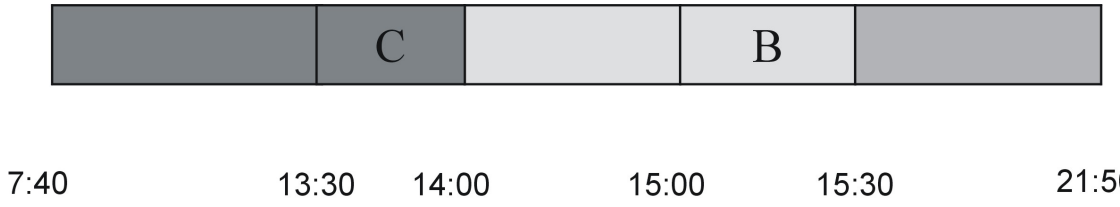

Figure 1.3: The optimal crew schedule given the best vehicle schedule consists of 3 duties

*However, the overall optimal solution has costs 3,160, which is a cost decrease of 22.9%! Notice that this saving will be even larger when crew costs increase. This solution has 2 duties and 1 vehicle from depot 2 (costs: 2 * 1,000 + 1,000 + 50 + 30 + 30 + 50 = 3,160). The first duty starts in depot 2 at 7:10 and finishes in B at 15:00 (see Figure 1.4). Notice that the length of this duty is exactly 8 hours, since there is a walking time of 10 minutes from B to depot 2. The second one starts in B at 15:00 and finishes in depot 2 at 22:20 (including a walking time of 10 minutes, the duty length is 7:30 hours).*

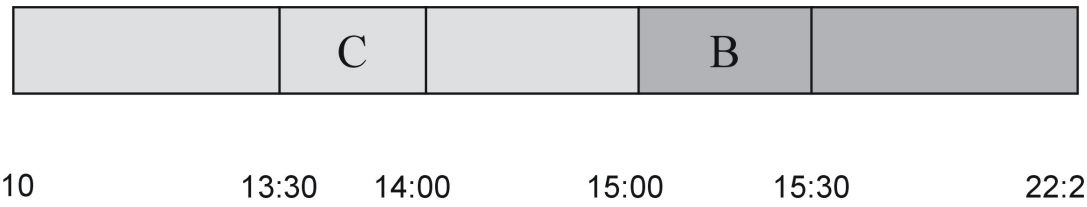

Figure 1.4: The optimal overall solution consists of 2 duties

Although the potential saving is large, in the literature, only small (mostly single-depot) problems have been solved using such an integrated approach. This is due to the computational complexity of the problem. Therefore, the challenge is to solve larger problems, problems with multiple depots and to incorporate difficult rules from public transport companies.

Next to cost reductions, the reliability of the public transport services to the passengers is an important issue. Therefore, we will consider new ways of vehicle and crew scheduling such that the number of delays can be reduced. Instead of the traditional static approach, which is currently used, we will look at a dynamic approach. The static approach, i.e. there are no changes in the schedules during a timetable period, has as disadvantage that when a delay occurs on a certain day, the next trip performed by that vehicle and/or driver will sometimes start late. Therefore, new delays can occur which possibly have a similar “snowball” effect.

The basic idea of a dynamic approach is that the schedules are constructed for a certain part of the day, then for the next part and so on. We will illustrate this basic idea and the potential advantage of such an approach with the following, simple example.

Example 1.2 *We consider six trips (called 1, 2, 3, 4, 5 and 6) where the first three trips end at a certain bus station at 10:00, 10:05 and 10:10, respectively. Furthermore, the other trips start from the same bus station at 10:15, 10:20 and 10:35, respectively. It is obvious that those three trips should be executed by at least three vehicles. Moreover, suppose that the drivers do not change their vehicle and that the only constraint with respect to the feasibility of a drivers' duty is a minimum break length of 10 minutes every time passing this bus station. Then, the optimal (static) integrated solution consists of 3 drivers (e.g. driver 1 performs trips 1 and 4 on vehicle 1, driver 2 trips 2 and 5 on vehicle 2, and driver 3 trips 3 and 6 on vehicle 3).*

Furthermore, suppose we use the schedule above and that a certain day trip 2 has a delay at arrival of 7 minutes, which means that it arrives at 10:12. Then its successor, trip 5, can still start at the right time, however, the break for the second driver is only 8 minutes, which is less than the required minimum break length. If the delay would be even larger, say 20 minutes, trip 5 would start 5 minutes late.

If we would use a dynamic approach here instead of a static one, we would probably be able to find another solution, where also three vehicles and drivers are used, but without trips starting late and violating the break rule. This can be illustrated in the case where we construct a schedule every hour. Then, we first construct a schedule until 10:00, i.e. we assign three vehicles and drivers to the trips 1, 2 and 3, respectively. At 10:00 we construct the schedule for the next hour, however, we already know at that time that trip 2 will have a delay of approximately 7 minutes. Then we optimize again and assign trip 4 to vehicle/driver 1, trip 5 to vehicle/driver 2 and trip 6 to vehicle/driver 3. Note that we would get exactly the same solution if the delay was 20 minutes. So in that case, all trips would start on time.

Of course, in reality the problems are much more complex than in this example. Therefore, the challenge is to develop such a dynamic approach that is able to solve problem instances of realistic size and can deal with several standard labor rules.

All together, we hope to achieve the following three purposes.

1. We show that integrated approaches can lead to significant improvements over sequential approaches, especially in certain specific cases.

2. By applying the integrated approaches to practical problems of a reasonable size, we show that they can be used to solve real-world problems.

3. We show different ways to take delays into account during the planning process such that the service to the passengers can sometimes be increased at equal costs.

1.3 Combinatorial Optimization Problems

In a *combinatorial optimization problem*, one would like to find an optimal solution in a certain finite set of feasible solutions. In this section we discuss some well-known combinatorial optimization problems, which will be used in the remainder of this thesis. Furthermore, we provide some complexity results and mathematical formulations of these problems. We assume that the reader is familiar with the basic concepts of complexity theory and mathematical programming.

1.3.1 Minimum Cost Flow problem

The *minimum cost flow problem* can be formally defined as follows (see Ahuja *et al.* (1993)).

Definition 1.3 *Determine a least cost shipment of a commodity through a network that will satisfy the flow demands at certain nodes from available supplies at other nodes.*

Define $G = (N, A)$ as a directed network with N as the set of nodes and A as the set of directed arcs. Each arc $(i, j) \in A$ has cost c_{ij} per unit flow on that arc. Furthermore, the flow cost varies linearly with the amount of flow. Define u_{ij} (l_{ij}) as the maximum (minimum) amount of flow on arc $(i, j) \in A$. Finally, associate integer b_i for each $i \in N$, which can be seen as the demand (supply) of such a node if b_i is negative (positive). If $b_i = 0$, node i is a transshipment node.

The minimum cost flow problem, in which x_{ij} represents the flow on an arc $(i, j) \in A$, can be formulated as follows:

$$\min \quad \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{1.1}$$

$$\text{s.t.} \quad \sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = b_i \quad \forall i \in N, \tag{1.2}$$

$$l_{ij} \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A. \tag{1.3}$$

In the above formulation the total costs are minimized such that the inflow minus the outflow in each node is equal to its demand/supply and the flow on each arc is between its lower and upper capacity. Like in Ahuja *et al.* (1993), we will call constraints (1.2) the *mass balance constraints* and constraints (1.3) the *flow bound constraints*.

Notice that integer requirements on the decision variables are not necessary. Since the constraint matrix is totally unimodular (see e.g. Wolsey (1998)) and the right-hand side are integers, every basic solution is integral. In other words, solving the problem as a linear program with a simplex method always yields an integral solution. However, in general, minimum cost flow problems are not solved as linear programs but by specialized

algorithms. These algorithms run in polynomial time, but have a lower complexity than the fastest linear programming algorithms.

A lot of well-known problems are special cases of the minimum cost flow problem. We will discuss here two of them, which are relevant to this thesis: the shortest path problem and the linear assignment problem.

In the shortest path problem, the shortest path, i.e. the one with the lowest costs, between two nodes s and t has to be found. It can be easily seen that this is a special case of the minimum cost flow problem, namely take $b_s = 1$, $b_t = -1$, and $b_i = 0$ for all $i \in N \backslash \{s, t\}$ and set $l_{ij} = 0$ and $u_{ij} = 1$ for each arc $(i, j) \in A$. Then one unit of flow will be sent through the network from s to t along the shortest path.

The linear assignment problem can be defined as follows: find one-to-one assignments of objects in a set N_1 with those in a set N_2, where $|N_1| = |N_2|$, such that the total costs are minimized. The linear assignment problem is a minimum cost flow problem in a network $G = (N_1 \cup N_2, A)$, where A is the set of all possible assignments, with $b_i = 1$ for all $i \in N_1$, $b_i = -1$ for all $i \in N_2$, and $l_{ij} = 0$ and $u_{ij} = 1$ for all $(i, j) \in A$.

In the remainder of this thesis, the shortest path problem and the linear assignment problem will return as subproblems of more complicated optimization problems.

1.3.2 Multicommodity Flow Problem

The *multicommodity flow problem* is an extension of the minimum cost flow problem, since instead of a single commodity several commodities use the same underlying network. Different commodities have different origins and destinations, and commodities have separate mass balance constraints at each node. However, the coupling of the commodities is caused by the arc capacities (flow bound constraints). In the remainder of this subsection, we assume that the flow must be integral.

The decision version of the multicommodity (integral) flow problem has been proven to be NP-complete (see Garey & Johnson (1979)) if there are at least two commodities. Therefore, it is very unlikely that there exists an algorithm running in polynomial time, to solve this problem. This is the main difference with the single commodity case, which as stated earlier can be solved in polynomial time.

Below we give the formulation of the multicommodity formulation with K as the set of commodities and without lower bound constraints. The notation is similar as in the previous subsection except that there is an extra index k for each commodity in b_i^k and c_{ij}^k, and in the decision variables x_{ij}^k.

$$\min \quad \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k \tag{1.4}$$

$$\text{s.t.} \quad \sum_{\{j:(i,j) \in A\}} x_{ij}^k - \sum_{\{j:(j,i) \in A\}} x_{ji}^k = b_i^k \qquad \forall i \in N, \forall k \in K, \tag{1.5}$$

$$\sum_{k \in K} x_{ij}^k \leq u_{ij} \qquad \forall (i,j) \in A, \tag{1.6}$$

$$x_{ij}^k \geq 0 \quad \text{integer} \quad \forall (i,j) \in A, \forall k \in K. \tag{1.7}$$

1.3.3 Set Partitioning Problem

The (decision version of the) *set partitioning problem* can be formally defined as follows (see Lenstra & Rinnooy Kan (1979)).

Definition 1.4 *Given a finite set S and a finite family F of subsets of S, does F include a subfamily F' of pairwise disjoint sets such that $\cup_{S' \in F'} S' = S$?*

This problem is also known as the exact cover problem. Furthermore, it has been proven that this problem is NP-complete if there are at least three elements in each family F (see Garey & Johnson (1979)). Therefore, the corresponding optimization problem, where the subfamily F' with the lowest costs has to be found and each subset $f \in F$ has costs c_f, is NP-hard. Let a_{if} be 1 if subset f contains element i for each $f \in F$ and $i \in S$ and 0 otherwise. Furthermore, by using decision variables x_f equal to 1 if subset f is in the optimal solution and 0 otherwise, the set partitioning problem can be formulated as follows:

$$\min \quad \sum_{f \in F} c_f x_f \tag{1.8}$$

$$\text{s.t.} \quad \sum_{f \in F} a_{if} x_f = 1 \qquad \forall i \in S, \tag{1.9}$$

$$x_f \in \{0,1\} \quad \forall f \in F. \tag{1.10}$$

Many real-world problems can be formulated as set partitioning problems. In the remainder of this thesis, we will provide some examples. They often have a huge amount of variables.

1.3.4 Set Covering Problem

The *set covering problem* is slightly different from the set partitioning problem. Each element must be in at least one subset instead of in exactly one. It is obvious that each feasible solution of the set partitioning problem is also a feasible solution of the

set covering problem. Furthermore, the optimal solution (and thus each lower bound) of the set covering problem is a lower bound of the set partitioning problem. This fact will be used several times in this thesis, namely when we calculate lower bounds on set partitioning problems by computing lower bounds on the corresponding set covering problem, i.e. replacing the equality sign in constraints (1.9) by a "$\geq$" sign.

1.4 Combinatorial Optimization Techniques

In this section we discuss two important combinatorial optimization techniques, *Lagrangian relaxation* and *column generation*, that will be used in the remainder of this thesis. Furthermore, we discuss the combination of both techniques.

1.4.1 Lagrangian Relaxation

Lagrangian relaxation can be used to obtain lower bounds on the optimal solution value of a combinatorial optimization problem. Notice hereby that we assume a minimization problem. The idea of Lagrangian relaxation is to relax some of the difficult constraints and penalize their violations by a certain weight in the objective function. Here, we will explain the technique by applying it to the general integer programming problem given below. The set of constraints is split into two sets, hard and easy constraints (N^1 and N^2, respectively).

$$\text{(P)} \quad \min \quad \sum_{j \in J} c_j x_j \tag{1.11}$$

$$\text{s.t.} \quad \sum_{j \in J} a_{ij} x_j = b_i \qquad \forall i \in N^1, \tag{1.12}$$

$$\sum_{j \in J} d_{kj} x_j = e_k \qquad \forall k \in N^2, \tag{1.13}$$

$$x_j \geq 0 \quad \text{integer} \quad \forall j \in J. \tag{1.14}$$

We introduce a *Lagrangian multiplier* λ_i for each hard constraint i. That is, we delete those constraints from the model and penalize them in the objective function. The Lagrangian subproblem can then be formulated as follows:

$$\Phi(\lambda) = \quad \min \quad \sum_{j \in J} c_j x_j + \sum_{i \in N^1} \lambda_i (b_i - \sum_{j \in J} a_{ij} x_j) \tag{1.15}$$

$$\text{s.t.} \quad \sum_{j \in J} d_{kj} x_j = e_k \qquad \forall k \in N^2, \tag{1.16}$$

$$x_j \geq 0 \quad \text{integer} \quad \forall j \in J. \tag{1.17}$$

$\Phi(\lambda)$ is a lower bound on the optimal solution value of the original problem for each vector $\lambda = (\lambda_i)_{i \in N^1}$, since each feasible solution in the original problem is also feasible in the Lagrangian subproblem and $\Phi(\lambda)$ is equal to the objective value of such a feasible solution. In a similar way, we can deal with inequality constraints in set N^1. The corresponding multipliers are then restricted in sign, i.e. in case of "$\leq$" constraints, $\lambda_i \geq 0, \forall i \in N^1$ and in case of "$\geq$" constraints, $\lambda_i \leq 0, \forall i \in N^1$.

Since each vector λ yields a lower bound, the best lower bound is obtained by solving the *Lagrangian dual problem* $\max_\lambda \Phi(\lambda)$. The remaining questions are then (1) "How can we obtain this best (or at least a good) lower bound?" and (2) "What is the quality of that lower bound compared to other lower bounds?". First, we will answer the second question, where we will compare the best Lagrangian lower bound, denoted by $Z(LR)$, with the lower bound obtained from the LP-relaxation of (P), i.e. removing the integrality requirements on the variables, denoted by $Z(LP)$.

Theorem 1.5 $Z(LR) \geq Z(LP)$

Proof. We first repeat the definition of $Z(LR)$:

$$\begin{aligned} \max_\lambda \{ & \min \sum_{i \in N^1} b_i \lambda_i + \sum_{j \in J} (c_j - \sum_{i \in N^1} a_{ij} \lambda_i) x_j | \\ & \sum_{j \in J} d_{kj} x_j = e_k, \forall k \in N^2; x_j \geq 0 \quad \text{integer}, \forall j \in J \}. \end{aligned} \tag{1.18}$$

Below, we rewrite $Z(LP)$, where we only consider the relevant case that the feasible region is non-empty and bounded:

$$\begin{aligned} Z(LP) \quad = \quad & \{ \min \sum_{j \in J} c_j x_j | \sum_{j \in J} a_{ij} x_j = b_i, \forall i \in N^1; \\ & \sum_{j \in J} d_{kj} x_j = e_k, \forall k \in N^2; x_j \geq 0, \forall j \in j \} \\ =^{(1)} \quad & \{ \max_\lambda \sum_{i \in N^1} b_i \lambda_i + \sum_{k \in N^2} e_k \mu_k | \\ & \sum_{i \in N^1} a_{ij} \lambda_i + \sum_{k \in N^2} d_{kj} \mu_k \leq c_j, \forall j \in J \} \\ =^{(2)} \quad & \{ \max_\lambda \sum_{i \in N^1} b_i \lambda_i + \{ \max_\lambda \sum_{k \in N^2} e_k \mu_k | \\ & \sum_{k \in N^2} d_{kj} \mu_k \leq c_j - \sum_{i \in N^1} a_{ij} \lambda_i, \forall j \in J \} \} \\ =^{(3)} \quad & \{ \max_\lambda \sum_{i \in N^1} b_i \lambda_i + \{ \min \sum_{j \in J} (c_j - \sum_{i \in N^1} a_{ij} \lambda_i) x_j | \\ & \sum_{j \in J} d_{kj} x_j = e_k, \forall k \in N^2; x_j \geq 0, \forall j \in J \} \} \end{aligned}$$

$$=^{(4)} \quad \{\max_{\lambda}\{\min \sum_{i\in N^1} b_i\lambda_i + \sum_{j\in J}(c_j - \sum_{i\in N^1} a_{ij}\lambda_i)x_j| \\ \sum_{j\in J} d_{kj}x_j = e_k, \forall k \in N^2; x_j \geq 0, \forall j \in J\}\}. \tag{1.19}$$

The first equality sign is obtained by LP-duality; the second by nesting the maximization problem; the third by applying LP-duality to the nested problem and the fourth by rewriting the problem.

Since the only difference between (1.18) and (1.19) is that the feasible region is restricted to integer solutions or not, $Z(LR) \geq Z(LP)$. Furthermore, it follows that this is an equality sign if the integrality conditions in the Lagrangian subproblem have no effect. A similar proof was originally proposed by Geoffrion (1974). ■

We still need to answer the first question "How can we compute the optimal vector of Lagrangian multipliers λ^* that yields the best Lagrangian lower bound?". Since this maximization problem is nondifferentiable, an iterative procedure, called *subgradient optimization*, is mostly used to obtain λ^* for integer programming problems. In this context, this procedure was introduced by Held & Karp (1971).

Step 0: Initialization
Initialize parameters $n_{\max}, \alpha^0, \gamma, \epsilon$.
Compute an upper bound UB.
Start with initial multipliers λ^0.
Step 1: Solve Lagrangian subproblem
Compute the lower bound $\Phi(\lambda^n)$ with optimal solution $\mathbf{x}^n$.
Compute the subgradient $y_i^n := b_i - \sum_{j\in J} a_{ij}x_j^n$.
Step 2: Compute Lagrangian multipliers
Compute $\lambda_i^{n+1} := \lambda_i^n + \alpha^n \frac{UB-\Phi(\lambda^n)}{\sum_{i\in N^1}(y_i^n)^2} y_i^n$
Step 3: Update parameters
If $\Phi(\lambda^n)$ is an improvement: $m := 0$; otherwise $m := m + 1$.
If $m = \gamma$, then $\alpha^{n+1} := \alpha^n/2$; otherwise $\alpha^{n+1} := \alpha^n$.
Step 4: Termination Criterion
If $UB = \Phi(\lambda^n), \sum_{i\in N^1}(y_i^n)^2 \leq \epsilon, \alpha^n \leq \epsilon$ or $n \geq n_{\max}$,then stop; otherwise, return to Step 1.

Figure 1.5: Subgradient Algorithm

In Figure 1.5, we describe in detail the version of the subgradient algorithm, which we will use later on in this thesis. We initialize the algorithm with some initial multipliers and an upper bound on the objective value. This can be obtained by applying a, mostly simple, heuristic to the problem. In step 1 the Lagrangian subproblem is solved and a subgradient is calculated. This vector is used to update the multipliers in the direction

of the subgradient. The step size is determined by the difference between the lower and upper bound, the norm of the subgradient and a certain parameter α. This parameter is updated in step 3 in such a way that convergence of the subgradient algorithm can be proved. That is, α is halved after a certain number of iterations (γ) without improvement in the lower bound. Finally, the procedure terminates when the best lower bound is found, i.e. the subgradient vector is the 0-vector and/or the upper bound is equal to the lower bound, α is very small or a maximum number of iterations ($n_{\max}$) is reached.

Notice that in every iteration, we get a solution, which is, in general, infeasible, since some of the constraints are relaxed. However, we can often transform such an infeasible solution into a feasible one by use of a heuristic. Such a heuristic is called a *Lagrangian heuristic*. Of course, it is also possible to apply such a heuristic in each iteration (or in some iterations) such that the upper bound can be improved during the iterative procedure.

We refer the reader who likes to know more about Lagrangian relaxation and subgradient optimization, to surveys on these topics by Geoffrion (1974), Fisher (1981) and Beasley (1995).

We will now return to the set partitioning problem introduced in the previous section. The Lagrangian subproblem of the set partitioning problem where the constraints (1.9) are relaxed and a multiplier λ_i is introduced for each constraint i, can thus be written as:

$$\Phi(\lambda) = \quad \min \quad \sum_{i \in S} \lambda_i + \sum_{f \in F} (c_f - \sum_{i \in S} \lambda_i a_{if}) x_f \tag{1.20}$$

$$\text{s.t.} \quad x_f \in \{0,1\} \quad \forall f \in F. \tag{1.21}$$

From this, the optimal solution follows immediately:

$$x_f = \begin{cases} 1, \text{ if } c_f - \sum_{i \in S} \lambda_i a_{if} < 0 \\ 0, \text{ otherwise} \end{cases} \qquad \forall f \in F. \tag{1.22}$$

Since the optimal solution of the Lagrangian subproblem without integrality constraints is already integral, it follows that the best Lagrangian lower bound is equal to the value of the LP-relaxation (see last line of the proof of Theorem 1.5).

1.4.2 Column Generation

Column generation is a technique that is used for problems with a huge number of variables. The general idea, introduced by Dantzig & Wolfe (1960), is to solve a sequence of reduced problems, where each reduced problem contains only a small portion of the set of variables (columns). After a reduced problem is solved, a new set of columns is obtained by using dual information of the solution. The column generation algorithm converges once it has established that the optimal solution based on the current set of columns cannot be improved upon by adding more columns. Then the optimal solution

of the reduced problem is the optimal solution of the overall problem. We will refer to the reduced problem as the *master problem* and to the problem of generating a new set of columns as the *pricing problem.*

Before we continue our discussion on integer programming problems, we will explain the concept in more detail using the following, general linear programming problem (P') with a huge number of variables $|J|$:

$$(P') \quad \min \quad \sum_{j \in J} c_j x_j \tag{1.23}$$

$$\text{s.t.} \quad \sum_{j \in J} a_{ij} x_j = b_i \quad \forall i \in N, \tag{1.24}$$

$$x_j \geq 0 \quad \forall j \in J. \tag{1.25}$$

If we apply column generation to solve (P'), we start with an initial set of columns $J^0 \subseteq J$ such that these columns contain a feasible solution of the reduced problem. We solve this problem and compute the optimal values of the dual variables. These dual variables are needed in the pricing problem to select new columns with negative reduced costs, which will be added to the set J^0. The reduced cost of a column is defined as $c_j - \sum_{i \in N} a_{ij} u_i$, where u_i is the optimal value of the dual variable corresponding to each constraint in (1.24). If there are no columns left with negative reduced costs, it can be easily seen that the optimal solution of the reduced problem is also the optimal solution of (P'). This follows from the rules of the well known simplex method to solve linear programs (see e.g. Chvátal (1983)). Notice that if $\min_{j \in J}\{c_j - \sum_{i \in N} a_{ij} u_i\} > 0$, all columns have positive reduced costs. Due to this trivial observation, it is not necessary that all columns are explicitly known. The pricing problem can be seen as an optimization problem by itself and columns can be generated implicitly by solving this optimization problem. In our applications, these problems are most often shortest path types of problems (simple shortest path or shortest path problems with resource constraints, where the reader should notice that the first one can be solved in polynomial time, while the second one is, in general, NP-hard).

If (some of) the decision variables need to be integral, branch-and-bound algorithms are most often used to solve the resulting (mixed-)integer programming problem. Column generation can be embedded in the branch-and-bound tree by solving the relaxations in each node with a column generation algorithm. This solution approach is called *branch-and-price*. For a general discussion on column generation in an integer programming context, we refer to Barnhart *et al.* (1998). In almost all papers column generation is used to solve LP-relaxations, however, it can also be combined with Lagrangian relaxation. This is the topic of Subsection 1.4.3.

The technique has been frequently applied to solve a wide variety of applications. For an overview on these applications we refer to Lübbecke & Desrosiers (2002). About half of the papers mentioned there is in the area of transportation, where the reader can think

of problems as vehicle scheduling and crew scheduling. Since these topics are extensively discussed in Chapter 2, we focus here on two other applications: one in vehicle routing and one in railway planning. The first problem is the vehicle routing problem with time windows and the second one is the shunting problem.

Vehicle Routing with Time Windows

The *vehicle routing problem with time windows* consists of designing a set of routes for a fleet of vehicles of minimal costs, such that all customers (with known demand) are served exactly once within their predefined time window, the capacity of the vehicle is not exceeded and the route starts and ends in the depot. This problem is extensively discussed in the literature, see e.g. Desrosiers *et al.* (1995) and Bodin *et al.* (1983) for surveys.

Desrochers *et al.* (1992b) first applied column generation techniques to this particular problem. However, notice that similar ideas had been applied to several other variants of vehicle routing problems by the same group of authors since 1984 (see e.g. Desrosiers *et al.* (1984)). They formulate the problem as a set partitioning problem and show that the LP-relaxation of this formulation is tight. However, they apply the column generation technique to a set covering formulation of this problem, since that is computationally more stable. Feasible columns are added by solving a shortest path problem with time windows and capacity constraints using a dynamic programming algorithm. They use the columns generated while solving the LP-relaxation to obtain an integer solution using a branch-and-bound algorithm. That means that no columns were generated after solving the LP-relaxation. The results show that the value of the LP-relaxation is mostly very close to the feasible solution they found. These gaps are about 1.5% on average, with a maximum of 12.1%.

Shunting Problem

Within the rush hours, the rolling stock of a passenger railway operator is typically operating the timetable or it is in maintenance. However, outside the rush hours, an operator usually has a surplus of rolling stock. These surplus train units can be parked at a shunt yard in order to be able to fully exploit the main railway infrastructure. Especially during the night, many passenger train units have to be parked, since usually there are just a few night trains. In the Netherlands, mainly freight trains operate at night. The process of parking train units together with several related processes is called *shunting*. Notice that the term shunting is also used in the context of cargo trains, which is a different problem in many aspects. For a more comprehensive description of the problem we refer to Freling *et al.* (2002).

The shunting problem can be divided into two steps. In the first step arriving trains

have to be assigned to departing trains which results in so called blocks and in the second step these blocks have to be parked at a shunt track. For each block we know the arrival and departure times and the arrival and departure platforms. In addition, we also have the costs of assigning a block to a shunt track. (These costs consist of several components, which fall outside the scope of this discussion.) Given this information, the second step in the solution approach consists of assigning blocks to shunt tracks at minimum costs, such that the capacity of the shunt tracks is never exceeded and that there are no crossings among the blocks, where a crossing occurs if one block obstructs the arrival or departure of another block.

This latter problem can be formulated as a set partitioning problem for which the LP-relaxation can be solved by a column generation algorithm (see Freling *et al.* (2002)). The master problem is solved by linear programming and the pricing problem by dynamic programming techniques, where shortest path problems with resource constraints have to be solved. Furthermore, they introduce an efficient solution technique to solve the pricing problem. That is, so-called "not-nodes" are introduced in the network such that the number of arcs is linear instead of quadratic in the number of nodes and therefore, the pricing problem can be solved faster. For details, we refer again to Freling *et al.* (2002). However, it should be mentioned that this idea can only be used with certain special cost functions on the arcs in the network.

1.4.3 Column Generation in Combination with Lagrangian Relaxation

As mentioned earlier column generation is usually used in the context of linear programming, but in this thesis it will be used in combination with Lagrangian relaxation (see also Freling (1997) and Carraresi *et al.* (1995)) to compute lower bounds on the optimal solution. Furthermore, we will use the columns which are generated to compute the lower bound to construct a feasible solution. This approach has been chosen to solve the different models for the following reasons.

1. For these problems, a set of columns generated to compute the lower bound turns out to be a set from which we can select a reasonably good feasible solution.

2. Since we compute a lower bound on the optimal solution, we obtain an indication about the quality of the constructed feasible solution.

3. Lagrangian relaxation has shown to provide tight bounds for set partitioning type of problems (see e.g. Beasley (1995)).

4. Because of the number of constraints, using a linear programming relaxation is not a realistic option for some of the proposed models.

Now, we discuss how column generation can be used in Lagrangian relaxation. Therefore, we use the set partitioning problem, which has been discussed before. Recall that the Lagrangian subproblem can be written as:

$$\begin{aligned} \Phi(\lambda) = \quad & \min \quad \sum_{i \in S} \lambda_i + \sum_{f \in F} (c_f - \sum_{i \in S} \lambda_i a_{if}) x_f \\ & \text{s.t.} \quad x_f \in \{0,1\} \quad \forall f \in F. \end{aligned}$$

The reduced cost of a column is now defined as $c_f - \sum_{i \in S} \lambda_i a_{if}$, i.e. it is equal to the Lagrangian cost. Before, we generate new columns we use the subgradient algorithm to get good multipliers λ such that we obtain a good lower bound. Furthermore, to prevent that columns are generated twice, the Lagrangian multipliers should be modified before generating new columns. Suppose that in a certain iteration of the column generation algorithm, the set F' contains the columns used in the master problem. The greedy heuristic described in Figure 1.6 (see also Freling (1997) and Carraresi *et al.* (1995)) modifies the multipliers such that columns in F' have non-negative reduced costs and the value of the Lagrangian function does not decrease.

For each column $f \in F'$ with $c_f - \sum_{i \in S} \lambda_i a_{if} < 0$:
$\delta := \frac{c_f - \sum_{i \in S} \lambda_i a_{if}}{\sum_{i \in S} a_{if}}$;
for each $i \in S$ with $a_{if} = 1$: $\lambda_i := \lambda_i + \delta$;
update the reduced costs for all columns $g \in F'$ and $g > f$.

Figure 1.6: Greedy heuristic to modify Lagrangian multipliers

In each iteration of the column generation algorithm, a lower bound on the overall problem can be easily computed if all columns with negative reduced costs are explicitly available. Since for each λ,

$$\Phi'(\lambda) = \min\{\sum_{i \in S} \lambda_i + \sum_{f \in F'} (c_f - \sum_{i \in S} \lambda_i a_{if}) x_f | x_f \in \{0,1\}, \forall f \in F'\}$$

is a lower bound for the current set of columns, the expression

$$\Phi'(\lambda) + \sum_{f \in F \setminus F'} \min(c_f - \sum_{i \in S} \lambda_i a_{if}, 0)$$

is a lower bound on the optimal value of the overall problem for each vector λ. This can be easily seen by rewriting $\Phi(\lambda)$:

$$\begin{aligned}
\Phi(\lambda) &= \min\{\sum_{i\in S}\lambda_i + \sum_{f\in F}(c_f - \sum_{i\in S}\lambda_i a_{if})x_f | x_f \in \{0,1\}, \forall f \in F\} \\
&= \min\{\sum_{i\in S}\lambda_i + \sum_{f\in F'}(c_f - \sum_{i\in S}\lambda_i a_{if})x_f | x_f \in \{0,1\}, \forall f \in F' \\
&\quad + \sum_{f\in F\setminus F'}(c_f - \sum_{i\in S}\lambda_i a_{if})x_f | x_f \in \{0,1\}, \forall f \in F\setminus F'\} \\
&= \Phi'(\lambda) + \sum_{f\in F\setminus F'} \min(c_f - \sum_{i\in S}\lambda_i a_{if}, 0).
\end{aligned}$$

1.5 Overview of the Thesis

In this thesis we consider mathematical formulations and algorithms to solve such integrated problems. Moreover, we consider the aspect of dynamic planning, i.e. replanning during the day, to prevent delays. The thesis is set up as follows.

In Chapter 2 we discuss the traditional sequential approach, which means that first the vehicle scheduling problem and afterwards the crew scheduling problem is solved. We discuss the basic models and algorithms which are used in the literature. We use some of those ideas in the more complex problems in Chapters 3 and 4, where the integrated vehicle and crew scheduling problem is discussed.

Chapter 3 deals with models, relaxations and algorithms for an integrated approach to vehicle and crew scheduling for an urban mass transit system with a single depot. We discuss potential benefits of integration and provide an overview of the literature where mainly partial integration is considered. Our approach is new in the sense that we can tackle integrated vehicle and crew scheduling problems of practical size. We propose new mathematical formulations for integrated vehicle and crew scheduling problems and we discuss corresponding Lagrangian relaxations and Lagrangian heuristics. To solve the Lagrangian relaxations, we use column generation applied to set partitioning type of models. The chapter is concluded with a computational study using real life data, which shows the applicability of the proposed techniques to practical problems. Furthermore, we also address the effectiveness of integration in different situations.

Chapter 4 presents two different models and algorithms for integrated vehicle and crew scheduling in the multiple-depot case. The algorithms are both based on a combination of column generation and Lagrangian relaxation. Furthermore, we compare those integrated approaches with each other and with the traditional sequential one on randomly generated as well as real-world data instances for a suburban/extra-urban mass transit system. To simulate such a transit system, we propose a new way of randomly generating data instances such that their properties are the same as for our real-world instances.

A solution approach to the dynamic vehicle scheduling problem and the dynamic vehicle and crew scheduling problem is presented in Chapter 5. This approach consists of solving a sequence of optimization problems, where we take into account different scenarios for future travel times. We discuss the potential benefit of our approach compared to the traditional one, where the vehicle (and crew) scheduling problem is solved only once for a whole period and the travel times are assumed to be fixed. Afterwards, we discuss separately the dynamic vehicle scheduling problem and the dynamic vehicle and crew scheduling problem. In the first part on dynamic vehicle scheduling, we use a "cluster-reschedule" heuristic where we first assign trips to depots by solving the static problem and then solve dynamic single-depot problems, since in the multiple-depot case we cannot solve the overall problem exactly. We use new mathematical formulations of these problems that allow fast solutions by standard optimization software. Results of a computational study with real life data are presented, in which we compare different variants of our approach and perform a sensitivity analysis with respect to deviations of the actual travel times from estimated ones. In the second part of this chapter, we extend the ideas of dynamic vehicle scheduling to dynamic vehicle and crew scheduling. That is we combine those ideas with the ones from the chapters on integrated vehicle and crew scheduling.

Each chapter is finished with an application from a public transport company. To be more precise, in Chapter 2 we solve a crew scheduling problem from Stadsbus Maastricht. We show in Chapter 3 that all kind of crew rules can be implemented in an integrated approach and that such an approach leads to improvements over the sequential approach. This is demonstrated with data provided by the RET. In Chapters 4 and 5, we use data from Connexxion. In the first one, we show that large real-world instances can be solved by an integrated approach by splitting up the problem, while in the second one we discuss the impact of several strategies to reduce delays for the same real-world problem.

Finally, Chapter 6 contains a summary and some concluding remarks.

Chapter 2

Traditional Vehicle and Crew Scheduling

This chapter deals with the "traditional" sequential approach to vehicle and crew scheduling, where first the vehicle scheduling problem is solved and afterwards the crew scheduling problem. We will discuss some models and algorithms for the vehicle scheduling as well as the crew scheduling problem. Furthermore, we will provide a literature review on both problems.

The discussion about the vehicle and crew scheduling problem is not only interesting in itself, but it also provides a useful introduction to the integrated problem, which will be discussed in the next chapters, for two reasons. First of all, the models and algorithms for the integrated problem are based on the traditional models and algorithms of the underlying problems. Secondly, we like to compare those integrated approaches with a traditional sequential approach. For this comparison, we will use the models and algorithms presented in this chapter.

The chapter is organized as follows. The discussion on the vehicle scheduling problem is split into two sections, one dealing with the single-depot case (Section 2.1) and the other one with the multiple-depot case (Section 2.2). Next, in Section 2.3 we discuss the crew scheduling problem. Finally, we will conclude this chapter in Section 2.4 with some computational results for an application of crew scheduling at Stadsbus Maastricht.

2.1 Single-Depot Vehicle Scheduling

Vehicle scheduling is the process of assigning vehicles to trips such that the total vehicle costs are minimal. Hereby, it is assumed that the start and end time as well as the start and end location of all trips are fixed. In the case that there is only one depot, all vehicles are identical and there are no time constraints, we have the standard *single-depot vehicle scheduling problem* (SDVSP). A solution to the SDVSP is feasible if all trips are assigned

to a vehicle and if each vehicle starts in the depot, performs a sequence of trips and ends again in the depot. Furthermore, if two trips i and j are consecutively assigned to the same vehicle, the start time of trip j should be larger than or equal to the end time of trip i plus the travel time from the end location of trip i to the start location of trip j. We will call such trips i and j *compatible*. The vehicle costs consist of a fixed component for every vehicle and variable costs for idle and travel time.

Recall from Chapter 1 that a deadhead is a period that a vehicle is moving to or from the depot, or a period between two trips where a vehicle is outside of the depot (possibly moving without passengers). In most practical cases, it is allowed that a vehicle returns to its own depot between two trips if there is enough time to do this. In some of the algorithms, we will make use of this fact.

It is well known that the SDVSP can be solved in polynomial time (see e.g. Freling *et al.* (2001b)). Although the problem can be solved in polynomial time, there has been a lot of attention to developing very fast algorithms to solve this problem in the last decade (we refer to Desrosiers *et al.* (1995) for an overview on models and algorithms described in the older literature). This is due to the fact that the SDVSP is not only an interesting problem in itself, but it also appears as a subproblem in much more complicated problems like the multiple-depot vehicle scheduling problem and the integrated vehicle and crew scheduling problem. One of these recently developed algorithms is proposed by Löbel (1997) and is based on single-commodity flow theory. Freling *et al.* (2001b) describe an efficient auction algorithm, which will be used later on in this thesis to solve several SDVSP's which appear as subproblems in more complicated problems. Therefore, we will summarize this algorithm in Subsection 2.1.2. However, we will first introduce some notation and a mathematical model for the SDVSP.

2.1.1 Model

Let $N = \{1, 2, ..., n\}$ be the set of trips, numbered according to increasing starting time, and let $E = \{(i, j) \mid i < j,\ i, j \text{ compatible},\ i \in N, j \in N\}$ be the set of deadheads. Let r and t both represent the depot. We define the vehicle scheduling network $G = (V, A)$, which is an acyclic directed network with nodes $V = N \cup \{r, t\}$, and arcs $A = E \cup (r \times N) \cup (N \times t)$. A path from r to t in the network represents a feasible schedule for one vehicle, and a complete feasible vehicle schedule is a set of disjoint paths from r to t such that each node in N is covered. Let c_{ij} be the vehicle cost of arc $(i, j) \in A$, which is usually some function of travel and idle time. Furthermore, fixed costs for using a vehicle can be added to the cost of arcs (r, i) or (j, t) for all $i, j \in N$.

Using decision variables y_{ij}, with $y_{ij} = 1$ if a vehicle covers trip j directly after trip i, $y_{ij} = 0$ otherwise, the SDVSP can be formulated as follows as a quasi-assignment problem.

(SDVSP):

$$\min \quad \sum_{(i,j)\in A} c_{ij} y_{ij} \tag{2.1}$$

$$\text{s.t.} \quad \sum_{\{j:(i,j)\in A\}} y_{ij} = 1 \qquad \forall i \in N, \tag{2.2}$$

$$\sum_{\{i:(i,j)\in A\}} y_{ij} = 1 \qquad \forall j \in N, \tag{2.3}$$

$$y_{ij} \in \{0,1\} \quad \forall (i,j) \in A. \tag{2.4}$$

In terms of the underlying network, $y_{ij} = 1$ means that the corresponding arc $(i,j) \in A$ is used by a vehicle. Constraints (2.2) and (2.3) assure that each trip is assigned to exactly one predecessor and one successor, that is, these constraints guarantee that the network is partitioned into a set of disjoint paths from r to t. The binary condition on the variables can be relaxed to $y_{ij} \geq 0$ because it is well-known that a simplex algorithm produces optimal 0/1-solutions for the LP-relaxation.

A bottleneck for solving SDVSP when using a network such as G is the large size of those networks due to a large number of arcs in E. When a vehicle has an idle time between two consecutive trips which is long enough to let it drive to the depot, we assume that is does so. In that case the arc between the trips is called a *long arc*; the other arcs between trips are called *short arcs.*

2.1.2 Auction Algorithm

The auction algorithm proposed by Freling (1997) (see also Freling *et al.* (2001b)) is a combined forward and reverse auction algorithm for the quasi-assignment problem. The algorithm is straightforwardly adapted from an algorithm proposed by Bertsekas & Castañon (1992) for several other type of assignment problems. In this subsection we will only sketch the idea of the algorithm. For a more detailed description of the algorithm and for a proof of the correctness we refer to Freling (1997).

The auction algorithm consists of forward and backward iterations. In a forward iteration trips are forward assigned (a trip i is forward assigned to a trip j if $y_{ij} = 1$) and in a backward iteration trips are backward assigned (a trip i is backward assigned to a trip j if $y_{ji} = 1$). In a forward assignment each candidate trip i is assigned to another trip j or to the sink t with the maximum corresponding value. This value is a kind of measure for the attractiveness of this assignment. The backward assignment is a similar but reverse procedure.

2.2 Multiple-Depot Vehicle Scheduling

In the *multiple-depot vehicle scheduling problem* (MDVSP), the total vehicle costs have to be minimized subject to the following constraints:

- every vehicle is associated with a single depot;
- every trip has to be assigned to exactly one vehicle;
- some trips have to be assigned to vehicles from a certain subset of depots.

In some cases there are also other constraints, such as depot capacity constraints, which specify for every depot a maximum number of vehicles. We do not consider these type of constraints in this section. However, they can easily be incorporated.

The vehicle costs can be defined in the same way as in the single-depot case (see Section 2.1). Again, it is allowed that a vehicle returns to its own depot between two trips if there is enough time to do this.

2.2.1 Literature Review

The MDVSP is shown to be NP-hard by Bertossi *et al.* (1987) if there are at least two depots. Moreover, Löbel (1997) shows that even ϵ-approximation of the MDVSP is NP-hard. Therefore, it took until 1989 before exact methods where used to solve this problem. Since then, the models in the literature can be classified into three basic types:

1. single-commodity flow formulations;
2. multicommodity flow formulations;
3. set partitioning formulations.

The first exact algorithms were based on models of the first category (e.g. Carpenato *et al.* (1989)). The disadvantage of these kind of formulations is the weakness of the LP-relaxation. Recently, Fischetti *et al.* (2001) have developed a branch-and-cut algorithm to overcome this weakness. Their results seems to be very promising, especially for more realistic instances, where the average number of trips per vehicle is large.

Most recent algorithms are based on formulations of the second type (see e.g. Forbes *et al.* (1994), Mesquita & Paixão (1999), Löbel (1997) and Löbel (1998)). Sometimes these formulations use two types of variables (one for the flow and one for the assignment of a trip to a depot), while other times only one type of variable is used by substitution of the assignment variables. Mesquita & Paixão (1999) discuss both variants and prove that the LP-relaxations of these formulations are equivalent. Furthermore, they prove that those LP-relaxations are tighter than the ones from single-commodity flow formulations.

Almost all successful implementations in practice are based on models of this type (see e.g. Löbel (1997) who solves very large real-world problems).

The third category consists of the models based on set partitioning formulations that have a very large (exponential) number of variables. These models are used by, among others, Ribeiro & Soumis (1994) and Hadjar *et al.* (2001). Ribeiro & Soumis (1994) prove that the LP-relaxation of their formulation is equal to the LP-relaxation of one of the multicommodity flow formulations. Due to the large number of variables, column generation is necessary to solve these types of models. However, an important disadvantage of such an approach is that the performance decreases significantly if the average number of trips per vehicle is large. To the best of our knowledge these types of models are only used on randomly generated test instances where the average number of trips per vehicle is about 3 or 4. Of course, such instances are not very realistic.

2.2.2 Model

In this subsection, we will present a non-standard mixed integer programming formulation of the MDVSP. The formulation is of the second type, but slightly differs from the formulations proposed before.

Let D and $N = \{1, 2, ..., n\}$ denote the set of depots and trips, respectively. Let N^d be the subset of all trips that are allowed to be assigned to depot $d \in D$, and define D_i as the subset of depots to which trip $i \in N$ can be assigned. Let r^d and t^d denote the source and sink, respectively, of the network corresponding to depot d. This network is denoted by $G^d = (N^d \cup r^d \cup t^d, A^d)$, where A^d is the set of arcs between two compatible trips in N^d, from r^d to every trip in N^d and from every trip in N^d to t^d.

Figure 2.1 shows an example of such a vehicle network with two depots and 7 trips, where $N^1 = \{1, 2, 4, 5, 6\}$ and $N^2 = \{2, 3, 5, 6, 7\}$. Notice that, for reasons of clarity, not all arcs have been drawn.

Let c^d_{ij} be the vehicle cost of arc $(i, j) \in A^d$, which is usually some function of travel and idle time and let c be the fixed vehicle cost.

We could now give a typical multicommodity flow formulation of the MDVSP. We will, however, present a different formulation that, in general, requires much less variables at the expense of a relatively small number of additional constraints. This formulation, which is valid under a certain assumption, turns out to be easier to solve when standard mixed integer programming software is used. The main idea behind the formulation is originally proposed by Haase *et al.* (2001) in the context of integrated vehicle and crew scheduling.

The idea is to reduce the size of the arc sets, which may be very large. First of all, we assume that a vehicle always returns to the depot between two consecutive trips if this is possible and not more expensive. Recall from Subsection 2.1.1 that the arc between such

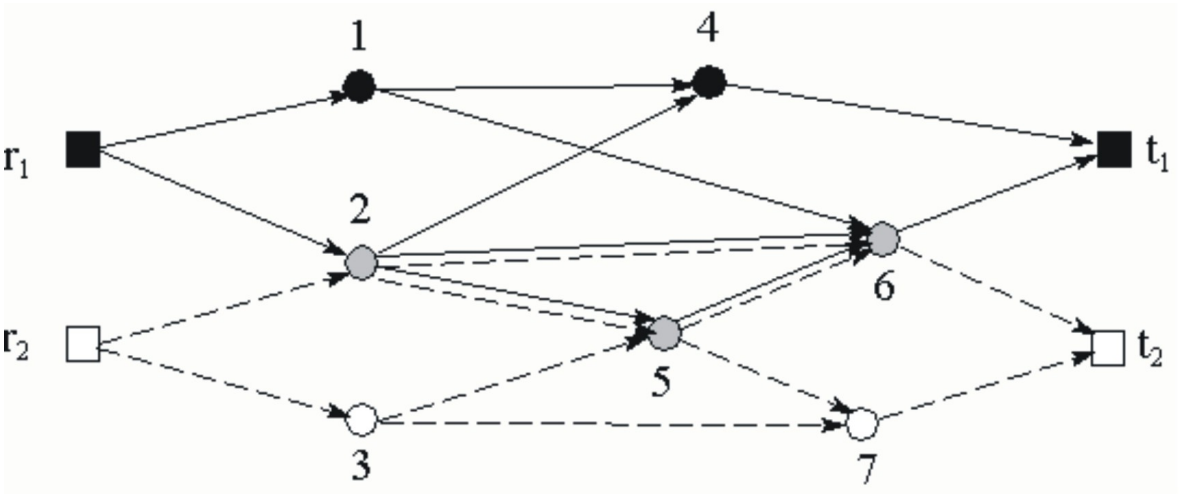

Figure 2.1: Example - Network G^1 and G^2

trips is called a long arc. If we further assume that there are no costs involved when a vehicle is idle at the depot, we can delete all long arcs. Let A^{d*}, $d \in D$, denote the part of A^d without long arcs.

Let tp^{dh}, $h = 1, 2, ..., m$ be the time points at which a vehicle may leave depot d to drive to the start location of a trip, i.e. the start time of the trip minus the driving time from the depot to the start location. Moreover, define H^d as the corresponding set of timepoints with $tp^{d1} < tp^{d2} < ... < tp^{dm}$. Furthermore, define st_i and et_i as respectively the start and ending time of trip i, $trav(r^d, i)$ and $trav(i, t^d)$ as the deadhead travel time from depot d to the start location of trip i and from the end location of trip i to depot d, respectively. For trip i, let parameter b_i^{dh} be equal to 1 if $st_i \leq tp^{dh} < et_i$, and 0 otherwise. Similarly, let parameters

$$a_{ij}^{dh} = \begin{cases} 1, \text{ if } et_i \leq tp^{dh} < st_j, \\ 0 \text{ otherwise,} \end{cases} \quad \text{for each arc } (i,j) \text{ with } i,j \in N^d,$$

$$a_{r_dj}^{dh} = \begin{cases} 1, \text{ if } st_j - trav(r^d, j) \leq tp^{dh} < st_j, \\ 0 \text{ otherwise,} \end{cases} \quad \text{for each arc } (r^d, j) \text{ with } j \in N^d, \text{ and}$$

$$a_{it_d}^{dh} = \begin{cases} 1, \text{ if } et_i \leq tp^{dh} < et_i + trav(i, t^d), \\ 0 \text{ otherwise,} \end{cases} \quad \text{for each arc } (i, t^d) \text{ with } i \in N^d.$$

Using decision variables y_{ij}^d, $(i,j) \in A^{d*}$, with $y_{ij}^d = 1$ if a vehicle from depot d is assigned to arc (i,j), $y_{ij}^d = 0$ otherwise, variables x_i^d, $i \in N^d$, with $x_i^d = 1$ if trip i is assigned to a vehicle from depot d, $x_i^d = 0$ otherwise, and variables B^d to denote the number of vehicles used from depot $d \in D$, the MDVSP can be formulated as follows.

(MDVSP):

$$\min \quad \sum_{d\in D}(cB^d + \sum_{(i,j)\in A^{d*}} c_{ij}^d y_{ij}^d) \tag{2.5}$$

$$\text{s.t.} \quad \sum_{\{i:(i,j)\in A^{d*}\}} y_{ij}^d - x_j^d = 0 \qquad \forall d \in D, \forall j \in N^d, \tag{2.6}$$

$$\sum_{\{j:(i,j)\in A^{d*}\}} y_{ij}^d - x_i^d = 0 \qquad \forall d \in D, \forall i \in N^d, \tag{2.7}$$

$$\sum_{d\in D_i} x_i^d = 1 \qquad \forall i \in N, \tag{2.8}$$

$$\sum_{i\in N} b_i^{dh} x_i^d + \sum_{(i,j)\in A^{d*}} a_{ij}^{dh} y_{ij}^d \leq B^d \qquad \forall d \in D, \forall h \in H^d, \tag{2.9}$$

$$x_i^d \in \{0,1\} \quad \forall d \in D, \forall i \in N^d, \tag{2.10}$$

$$y_{ij}^d \in \{0,1\} \quad \forall d \in D, \forall (i,j) \in A^{d*}. \tag{2.11}$$

The objective function of this formulation is trivial. Recall, however, that we can ignore the long arcs because of the assumption that there are no costs involved when a vehicle is idle at the depot. Constraints (2.6) and (2.7) assure that each trip is assigned to exactly one predecessor and one successor if this trip is assigned to depot d. Furthermore, constraints (2.8) assure that every trip is assigned to exactly one depot. Constraints (2.9) state that the number of vehicles used from depot d should be at least the number of vehicles needed for trips and deadheads at any time point. It suffices to consider only time points in H^d, since only at these time points the number of vehicles in use can increase, i.e. a departure from the depot may occur. Moreover, if there are two consecutive time points in H^d between which no arrival at the depot can occur, then the number of vehicles at the latest time point is at least the number of vehicles at the earlier one. This means that the constraint for the earlier time point can be left out. Finally, the binary conditions on the x and y variables combined with constraints (2.9) guarantee that the variables B^d are positive integers.

2.3 Crew Scheduling

Crew scheduling is one of the most important planning problems of a public transport company, since the crew costs are mostly dominant. The problem deals with assigning tasks to duties such that each task is performed, each duty is feasible w.r.t. a set of working rules and the total costs of the duties are minimized. In the basic version of the problem, which is considered in this thesis, there are only crew rules that are defined on an individual duty. That is, rules which require that a (maximum/minimum) number (percentage) of the duties has certain properties are not taken into account. Recall from Chapter 1 that a task is defined as the sequence of deadheads and trips between two relief

points on a vehicle block. Examples of working rules are a maximum spread (length) of the duty, a minimum length of a break in the duty and a maximum working time in the duty which is the length of the duty minus the length of the break. Of course, in practical applications there can be many more of such rules. However, in practice, the objective is most often very simple, namely minimizing the total number of duties. Therefore, only fixed crew costs are necessary to take into account. This is also the cost function, which we will consider during the computational tests in this thesis.

2.3.1 Literature Review

The literature on crew scheduling is very rich and can be divided into two main classes, namely theoretical and application-based papers. The first class of theoretical papers deals with formulations and algorithms for simplified crew scheduling problems, e.g. with only a few constraints w.r.t. the feasibility of a duty. In the second class those algorithms are applied to solve problems which arise from practical applications.

Before we will discuss the first class of papers, we have to remark that the *crew scheduling problem* (CSP) is NP-hard even in the case that there are only spread time or working time constraints. That is, the only requirement w.r.t. the feasibility of a duty is its length or the working time in that duty, respectively. For the proofs we refer to Fischetti *et al.* (1987) and Fischetti *et al.* (1989). In most papers the CSP is formulated as a set partitioning or covering problem and solved with a column generation approach. Therefore, the working rules in a duty have to been taken into account only in the pricing problem. The master problem can be solved by LP-relaxation or by Lagrangian relaxation. In most papers, e.g. Desrochers & Soumis (1989), Falkner & Ryan (1992) and Desrochers *et al.* (1992a), the first one is used. However, there are also some papers where Lagrangian relaxation is used. These are, among others, Carraresi *et al.* (1995) and Freling (1997). We refer to Chapter 1 for a more detailed discussion about column generation and the benefits of combining it with Lagrangian relaxation. Alternatively to column generation, the set of columns can be enumerated or generated heuristically. To get integer solutions different algorithms, exact or heuristical, can be used. Hoffman & Padberg (1993) describe a branch-and-cut algorithm where cutting planes are generated based on the underlying structure of the polytope. In Caprara *et al.* (1999a) Lagrangian heuristics and variable fixing techniques are used to find integer solutions. Recently, Mingozzi *et al.* (1999) suggest an approach to compute a dual solution of the LP-relaxation by combining a number of different bounding procedures. Afterwards, the dual solution is used to reduce the number of primal variables such that the resulting problem can be solved by a branch-and-bound algorithm. Furthermore, the different procedures use lower bounds of previous procedures and improve them. In some of these procedures Lagrangian relaxation and column generation are used.

In an exact algorithm for the overall problem, a column generation algorithm should be embedded in the search for integer solution. This is most often done by a branch-and-price algorithm, where not only in the root node but in each node of the branch-and-bound tree column generation is applied (see e.g. Barnhart *et al.* (1998)).

A completely different formulation is used by Fischetti *et al.* (2001), namely a single-commodity flow formulation. The disadvantage of such a formulation is the poor LP-relaxation, however, they solve this weakness by adding valid inequalities to strengthen the formulation. To solve the problem these inequalities are embedded in a branch-and-cut algorithm.

The second class of papers are based on applications. These applications are not only from bus transport, but also in the field of railways (see e.g. Caprara *et al.* (1999b), Freling *et al.* (2001c) and Kroon & Fischetti (2001)) and airlines (see e.g. Desaulniers *et al.* (1997) and Desaulniers *et al.* (1999)).

Recent developments in crew scheduling focus on integrating with vehicle scheduling (the topics of Chapters 3 and 4) and on taking disturbances into account (see Chapter 5). The literature review on these issues is postponed to the corresponding chapters.

Nowadays, a lot of public transport companies use commercial software packages. The most used package worldwide is the HASTUS system, which already exists for more than 20 years (see Rousseau & Blais (1985)). Other packages are, among others, GIST (see Dias *et al.* (2001)), TURNI (see Kroon & Fischetti (2001)), TRACS II (see Fores *et al.* (2001)) and DOPT (see Section 2.4).

2.3.2 Mathematical Formulation

In this subsection we give a mathematical formulation for the crew scheduling problem. Let f_k be the cost of duty $k \in K$, where K is the set of all feasible duties, and define $K(i) \in K$ as the set of duties covering task $i \in I$. Consider binary decision variable x_k indicating whether duty k is selected in the solution or not. In the *set covering formulation* of the CSP, the objective is to select a minimum cost set of feasible duties such that each task is included in at least one of these duties, which leads to the following 0-1 linear program.

(CSP):

$$\min \quad \sum_{k \in K} f_k x_k \tag{2.12}$$

$$\text{s.t.} \quad \sum_{k \in K(i)} x_k \geq 1 \qquad \forall i \in I, \tag{2.13}$$

$$x_k \in \{0,1\} \quad \forall k \in K. \tag{2.14}$$

The advantage of working with this formulation instead of a *set partitioning* one is that it is easier to solve (a similar remark can be made for relaxations of the respective

problems). After solving the set covering formulation, we can always change the solution into a set partitioning solution by deleting over-covers of tasks. In fact, this boils down to changing some of the selected duties. For instance, instead of being the driver, the person who is assigned to such a duty will make a trip as a passenger. Note that such changes affect neither the feasibility nor the cost of the duties involved.

2.3.3 Algorithm

We solve the CSP in two steps. First we have to solve the vehicle scheduling problem (either the SDVSP or the MDVSP dependent of the number of depots), generate all feasible pieces and (optional) all feasible duties first. Then we select the optimal duties by solving the set covering model of the previous subsection with a Lagrangian heuristic.

In the remainder of this subsection we will assume that there is only one depot. However, the algorithm can be straightforwardly generalized to the multiple-depot case. The solution of the SDVSP gives a set of vehicle blocks on which we can define the relief points. Recall from Chapter 1 that a relief point is the point on a vehicle block, where a driver can have his break or can be relieved by another one. This results in a set of tasks from which we can easily enumerate all feasible pieces, since a piece is a feasible sequence of consecutive tasks on the same vehicle block only restricted by its duration. Next, we generate all feasible duties. Since duties often consist, in practice, of at most 3 pieces, we can do this by enumerating all possible combinations of pieces and check if such a combination is feasible. One may use a duty generation network, as proposed by Freling (1997), to generate duties consisting of more than 3 pieces. Finally, we select the duties with the Lagrangian heuristic, for which a global description is given in Figure 2.2.

Step 0: Initialization
Generate a set of pieces such that each task can be covered by at least one piece.
The initial set of columns consists of these pieces.
Step 1: Computation of dual multipliers
Solve a Lagrangian dual problem with the current set of columns.
This yields a lower bound for the current set of columns.
Step 2: Selection of additional columns
Select columns from the previously generated duties with negative reduced cost.
If no such columns exist, which means that the lower bound computed in Step 1 is a lower bound for the overall problem, (or another termination criterion is satisfied), go to Step 3; otherwise, return to Step 1.
Step 3: Construction of feasible solution
Use all the columns selected in Step 0 and Step 2 to construct a feasible solution.

Figure 2.2: Solution method for CSP

Suppose that, at some point, K is the set of columns (duties) that we are considering in the master problem. We compute the lower bound with respect to these columns as follows. We associate non-negative Lagrangian multipliers $\lambda_1, \lambda_2, ..., \lambda_{|I|}$ with the constraints (2.13) and then the remaining Lagrangian subproblem amounts to *pricing out* the x variables. We use subgradient optimization to solve the Lagrangian dual problem approximately. Besides a good lower bound, we also obtain a final set of Lagrangian multipliers. These are used to compute the reduced cost $\overline{f}_k$ of columns $k \notin K$ in the column generation step, where $\overline{f}_k = f_k - \sum_{i \in I(k)} \lambda_i$ with respect to the Lagrangian multipliers $\lambda = (\lambda_1, \lambda_2, ..., \lambda_{|I|})$, and $I(k)$ is the set of tasks in duty k. We add a number of duties with negative reduced cost to K and repeat the subgradient optimization. Actually, to assure that we do not generate columns twice, we modify the Lagrangian multipliers before we select new columns. The modifications are such that all columns already in K get non-negative reduced cost, while not decreasing the value of the Lagrangian function. For details on this procedure, we refer to Subsection 1.4.3.

We stop if there are no duties left with negative reduced cost or if the lower bound obtained by the subgradient optimization does not significantly change for a number of iterations. In the first case we obtain a true lower bound (although not necessarily the best Lagrangian lower bound); in the second case we do not have this guarantee.

Finally we compute a feasible solution by solving a set covering problem in which we consider all the columns which have been selected along the way. We can either do this exactly, using a general or specialized integer programming solver, or heuristically. Most often we use the set covering heuristic of Caprara *et al.* (1999a) with a few changes proposed in Freling (1997). In the remainder of this subsection, we briefly discuss this heuristic (called CFT). For the details we refer to the original paper of Caprara *et al.* (1999a). The CFT heuristic consists of two main loops, one outer loop and one inner loop. In the inner loop a procedure consisting of 3 phases is applied (called 3-PHASE). In the first phase a set of near-optimal Lagrangian multipliers is calculated using subgradient optimization. The second phase starts from these multipliers and generates a sequence of multipliers, for each of which a heuristic solution is calculated. The best heuristic solution is used in the next phase, where a subset of the variables is fixed to 1. If there is an improvement in the solution the 3 phases are repeated, otherwise the inner loop is terminated. To reduce the computation time of the 3-PHASE procedure only a small subset of columns is considered. Therefore, a column generation algorithm is used in the outer loop. Furthermore, there is also a refining procedure in the outer loop where the solution of the 3-PHASE procedure is improved. This is done by fixing variables to 1 corresponding to some "good" columns and by solving the remaining subproblem.

2.4 Application: SBM

We conclude this chapter with a discussion on an application of crew scheduling at the local public transport company in Maastricht, Stadsbus Maastricht (SBM). Since March 2001, SBM has used the DOPT ("Duty Optimization for Public Transport") system to solve their crew scheduling problems. This system uses the algorithm presented in Subsection 2.3.3.

The bus network in Maastricht has a quite simple structure. All different lines go from a district in the western part of the city, via the city centre, across the river Maas, and via the central station to a district in the eastern part of the city and vice versa. In Figure 2.3, this structure is illustrated with an example of five lines (A-B,C-D,E-F,G-H,I-J).

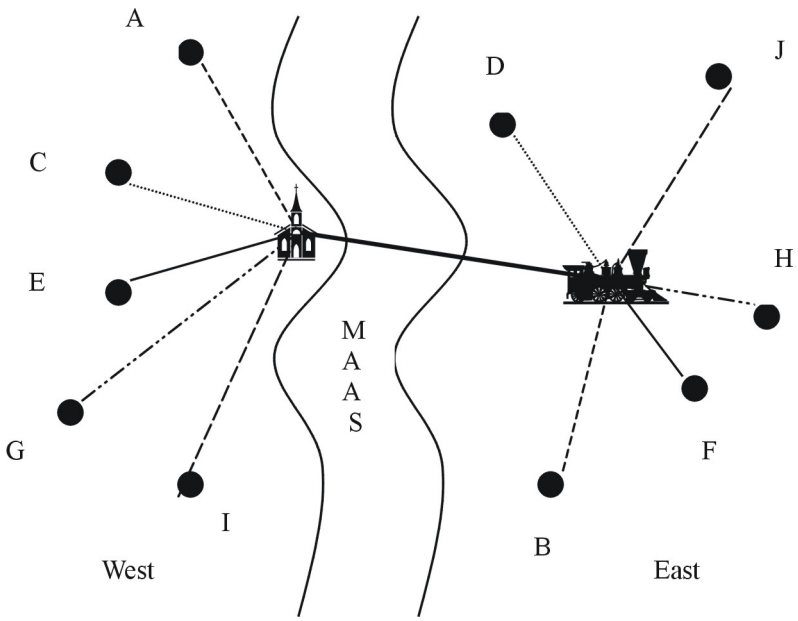

Figure 2.3: Structure of the bus network in Maastricht

Furthermore, there is only one central depot where all buses are parked during the night. Therefore, the vehicle schedules are easy to construct and have the following structure: start in the depot with a deadhead trip to a start location of one of the lines (e.g. A), then travel the whole day on that line (from A to B vv.) and after the last trip return to the depot.

The duties are divided into two main types: full-time and part-time duties. The first one consists of two pieces and one break in between at the central station, which is the only relief location. Furthermore, the tasks in the second piece should be from a vehicle on a different line than the tasks in the first piece. This is to prevent that drivers always drive the same route and thus have some alternation in their work. The other rules with respect to the feasibility of a duty are just the standard ones like a minimum length of the

break, maximum working time, maximum length of the duty and maximum length of a piece. Finally, SBM prefers to have a certain maximum number of duties that have a duty length between certain levels (e.g. between 10 and 11 hours). These global constraints can be handled by introducing different fixed costs for such duties.

Problem Instances

We consider here four problem instances corresponding to a weekday, Saturday and Sunday in the winter-timetable 2000/2001 (named we-win, Sa-win and Su-win, respectively) and a weekday in the summer-timetable of 2001 (named we-sum). The main characteristics of these instances are summarized in Table 2.1.

instance	we-win	Sa-win	Su-win	we-sum
line-groups	8	0	0	8
vehicles	41	29	15	31
tasks	961	617	420	755

Table 2.1: Instances at SBM

In Table 2.1, we denote the number of line-groups, vehicles and tasks. A line-group consists of one or more bus lines that have a similar route. Recall that SBM prefers that drivers drive two different routes in a duty. Therefore, it is not allowed that a duty consists of two pieces with tasks in the same line-group. At SBM, 10 different types of duties are considered, where 9 of them consist of 2 pieces and 1 are the so-called trippers, which are 1-piece duties only restricted by a minimum and maximum working time. The objective is to minimize the total costs of the duties. In the final part of the algorithm, where we construct feasible solutions, we also take the costs of over-covers, which are tasks that are in at least two duties, into account. This is implemented by adding a fixed, relatively small, number for each task in a duty to the costs of each duty.

Results

In Table 2.2, we present the total number of pieces and duties which are generated initially, the upper and lower bound on the optimal solution, the relative gap between these two bounds, the cpu times for the different steps (see Figure 2.2), the total cpu time and the number of duties and over-covers in the best solution found by the algorithm. Notice that the computation times are in seconds. The runs are executed on a Pentium III pc (450 MHz, 128 MB RAM).

We can conclude from Table 2.2 that the best solution found for all instances is less than 6% away from the optimal solution. However, we have the feeling that these gaps are mostly induced by the "weakness" of the lower bound, since we do not take into account the costs of over-covers in the lower bound. Furthermore, we can see that the computation

instance	we-win	Sa-win	Su-win	we-sum
# pieces	5,433	3,278	2,514	4,098
# duties	1,347,159	241,354	166,191	780,924
upper	128,100	83,750	56,300	103,150
lower	125,250	81,150	53,100	101,000
gap (%)	2.2	3.1	5.7	2.2
cpu_step0	620	143	86	321
cpu_step1	25	11	6	17
cpu_step2	21	5	3	14
cpu_step3	328	116	111	248
cpu_total	1,059	297	217	660
duties	78	55	36	65
over-covers	8	12	16	14

Table 2.2: Results of DOPT at SBM

times are quite reasonable. The largest instance could be solved in less than 20 minutes, while it took a planner a few weeks to come up with an initial crew schedule. However, it is obvious that the result of the algorithm cannot be directly implemented in practice. For several reasons, the planners still have to make some modifications. It is for example possible that some "feasible" duties are very unpopular with the drivers. Furthermore, the planners need to remove the over-covers, which is "in theory" an easy job, since one or more tasks removing from a duty has, in general, no effect on the feasibility of the duty. However, it can lead to very large breaks or very short duties, which are often not preferred by the drivers. Therefore, the experience and knowledge of the planners is used here to choose which duty is changed, or if necessary, which tasks are interchanged between several duties. Finally, the planners can sometimes improve the solution due to two reasons. The first one is obvious and has to deal with the fact that the algorithm is a heuristic. If there is a gap between the lower and upper bound, the solution can sometimes be improved. The second one is less obvious and deals with the fact that sometimes infeasible duties (e.g. a break of 29 instead of 30 minutes) are allowed if a significant saving can be obtained or other duties can be made more attractive for the drivers. This is mostly a process of negotiations by the company and the drivers or their representatives.

Chapter 3

Integrated Vehicle and Crew Scheduling (Single-Depot)

Although in the early eighties several researchers recognized the need to integrate vehicle and crew scheduling for an urban mass transit system, most of the algorithms published in the literature still follow the sequential approach where vehicles are scheduled before, and independently of, crews. Algorithms incorporated in commercially successful computer packages use this sequential approach as well, while sometimes integration is dealt with at the user level (see e.g. Darby-Dowman *et al.* (1988)). In the operations research literature, only a few publications address a simultaneous approach to vehicle and crew scheduling. None of those publications makes a comparison between simultaneous and sequential scheduling. Hence, they do not provide any indication of the benefit of a simultaneous approach. Only in the airline literature, there recently appeared a publication about the benefits of an integrated approach between aircraft routing (which is equivalent to vehicle scheduling) and crew scheduling. Klabjan *et al.* (2002) show that partial integration of schedule planning, aircraft routing and airline crew scheduling can improve the quality of the crew scheduling solution enormously. To be more specific, the flight time credit (FTC), which is defined as the percentage of excess cost above flying, is improved by a factor of two. Of course, this does not imply that such an improvement can also be obtained for bus transport, but it shows the necessity of making such a comparison.

In this chapter which is partly based on Freling *et al.* (2003), we discuss a complete integration of vehicle and crew scheduling for the single-depot case. The multiple-depot case is postponed to the next chapter. An overview of the potential benefits of integration is provided in Freling *et al.* (1999). To evaluate the effectiveness of this approach, we also consider the traditional sequential approach and the opposite sequential approach of scheduling crews before, and independently of, vehicles.

This chapter is organized as follows. Section 3.1 deals with problem definitions. An overview of the literature on simultaneous vehicle and crew scheduling approaches is provided in Section 3.2. All algorithms proposed in this section rely on column generation

applied to set partitioning type of models. We use Lagrangian relaxations and Lagrangian heuristics with column generation (see Chapter 1) and we discuss quality guarantees for both the integration and independent crew scheduling. The column generation (pricing) problem allows us to exploit the network flow structure of the crew scheduling problem, and is decomposed in two or three phases, each consisting of (constrained) shortest path type of problems. In Sections 3.3-3.5, we propose mathematical formulations and algorithms for complete integration of vehicle and crew scheduling, where a crew is allowed to work on more than one vehicle during a duty, and for the situation where this is not allowed, and for independent crew scheduling, respectively. Furthermore, we provide a computational study using real life data from the RET, the mass transit company in Rotterdam (Section 3.6). However, we do not consider all constraints of the RET in this study, since some constraints are very particular for the RET and we like to show that our approach can be applied in a general setting. Therefore, we discuss the actual problem of the RET in a separate section (Section 3.7, which is based on Freling *et al.* (2001a)), where the changes to the general approach are discussed. Furthermore, we show that an integrated approach can be applied to solve a real practical problem. We conclude this chapter with a short summary (Section 3.8).

3.1 Problem Definition

The *vehicle and crew scheduling problem* (VCSP) is the following: given a set of service requirements or *trips* within a fixed planning horizon, find a minimum cost schedule for the vehicles and the crews, such that both the vehicle and the crew schedules are feasible and mutually compatible. Each trip has fixed starting and ending times, and the travelling times between all pairs of locations are known.

A vehicle schedule is feasible if (1) each trip is assigned to a vehicle, and (2) each vehicle performs a feasible sequence of trips, where a sequence of trips is feasible if it is feasible for a vehicle to execute each pair of consecutive trips in the sequence. From a vehicle schedule it follows which trips have to be performed by the same vehicle and this defines so-called vehicle *blocks*. The blocks are subdivided at *relief points*, defined by location and time, where and when a change of driver may occur. A *task* is defined by two consecutive relief points and represents the minimum portion of work that can be assigned to a crew. These tasks have to be assigned to crew members. The tasks that are assigned to the same crew member define a crew *duty*. Together the duties constitute a crew schedule. Like in Chapter 2 we only consider the basic crew scheduling problem, where a schedule is feasible if (1) each task is assigned to one duty, and (2) each duty is a sequence of tasks that can be performed by a single crew, both from a physical and a legal point of view. In particular, each duty must satisfy several complicating constraints corresponding to work load regulations for crews. Typical examples of such constraints

are maximum working time without a break, minimum break duration, maximum total working time, and maximum duration. The cost of a duty is usually a combination of fixed costs such as wages, and variable costs such as overtime payment. Finally, we define a *piece (of work)* as a sequence of tasks on one vehicle block without a break that can be performed by a single crew member without interruption.

We make the following assumptions:

1. There is only one depot, all vehicles are available at all time and they are identical. Recall from Section 2.1 that with these characteristics the underlying vehicle scheduling problem is polynomially solvable.

2. The cost function for the VCSP is the summation of the vehicle and crew scheduling cost functions, where the primary vehicle scheduling objective is to minimize the number of vehicles, while the primary crew scheduling objective is to minimize the number of crews. An additional term can be added to the cost function which corresponds to variable vehicle costs (e.g. distance travelled) and variable crew costs (e.g. overtime payment).

3. The feasibility of a piece only depends on its duration, which is limited by a minimum and maximum piece length. This follows from legal regulations.

These three assumptions hold in a lot of real-world applications arising from public transport scheduling and from bus and driver scheduling, in particular.

We distinguish between two types of tasks, viz. *trip tasks* corresponding to (parts of) trips, and *dh-tasks* corresponding to deadheading. A *deadhead* is a period that a vehicle is moving to or from the depot, or a period between two trips that a vehicle is outside of the depot (possibly moving without passengers). All trip tasks need to be covered by a crew, while the covering of dh-tasks depends on the vehicle schedules and determines the compatibility between vehicle and crew schedules. In particular, each dh-task needs to be assigned to a crew if and only if its corresponding deadhead is assigned to a vehicle. Note that more than one trip task may correspond to a single trip, depending on the relief points along that trip. Similarly, more than one dh-task may correspond to a single deadhead. For example, a deadhead between the end location of trip i and the starting location of trip j which passes by the depot corresponds to two dh-tasks, one from the end location of trip i to the depot and the other from the depot to the starting location of trip j. Here we assume that waiting at the depot is not a task because vehicle attendance at the depot is not necessary.

3.2 Literature Review

The traditional sequential strategy is strongly criticized by Bodin *et al.* (1983). This is motivated by the fact that in North American mass transit organizations the crew costs dominate the vehicle operating costs, and in some cases reach as high as 80% of total operating costs. Although simultaneous vehicle and crew scheduling is of significant practical interest, only a few approaches of this kind have been proposed in the literature. They mainly deal with bus and driver scheduling and fall into one of the following three categories:

1. scheduling of vehicles during a heuristic approach to crew scheduling;
2. inclusion of crew considerations in the vehicle scheduling process; the actual crew scheduling is only carried out afterwards;
3. complete integration of vehicle and crew scheduling.

Most of the approaches are of the first category and are based on a heuristic procedure proposed by Ball *et al.* (1983). This procedure involves the definition of a scheduling network, which consists of vertices characterized by parts of trips called *d-trips* that have to be executed by one vehicle and crew, and two vertices r and t representing the depot. The arcs can be grouped into two categories, those which indicate that a crew and vehicle proceed from one d-trip to another and those which indicate that only the crew proceeds from one d-trip to another (*crew-only* arcs). The solution procedure is decomposed into three components, emphasizing the crew scheduling problem: a piece construction component, a piece improvement component and a duty generation component. The piece construction routine generates a set of pieces whose time duration is less than some constant T. (Note that this corresponds to vehicle scheduling with time constraints.) This problem is solved by a heuristic based on matching. In the second step pairs of short pieces are combined into partial duties, while in a third step pairs of these duties and longer pieces are combined into 2 and 3-piece duties. The second step uses a matching based interchange heuristic and the third one can be solved as a matching problem. Vehicle schedules are generated simultaneously by deleting the crew-only arcs and fixing arcs used by pieces in the solution. This procedure is applied to large size VCSP instances which correspond to the entire physical network, i.e. all lines are considered at once, while no restrictions are placed on interlining, i.e. a crew may work on an arbitrary number of lines.

Other heuristic approaches of the first category are proposed by Tosini & Vercellis (1988), Falkner & Ryan (1992) and Patrikalakis & Xerocostas (1992). All these approaches use a similar crew scheduling network as in Ball *et al.* (1983).

Approaches of the second category are proposed by Darby-Dowman *et al.* (1988) as an interactive part of a decision support system, and by Scott (1985) who heuristically

determines vehicle schedules which take crew costs into account. An initial vehicle schedule is heuristically modified according to estimated marginal costs associated with a small change in the current vehicle schedule. The estimated marginal costs are obtained by solving the linear programming dual of the HASTUS crew scheduling model (see Rousseau & Blais (1985)). Results obtained with public transport scheduling problems in Montréal, show a slight decrease in estimated crew costs.

Only very recently, approaches of the third category, that is, a complete integration of vehicle and crew scheduling, have been proposed. The first mathematical formulation (for the single-depot case) is proposed by Patrikalakis & Xerocostas (1992), which is slightly changed by Freling *et al.* (1995). This formulation (see also Freling *et al.* (1999)) is similar to the one considered in this chapter for the case where a crew is allowed to work on more than one vehicle during a duty.

Haase & Friberg (1999) propose an exact algorithm for the single-depot vehicle and crew scheduling problem. Both the vehicle and crew scheduling aspects are modelled by using set partitioning type of constraints. A *branch-and-cut-and-price* algorithm is proposed, that is, column generation and cut generation are combined in a branch-and-bound algorithm. Computational results indicate that only small problems (up to 20 trips) could be solved.

Haase *et al.* (2001) propose an approach which solves a crew scheduling problem that incorporates side constraints for the vehicles. This is done in such a way that the solution of this problem guarantees that an overall optimal solution is found after constructing a compatible vehicle schedule. The solution approach is based on a multi-commodity network flow formulation for the crew scheduling problem with side constraints, which is solved by a branch-and-price algorithm. Computational experiments with random data instances, simulating an urban mass transit environment, show that instances with up to 150 trips can be solved in 82 minutes of cpu time (on average on a SUN ULTRA-10/440 workstation) and with an optimality gap of 0.3% on average and always less than 1.2%. Furthermore, out of these 10 instances, 6 instances could be solved to optimality within 3 hours of cpu time. For larger problem instances a heuristic version of the algorithm is used. With this approach instances up to 350 trips can be solved within 2 hours of cpu time on average. The average (maximum) integrality gap for these instances is 0.3% (1.5%). We note, however, that their computational experiments cannot be compared in a straightforward way with ours, since there are some important differences between their random data and our real life data (see Section 3.6). Currently, they work on an extension of their approach to the multiple-depot case (see Desaulniers *et al.* (2001)). In the airline world, a similar approach is used to integrate aircraft routing and crew scheduling (see Cordeau *et al.* (2001) and Klabjan *et al.* (2002)). Both incorporate plane count constraints in their crew scheduling problem, which is essentially the same as the side constraints in Haase *et al.* (2001). They use a LP based branch-and-bound

methodology.

To the best of our knowledge, only one paper written by Gaffi & Nonato (1999), deals with integration in the multiple-depot case. Their approach is like ours based on Lagrangian relaxation with column generation. Their mathematical formulation is similar to one of the formulations presented in this paper and their approach is developed for the extra–urban mass transit setting, where crews are tightly dependent on the vehicle activities or dead-heading of crew is highly constrained. Since they consider a particular application, they make some assumptions which are not valid in general. These assumptions are that a driver is assigned to the same vehicle during the whole duty, and that all pieces of work (part of the duty between two breaks) start and end at a depot. Therefore, pieces of work and vehicle blocks coincide, which makes the problem computationally much more attractive than without these assumptions. The authors provide some computational results for Italian public transit operators, which show some improvements over the results of a sequential approach. Cpu times (on a Power PC 604, 180 Mhz) are over 24 hours for cases with up to 257 trips, and on average 2 to 6 hours. However, they do not compute lower bounds, which makes it difficult to give insight in the quality of their approach.

3.3 Complete Integration: General Case

In this section, we propose a mathematical formulation for the VCSP in the general case. We deal separately with the situation where changeovers are not allowed (see Section 3.4). A *changeover* is the change of vehicle of a bus driver during his break.

3.3.1 Mathematical Formulation

Patrikalakis & Xerocostas (1992) presented the first mathematical formulation for the VCSP. This is a simplified version of the model presented here. Our formulation is valid under the assumptions that short arcs correspond to only one dh-task and *continuous attendance* is required, i.e. there is always a driver present if the bus is outside the depot. If these assumptions are not met, we can use a slightly different formulation; for details we refer to Subsection 3.3.2.

The mathematical formulation we propose for the VCSP is a combination of the quasi-assignment formulation for the vehicle scheduling problem discussed in Subsection 2.1.1, and the set partitioning formulation for crew scheduling discussed in Subsection 2.3.2. The quasi-assignment part assures the feasibility of vehicle schedules, while the set partitioning part assures that each trip task is assigned to a duty and each dh-task is assigned to a duty if and only if its corresponding deadhead is part of the vehicle schedule. Before providing the mathematical formulation, we need to recall and to introduce some notation.

Let $N = \{1, 2, ..., n\}$ be the set of trips, numbered according to increasing starting time, and let $E = \{(i, j) \mid i < j,\ i, j \text{ compatible},\ i \in N, j \in N\}$ be the set of deadheads. Two trips are compatible if they can be driven by the same vehicle after each other. Let r and t both represent the depot. Like in Subsection 2.1.1, we define the vehicle scheduling network $G = (V, A)$ with nodes $V = N \cup \{r, t\}$, and arcs $A = E \cup (r \times N) \cup (N \times t)$. Let c_{ij} be the vehicle cost of arc $(i, j) \in A$, which can be defined in the same way as in Subsection 2.1.1.

To reduce the number of constraints, we assume again that a vehicle returns to the depot if it has an idle time between two consecutive trips which is long enough to let it return. Recall that such an arc is called a long arc, while the other arcs are called short arcs. Denote $A^s \subset A$ and $A^l \subset A$ as the sets of short and long arcs, respectively.

Furthermore, recall from Chapter 2 that K denotes the set of all feasible duties and f_k denotes the crew cost of duty $k \in K$, respectively. We assume that deadheads to and from the depot correspond to one dh-task each. Suppose $(i, j) \in A^l$ and let el_i and bl_j denote the ending and starting location of trips i and j, respectively. Then we let $K(i, t)$ and $K(r, j)$ denote the set of duties covering the dh-task from el_i to the depot and from the depot to bl_j, respectively. Furthermore, I_1 denotes the set of trip tasks and $K(p)$ is the set of duties covering trip task $p \in I_1$, where we assume that a trip corresponds to one task. Note that this means that there is a one-to-one correspondence between trips and trip tasks. $K(i, j)$ denotes the set of duties covering dh-tasks corresponding to deadhead $(i, j) \in A^s$. Decision variables x_k indicate whether duty k is selected in the solution or not. Furthermore, there are decision variables y_{ij} that indicates whether a vehicle covers trip j immediately after trip i or not. The VCSP can be formulated as follows.

(VCSP1):

$$\min \quad \sum_{(i,j)\in A} c_{ij} y_{ij} + \sum_{k \in K} f_k x_k \tag{3.1}$$

$$\text{s.t.} \quad \sum_{\{j:(i,j)\in A\}} y_{ij} = 1 \qquad \forall i \in N, \tag{3.2}$$

$$\sum_{\{i:(i,j)\in A\}} y_{ij} = 1 \qquad \forall j \in N, \tag{3.3}$$

$$\sum_{k \in K(p)} x_k = 1 \qquad \forall p \in I_1, \tag{3.4}$$

$$\sum_{k \in K(i,j)} x_k - y_{ij} = 0 \qquad \forall (i, j) \in A^s, \tag{3.5}$$

$$\sum_{k \in K(i,t)} x_k - y_{it} - \sum_{\{j:(i,j)\in A^l\}} y_{ij} = 0 \qquad \forall i \in N, \tag{3.6}$$

$$\sum_{k \in K(r,j)} x_k - y_{rj} - \sum_{\{i:(i,j)\in A^l\}} y_{ij} = 0 \qquad \forall j \in N, \tag{3.7}$$

$$x_k, y_{ij} \in \{0, 1\} \quad \forall k \in K, \forall (i, j) \in A. \tag{3.8}$$

The objective is to minimize the sum of total vehicle and crew costs. The first two sets of constraints, (3.2) and (3.3), correspond to the quasi-assignment formulation for the SDVSP. They assure that each trip is assigned to exactly one predecessor and successor, that is, these constraints guarantee that the network is partitioned into a set of disjoint paths from r to t. Constraints (3.4) assure that each trip task p will be covered by one duty in the set $K(p)$. Furthermore, constraints (3.5), (3.6) and (3.7) guarantee the link between dh-tasks and deadheads in the solution, where deadheads corresponding to short and long arcs in A are considered separately. In particular, constraints (3.5) guarantee that each deadhead from i to j is covered by a duty in the set $K(i,j)$ if and only if the corresponding short arc is in the vehicle solution. However, notice that the constraints (3.5) are correct under the assumptions we made, but under different assumptions we may have to change the equality sign to an "$\geq$" sign (see the following subsection). The other two constraint sets, (3.6) and (3.7), ensure that the dh-tasks from el_i to t and from r to bl_j, possibly corresponding to long arc $(i,j) \in A$, are both covered by one duty if and only if the corresponding deadheads are in the solution. Note that the structure of these last three sets of constraints is such that each constraint corresponds to the possible selection of one duty from a large set of duties, where the fact whether or not a duty has to be selected depends on the values of the corresponding y variables. We will refer to these constraints as *partitioning type of constraints.* The model contains $|A| + |K|$ variables and $4\,|N| + |A^s| + |I_1|$ constraints, which is already a huge number of variables and a very large number of constraints for instances with a small number of trips. This is probably the main reason why complete integration has previously received very little attention.

3.3.2 A Note on Model VCSP1

In the previous subsection we proposed a mathematical formulation for the VCSP for the general case with two minor assumptions, namely that (1) there is continuous attendance (there is always a crew present if a vehicle is outside the depot) and (2) a short arc consists of only one dh-task. In this subsection we will look at the model for the general case without these assumptions.

If the assumptions (1) and (2) do not hold, we need to replace constraints (3.5) by the following set of constraints:

$$\sum_{k \in K(i,j)} x_k - y_{ij} \geq 0 \quad \forall (i,j) \in A^s. \tag{3.9}$$

We explain the correctness of this formulation for both cases with an example.

In the first case we do not assume continuous attendance. Suppose we have the following two duties:

duty 1: $r \rightarrow 1 \overset{i}{\rightarrow} 2 \rightarrow 3 \overset{j}{\rightarrow} 4 \rightarrow t$,

duty 2: $r \rightarrow 5 \overset{k}{\rightarrow} 6 \rightarrow 7 \overset{l}{\rightarrow} 8 \rightarrow t$,

where the odd numbers correspond to the start points of the four trips, called i, j, k and l and the even numbers correspond to the end points of the same trips. The break in the first duty is between 2 and 3 and in the second duty between 6 and 7. If duty 1 is selected, then it is allowed that there is a vehicle doing trip j or trip l immediately after trip i. So if $x_1 = 1$, then y_{ij} or y_{il} is 1. Constraints (3.2) and (3.3) guarantee that every trip has one successor and one predecessor, so we know that y_{ij} and y_{il} are not both equal to 1. This is the reason why equality in the constraints (3.5) is not valid and we need to have constraints (3.9). For this example, we then get the following constraints:

$$\begin{aligned} x_1 - y_{ij} &\geq 0, \\ x_1 - y_{il} &\geq 0, \\ x_2 - y_{kl} &\geq 0, \\ x_2 - y_{kj} &\geq 0. \end{aligned}$$

In the other case, a short arc consists of more than one dh-task. Suppose that we have one short arc that is corresponding to two dh-tasks. For example, we have the following five duties and again four trips:

duty 1: $r \rightarrow 1 \overset{i}{\rightarrow} 2 \rightarrow 7 \rightarrow 3 \overset{j}{\rightarrow} 4 \rightarrow t$,

duty 2: $r \rightarrow 7 \overset{l}{\rightarrow} 8 \rightarrow t$,

duty 3: $7 \overset{l}{\rightarrow} 8 \rightarrow t$,

duty 4: $r \rightarrow 5 \overset{k}{\rightarrow} 6 \rightarrow 3$,

duty 5: $r \rightarrow 5 \overset{k}{\rightarrow} 6 \rightarrow t$,

where the numbers are corresponding to the start and end points of the trips i, j, k and l as before. There is only a break in the first duty, namely between 7 and 3. The short arc (2,3) consists of two dh-tasks, namely (2,7) and (7,3). This means that if duty 1 is selected, a vehicle can do trip l (starts with 7) or trip j (starts with 3) directly after trip i (ends with 2). Therefore we cannot have equality in the constraint set (3.5), but we have the following constraints corresponding to set (3.9):

$$\begin{aligned} x_1 - y_{ij} &\geq 0, \\ x_1 - y_{il} &\geq 0, \\ x_4 - y_{kj} &\geq 0. \end{aligned}$$

There are two feasible solutions, namely duty 1, 2 and 5 with three vehicles, where every vehicle does exactly the same as the three duties, and duty 1, 3 and 4 with two vehicles, where there is a changeover in duty 1. It is obvious that above constraints give exactly these solutions.

Finally, notice that the model with constraints (3.9) is also a correct model if the assumptions (1) and (2) hold, since then the formulation with (3.5) is the same as with (3.9). This is the same, because constraints (3.4) assure that every trip task is in exactly

one duty. So in the case that both assumptions hold, we have that if a certain trip j follows directly after trip i in a selected duty, then no trip $k \neq j$ follows directly after trip i for another selected duty. Thus $y_{ik} = 0 \; \forall k \neq j$. Constraints (3.2) assure now that if $\sum_{k \in K(i,j)} x_k = 1$, then $y_{ij} = 1$. So the formulation with (3.5) is equivalent to the one with (3.9).

3.3.3 Algorithm

The algorithm for VCSP1 is shown in Figure 3.1.

Step 0: Initialization
Solve VSP and CSP (using the algorithm of Figure 2.2) and
take as initial set of columns the duties in the CSP-solution.
Step 1: Computation of dual multipliers
Solve a Lagrangian dual problem with the current set of columns.
This gives a lower bound for the current set of columns.
Step 2: Generation of columns
Generate columns (duties) with negative reduced cost.
Compute an estimate of a lower bound for the overall problem.
If the gap between this estimate and the lower bound found in Step 1 is small enough (or another termination criterion is satisfied), go to Step 3; otherwise, return to Step 1.
Step 3: Construction of feasible solution
Based on feasible vehicle solution(s) from Step 1, construct
corresponding feasible crew solution(s) using the algorithm of Figure 2.2

Figure 3.1: Solution method for VCSP1

We use the relaxation of model VCSP1, where all set partitioning type of constraints (3.4)-(3.7) are first replaced by set covering constraints, which are subsequently relaxed in a Lagrangian way. That is, we associate non-negative Lagrangian multipliers λ_p, μ_{ij}, ν_i and ξ_j with constraints (3.4), (3.5), (3.6) and (3.7), respectively. Then the remaining Lagrangian subproblem can be solved by pricing out the x variables and solving a SDVSP for the y variables. We can use the auction algorithm of Freling *et al.* (2001b), which is also discussed in Subsection 2.1.2, for solving the SDVSP. Notice that we get a feasible vehicle solution every time we solve the SDVSP.

Furthermore, we need an additional procedure to update the Lagrangian multipliers after solving the Lagrangian relaxation. This is necessary to assure that all duties in the current master problem have non-negative reduced cost so that these duties will not be generated again in the pricing problem. This procedure is explained in Subsection 1.4.3.

In contrast to our approach to the CSP, we cannot generate the complete set of duties before we run the actual optimization algorithm, because we do not have a vehicle solution in advance. Therefore, we have to generate the duties during the process. We do this by generating pieces of work, which we will describe in Subsection 3.3.5 and then combining these pieces to duties. This is the topic of Subsection 3.3.6.

In every iteration i we compute an estimate LBT_i of the lower bound for the overall problem. Let LBS_i denote the value of the Lagrangian lower bound in iteration i, then the estimate is computed as

$$LBT_i = LBS_i + \sum_{k \in K_i} \overline{f}_k \tag{3.10}$$

where K_i is the set of duties added in iteration i and $\overline{f}_k$ is the reduced cost of duty $k \in K$, which is defined below.

$$\overline{f}_k = f_k - \sum_{p \in I_1(k)} \lambda_p - \sum_{(i,j) \in A^s(k)} \mu_{ij} - \sum_{i \in N^t(k)} \nu_i - \sum_{j \in N^r(k)} \xi_j, \tag{3.11}$$

where $I_1(k)$, $A^s(k)$, $N^t(k)$ and $N^r(k)$ denote the set of trip tasks I_1 in duty k, the set of short arcs A^s in duty k, the set of trips that have a corresponding task in duty k with the end of this trip as starting location and the depot as ending location and the set of trips that have a corresponding task in duty k with the depot as starting location and the start of this trip as ending location, respectively.

LBT_i is only a true lower bound for the overall problem if all duties with negative reduced cost have been added to the master problem. Of course, we can stop if LBT_i is equal to LBS_i, but in practice we stop earlier, namely if the relative difference is small and LBT_i is a true lower bound, or if there has not been any significant improvement in LBS_i during a number of iterations.

At the end we compute a feasible crew schedule given the (feasible) vehicle schedule which resulted from solving the last Lagrangian subproblem. We do this by solving the CSP (using the algorithm described in Figure 2.2). Of course, it is also possible to compute more feasible solutions by solving the CSP not only for the vehicle solution from the last iteration, but also for vehicle solutions which were encountered earlier on. A reason to actually do this could be that the gap between the lower and upper bound is quite large, which is an indication that the upper bound could be improved upon.

3.3.4 The Column Generation Subproblem

For the VCSP, vehicle blocks are not known and a huge number of feasible pieces of work may exist. Although Ball *et al.* (1983) have proposed a duty generation network that does not consider pieces of work explicitly, this approach is not interesting for column generation purposes due to the size of the network. Therefore, we propose a two phase

procedure for the column generation pricing problem: in the first phase, a *piece generation network* is used to generate a set of pieces of work which serve as input for the second phase where duties are generated.

A disadvantage of the two-phase procedure is that it is possible to generate a duty twice in the second phase, which results in an unnecessary large amount of variables. For example, a duty containing the same tasks can be constructed from two different pieces of work. It is very time consuming to check for each new duty generated if it has been generated before. However, we have implemented an efficient procedure to check for double duties after a fixed number of iterations of our duty generation algorithm. The current list of generated duties is then updated so that it contains no double duties, and the algorithm for generating duties is continued.

3.3.5 Generation of Pieces of Work

Recall that we have defined a piece of work as a continuous sequence of trip tasks and dh-tasks corresponding to (a part of) one vehicle block, and that this sequence of tasks is only restricted by its duration.

Network Structure

The network for piece generation is an extension of the network G for vehicle scheduling (see Subsection 2.1.1). Let a *start point* (*end point*) be defined as the relief point corresponding to the start (end) of a vehicle trip. We define the network $G' = (N', A')$, where nodes correspond to the relief points on each trip, and the source r and the sink t represent the depot. Arcs in A' correspond to dh-tasks and trip tasks. Because every relief point has a unique time point and as we assume that the nodes are numbered according to increasing time, G' is acyclic.

Let b_{ij} be the cost associated with each arc $(i, j) \in A'$. These costs should be defined in such a way that the cost of each path is equal to the cost of the corresponding piece, which in turn is itself derived from the costs of the duties. Notice that this is not possible with each arbitrary cost function. Therefore, we implicitly assume that we have a cost function where this is possible, e.g. the case with only fixed crew costs. Then $b_{ij} = 0$ for each arc $(i, j) \in A'$. Recall from Subsection 3.3.3 that we associate Lagrangian multipliers λ_p, μ_{ij}, ν_i and ξ_j with constraints (3.4), (3.5), (3.6) and (3.7), respectively. Then, for model VCSP1, the reduced cost is defined as

$$\overline{b}_{ij} = \begin{cases} b_{ij} - \lambda_p, \text{ for each arc } (i,j) \text{ corresponding to a trip task } p, \\ b_{ij} - \mu_{ij}, \text{ for each arc } (i,j) \text{ corresponding to a short arc } (i,j), \\ b_{ij} - \xi_q, \text{ for each arc } (i,j) \text{ with } i = r \text{ and } j \text{ the start point of trip } q, \\ b_{ij} - \nu_q, \text{ for each arc } (i,j) \text{ with } j = t \text{ and } i \text{ the end point of trip } q. \end{cases}$$

Thus, the reduced costs on the arcs are defined such that the reduced cost of a path is equal to the reduced cost of the corresponding piece of work. Let tp_i be the point in time associated with node $i \in N' \setminus \{r, t\}$. Each path $P(u, v)$ between two nodes u and v in network G' corresponds to a feasible piece of work if its duration satisfies the time constraint

$$dur_{min} \leq tp_v - tp_u \leq dur_{max},$$

where dur_{min} and dur_{max} denote the minimum and maximum allowed duration of a piece of work, respectively. The duration of a piece of work starting at r and/or ending at t is determined by incorporating travel times. For example, a path $r, u, ..., v, t$ corresponds to a feasible piece of work if the inequalities

$$dur_{min} \leq tp_v - tp_u + trav(r, bl_u) + trav(el_v, t) \leq dur_{max}$$

hold, where $trav(d, bl_u)$ and $trav(el_v, d)$ denote the travel times (possibly including sign-on and sign-off times) from the depot to bl_u and from el_v to the depot, respectively.

An All-Pairs Shortest Path Algorithm

In this subsection, we propose a polynomial all-pairs shortest path algorithm for generating a set of pieces. Such a set is generated by solving a shortest path problem between each pair of nodes in network G' that satisfy the constraints on the piece duration. For all feasible paths $P(u, v) = u, ..., v$ between nodes $u \neq r$ and $v \neq t$ three additional paths are considered, namely path $r, u, ..., v$, path $u, ..., v, t$ and path $r, u, ..., v, t$. In some applications, all crew breaks need to be assigned to the depot and all crew starts and ends their duty at the depot. In this case, only paths of the form $r, u, ..., v, t$ are considered. The algorithm is shown in Figure 3.2.

```
for i ∈ N'\{r,t} do
begin
   Determine k ∈ arg max_{j∈N'\{r,t}} {tp_j − tp_i ≤ dur_max}
   SHORTESTPATHS(i, k)
   RETRIEVEPIECES(i, k)
end
```

Figure 3.2: Piece Generation Algorithm

The procedure SHORTESTPATHS(i, k) returns the shortest paths $P^*(i, j)$ from i to $j = i + 1, ..., k$ in $O(m)$ time because G' is acyclic, where m is the number of arcs in the network G'. The procedure RETRIEVEPIECES(i, k) is shown in Figure 3.3.

The term $r + P^*(i, j)$ denotes the extension of path $P^*(i, j)$ by adding r as the initial node, and the term $P^*(i, j) + t$ denotes the extension of path $P^*(i, j)$ by adding t as the

for $j = i + 1, ..., k$ do
begin
  if $tp_j - tp_i \geq dur_{min}$ then accept $P^*(i, j)$
  if $dur_{min} \leq tp_j - tp_i + trav(r, bl_i) \leq dur_{max}$ then accept $r + P^*(i, j)$
  if $dur_{min} \leq tp_j - tp_i + trav(el_j, t) \leq dur_{max}$ then accept $P^*(i, j) + t$
  if $dur_{min} \leq tp_j - tp_i + trav(r, bl_i) + trav(el_j, t) \leq dur_{max}$
    then accept $r + P^*(i, j) + t$
end

Figure 3.3: Procedure RETRIEVEPIECES(i, k)

terminal node. Accepted pieces are stored in a set of potential pieces to be used for the duty generation phase. The running time of procedure RETRIEVEPIECES(i, k) is $O(n)$, where n equals the number of trips. Thus, the overall running time of the piece generation algorithm is $O(nm)$.

When generating sets of pieces we have to assure that the column generation optimality condition is satisfied, that is, no negative reduced cost duties exist at convergence of the column generation procedure. To facilitate the discussion, we do not consider r and t as relief points. Hence, the first relief point of a path $r, i, ..., j$ is the point corresponding to node i. Let Q denote the set of pieces containing all pairs of shortest paths generated by the algorithm in Figure 3.2. Furthermore, we assume that the duty generation algorithm satisfies the column generation optimality condition, that is, negative reduced cost duties are detected as long as corresponding paths exist in the duty generation network. Recall that we have assumed that the feasibility of a piece of work depends on the starting and ending relief point only. We now have the following proposition.

Proposition 3.1 *If the duty generation algorithm with set Q as the set of pieces does not return a negative reduced cost duty under the assumptions stated above, then there are no duties with negative reduced costs at all.*

Proof. Consider an arbitrary feasible duty with pieces not in Q, and suppose that this duty has negative reduced cost. Any piece in this duty that is not in Q can be replaced by a piece in Q with the same starting and ending relief point and with lower or equal reduced cost, while the duty remains feasible due to the assumptions stated above. Thus, if at least one duty with negative reduced costs exists, then at least one such duty exists with pieces in Q only. Hence, if no negative reduced cost duties are found with pieces in Q only, then no negative reduced cost duties exist when considering all pieces. ■

3.3.6 Generation of Duties

Duties consist of a number of pieces with a given maximum number of pieces. In practice this maximum is very often equal to 2 or 3. This is the reason why we simply enumerate all possible combinations of pieces and check if such a combination is feasible, until we find a specified number of duties with negative reduced costs. (To generate duties consisting of more than 3 pieces, one may use a duty generation network, as proposed by Freling (1997).) The reduced cost of a duty can be easily computed if the reduced cost of a piece is already known: the reduced cost of a duty is equal to the sum of the reduced cost of the pieces it is built from, plus the reduced cost of the breaks between the pieces.

3.4 Complete Integration: No Changeovers

3.4.1 Model and Algorithm

The formulation presented in the preceding section is valid irrespective of the fact whether or not changeovers are allowed. It only differs in the set of feasible duties K. However, if changeovers are not allowed, we can also use a different formulation. Recall that a changeover is the change of vehicle of a driver during his break. We define a *combined duty* as a feasible vehicle duty *and* one or more corresponding feasible crew duties. The compatibility of vehicle and crew duties is guaranteed by the definition of a combined duty. Therefore, the VCSP without changeovers can be modelled in a straightforward way as a set covering problem. Before providing the formulation, we need to introduce some notation.

The cost of a combined duty k, denoted by g_k, is defined as the sum of the cost of the vehicle duty and the costs of the corresponding crew duties. Let N be the set of trips, K the set of all combined duties, and $K(i)$ the set of combined duties covering trip i. Furthermore, binary variable x_k indicates whether combined duty k is selected in the solution or not. The VCSP without changeovers can be formulated as follows.

(VCSP2):

$$\min \quad \sum_{k \in K} g_k x_k \tag{3.12}$$

$$\text{s.t.} \quad \sum_{k \in K(i)} x_k \geq 1 \qquad \forall i \in N, \tag{3.13}$$

$$x_k \in \{0,1\} \quad \forall k \in K. \tag{3.14}$$

Note that, for similar reasons as mentioned in Section 2.3, we use a set covering formulation, i.e. constraints (3.13) require that each trip must be covered by a combined duty, but possibly more than once.

The algorithm for VCSP2 is almost similar to the one given in Figure 3.1 for VCSP1. One difference is that we have to take into account that no changeovers are allowed in the initial solution. We do this by solving the CSP where the set of duties consists only of duties without changeovers. Since model VCSP2 is a pure set covering model, Lagrangian relaxations can be obtained in a similar way as discussed in Subsection 2.3.3. For the model VCSP2 we must generate combined duties instead of duties. How this can be done, will be discussed in detail in Subsection 3.4.2. The other difference with Figure 3.1 is that we obtain a feasible solution in a similar way as for CSP by solving a set covering problem with the columns generated during the lower bound phase. So the difference is that we do not compute a feasible vehicle schedule first, since combined duties already define a vehicle schedule in itself.

3.4.2 The Column Generation Subproblem

The column generation pricing problem for the integrated model without changeovers needs an additional procedure in order to generate combined duties using previously generated crew duties as input (see Subsection 3.3.4). This results in a three phase procedure. While the piece and duty generation networks are created only once, the combined duty generation network needs to be built every time combined duties are generated, using the duties generated previously.

Figure 3.4 demonstrates the procedures.

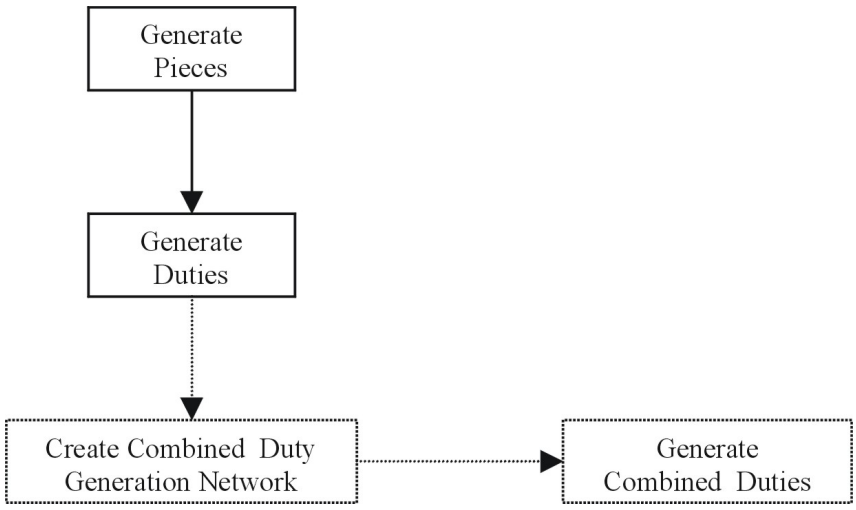

Figure 3.4: Column generation problem

A disadvantage of the three-phase procedure is that it is possible to generate a duty twice in the second phase, and to generate a combined duty twice in the third phase. This problem is already mentioned in Subsection 3.3.4.

We will discuss the differences in the generation of pieces compared to the integrated model with changeovers and describe the generation of combined duties.

Generation of Pieces of Work

We can use the same network for the generation of pieces as in Subsection 3.3.5, but the costs in this network are different. Let λ_p be the Lagrangian multiplier corresponding to a trip or trip task, and associated with the set covering constraints in model VCSP2. Then the reduced cost is defined as

$$\overline{b}_{ij} = \begin{cases} b_{ij} - \lambda_p, \text{ for each arc } (i,j) \text{ corresponding to a trip or trip task } p, \\ b_{ij}, \text{ otherwise.} \end{cases}$$

Of course, we can still use the same all-pairs shortest path algorithm to generate the set of pieces.

Generation of Combined Duties

An additional procedure is necessary to generate combined duties for model VCSP2. The input for this procedure is a set of duties generated by the two phase procedure, that is, pieces of work are generated as described in Subsection 3.3.5, while duties are generated as described in Subsection 3.3.6. The purpose of the third phase is to find combined duties with negative reduced cost. This is done by constructing a combined duty generation network based on the given duties, and solving a sequence of (unconstrained) shortest path problems in this network. The negative reduced cost combined duties generated in this way are added to the master problem, while convergence of the column generation algorithm is attained once no negative reduced cost combined duties are obtained.

Combined duty generation network G is constructed as follows. Nodes correspond to the set of duties previously generated plus a source and a sink. An arc from the source to a duty node exists if the starting relief point of the duty is located at the depot, while an arc from a duty to the sink exists if the ending relief point of the duty is located at the depot. Furthermore, an arc between two duty nodes i and j exists if duty i and j form a compatible pair of duties on one vehicle, that is, if $el_i = bl_j$ and $et_i \leq bt_j$. In addition, an arc exists if two duties can serve together on one vehicle while interrupting each other. Suppose for example that two duties i and j each contain two pieces p_i, q_i and p_j, q_j, respectively. Then, arc (i,j) exists if pairs $(p_i, p_j), (p_j, q_i)$ and (q_i, q_j) form compatible pairs of pieces.

Reduced vehicle trip costs and vehicle deadhead costs are incorporated in the reduced cost of pieces and duties. This is achieved by adjusting arc costs in the piece generation network. The costs of arc (i,j), $i \neq r$ in network G equals the reduced cost of duty j plus the vehicle cost of the corresponding deadhead. If $i = r$ then the fixed vehicle cost is added as well. A path starting at r and ending at t is denoted by an (r,t)-path. We can determine

the combined duty corresponding to the shortest (r, t)-path in network G in $O(m)$ time, where m is the number of arcs. This shortest path is added to the master problem if its cost is negative, while the column generation procedure is terminated otherwise. In order to speed up column generation by trying to generate more than one column per iteration, we solve several shortest path problems by leaving out nodes and arcs of paths already found. For more details, we refer to Freling (1997).

If the set Q of pieces is generated as described previously, i.e. all-pairs shortest paths are included, and all feasible duties that can be constructed using the pieces in Q are input for the combined duty generation, then optimality can be guaranteed at convergence of the column generation procedure. This can be proven in a similar way as Proposition 3.1.

3.5 Independent Crew Scheduling

It is an obvious observation that the specification of vehicle schedules will put certain constraints on the crew schedules (and vice versa). Because vehicles are often much more flexible to schedule than crews, it may be inefficient to schedule vehicles without considering crew scheduling. We can measure the potential benefit of integration with respect to cost efficiency by comparing the cost of the traditional sequential approach, i.e. first solve the SDVSP and then the Crew Scheduling Problem (CSP), with a lower bound on the optimal value of the VCSP. To this end, we define the *independent crew scheduling problem* (ICSP) as follows: given a set of trip tasks corresponding to a set of trips, and given the travelling times between each pair of locations, find a minimum cost crew schedule such that all trip tasks are covered in exactly one duty and all duties satisfy crew feasibility constraints. Note that vehicles are completely ignored in this problem. Clearly, a lower bound can be computed by independently solving the SDVSP and the ICSP and adding the respective optimal values.

If the cost of the sequential approach is much higher than the lower bound, it may be that the crew scheduling solution will improve significantly when considering integration as compared to the sequential approach. On the other hand, we know that there is no need to integrate if the cost of the sequential solution is close to the lower bound. Note that, because the vehicle schedules are identical for the sequential and independent approach, the potential benefit can be measured by comparing the cost of the crew schedules only.

For the ICSP we propose the same formulation as for the CSP, but the difference is that the set of possible duties is much larger, because the vehicle schedule is not given in advance. Furthermore, we know that some duties will never appear in the optimal solution. For example, consider two duties i and j which are equivalent except that j contains an extra dh-task from the depot to the first relief point. Then, duty j will never appear in an optimal solution if $d_i < d_j$. In general, if the cost of a duty increases with the duration, in an optimal solution no duty begins or ends at the depot. So vehicle

movements to and from the depot are not covered. In particular, only duties beginning and ending with a trip task need to be considered.

The algorithm to solve the ICSP is similar to the one of VCSP2 (see Subsection 3.4.1), but instead of combined duties we have only duties.

3.6 Computational Tests for Bus and Driver Scheduling

In this section we present a computational study of integrated bus and driver scheduling. The aim of this study is to investigate the effectiveness of integration as compared to the traditional sequential approach, and the applicability of the proposed techniques in practice.

3.6.1 Data and Parameter Settings

We have used data from the RET, the public transport company in Rotterdam, the Netherlands, that provides passenger services by bus, metro and tram. The data corresponds to bus and driver scheduling of individual bus lines. Some constraints at RET are very complex and particular for this company only. Therefore, we have changed some RET specific constraints into more general constraints. In Subsection 3.7, we consider the real RET restrictions, which will make the problem much more difficult.

The restrictions that we have taken into account, are as follows. A driver can only be relieved by another driver at the start or end of a trip or at the depot and continuous attendance is required. This implies that a trip always corresponds to exactly one trip task and the number of relief points is equal to twice the number of trips. If a driver starts/ends his duty at the depot, there is a sign-on/sign-off time of 11 and 12 minutes, respectively. Another restriction is that every bus waits at least 3 minutes after a trip at the end location before starting the next trip. There are four different types of duties and their properties are given in Table 3.1.

The following points are relevant for all the different approaches proposed in this section.

1. The primary objective is to minimize the sum of buses and drivers used in the schedule. For bus scheduling we have a secondary objective of minimizing a combination of travel and idle time as defined in Subsection 2.1.1. For crew scheduling we do not have a secondary objective. We use fixed costs in order to assure that the minimum number of buses and drivers is the primary objective. For the computational results presented in this section, we only report the number of buses and drivers since these are relevant for our study.

type	1 (tripper)		2 (early)		3 (normal)		4 (late)	
	min	max	min	max	min	max	min	max
# pieces	1	1	2	2	2	2	2	2
start time				10:45	10:46	15:14	15:15	
end time				19:37		22:14		
piece length	0:30	5:00	0:30	5:00	0:30	4:00	0:30	5:00
break length			0:15	1:30	0:15	1:30	0:15	1:30
duty length	0:30	5:00		8:52		7:00		9:30
work time	0:30	5:00		8:52		7:00		9:30

Table 3.1: Properties of all different types of duties

2. The pricing problems are solved separately for each type of duty. We work with reduced networks for each type of duty, that is, only nodes corresponding to pieces of work are considered which could be part of a duty of a certain type.

3. The Lagrangian multipliers are initialized to zero. The maximum number of iterations $k_{\max}$ is initially set to 100 for problem instances with a small number of trips; for instances with a larger number of trips, we take for the independent approach $k_{\max} = 1000$. In every iteration of the column generation algorithm, after solving the master problem, $k_{\max}$ is increased by 3. A greedy heuristic is used to get only non-negative Lagrangian cost variables in the master problem. The initial values of the Lagrangian multipliers for a new master problem are set to the values resulting from the previous master problem, before the greedy heuristic was applied.

4. The column generation is stopped once the difference between the estimate of the lower bound for the overall problem and the current lower bound is less than one percent (0.1 percent for VCSP1) or if no improvement in the lower bound is obtained for 5 iterations (250 iterations for VCSP1). The latter is the so-called *tailing-off criterion*.

5. For the integrated approach with changeovers allowed, we generate 10 feasible solutions, where 5 follow from the last iterations of the column generation and the other 5 follow from the last iterations of a subgradient optimization after the convergence of the column generation step. We choose the best feasible solution. If the gap between the lower bound and the best feasible solution is large, we can easily adapt our approach to compute some additional feasible solutions in the last subgradient optimization, i.e. the one after the convergence of the column generation step.

6. The strategies for generating columns in the pricing problem have been obtained after extensive testing and tuning on different types of problems (see Freling (1997)). We add for all types of duties the first 200 duties with negative reduced cost that

we find. This means that we do not take the duties with the most negative reduced cost. The advantage of this strategy is that we do not have to generate all the duties in all iterations. The disadvantage is that the convergence process is slower, but overall this is faster than the alternative strategy.

All tests are executed on a Pentium 400 pc with 128Mb of computer memory.

We cannot compare our test results with the results of others, because — to the best of our knowledge — only Haase *et al.* (2001) perform a computational study for simultaneous scheduling of vehicles and crews, but the characteristics of their random data and our real life data differ in the following four respects:

- their trip lengths are much longer than ours; on average they have between 2 and 3 trips per duty and about 4 per vehicle, whereas we have, for the largest problem, on average more than 10 trips per duty and 25 per vehicle;
- they consider a network of four lines, while we consider individual bus lines; note that this means that for the same total number of trips, the (average) frequency per line is higher for our problem instances;
- they assume that bus drivers are only allowed to start and end their duty at the depot, whereas in our case drivers can start and end at every relief point (i.e. at the start/end of every trip and at the depot);
- they have only one duty type with 2 pieces and one consisting of a single piece, while we consider three duty types with 2 pieces and one with a single piece.

3.6.2 Results of Different Approaches

In this subsection we show the results of the sequential, independent and integrated approach. For the sequential and the integrated approach we look at the cases with and without changeovers. Six different problem instances of the RET are considered. Their sizes vary between 24 and 238 trips. Notice that 238 trips is a lot for an individual bus line (on average about 6 trips per hour per direction). Furthermore, notice that the instance consisting of 24 trips is called prob24 and so on.

Sequential Approach

In the Tables 3.2 and 3.3 we present the computational results for the sequential method with and without changeovers, respectively. Since the primary objective is to minimize the sum of buses and drivers, we give the lower and upper bounds in terms of this sum.

Because the initial step of the integrated approach is the sequential approach, the difference in the computation time between the sequential and integrated approach is

	prob24	prob42	prob72	prob113	prob148	prob238
lower	12	19	15	23	34	27
upper	12	19	15	24	34	29
buses	6	9	5	8	11	9
drivers	6	10	10	16	23	20
gap (%)	0.00	0.00	0.00	4.17	0.00	6.90

Table 3.2: Results sequential approach RET data - changeovers

	prob24	prob42	prob72	prob113	prob148	prob238
lower	12	19	15	26	45	30
upper	12	20	15	26	45	33
buses	6	9	5	8	11	9
drivers	6	11	10	18	34	24
gap (%)	0.00	5.00	0.00	0.00	0.00	9.09

Table 3.3: Results sequential approach RET data - no changeovers

equal to the computation time of the lower and upper bound phase of the latter approach (which we will present later on). Moreover, the computation times of the sequential approach is negligible. For these reasons, we have not mentioned them here.

Independent Approach

In Table 3.4 we show the computational results for the independent crew scheduling. The top part of the table focuses on the lower bound phase and the bottom part on the upper bound phase. For the lower bound phase, we report the number of nodes and arcs in the piece generation network G', the number of column generation iterations, the average number of pieces of work during the column generation, the number of duties in the final master problem, the total computation times in seconds required for the master problems, the pricing problems, and the lower bound phase, and the resulting lower bound on the number of drivers in the solution, respectively. For the upper bound phase, we report the cost (number of drivers) of the feasible solution, the relative gap between lower and upper bounds and the computation times in seconds for the upper bound phase.

As can be seen from Table 3.4, the gap between the lower and upper bounds for the independent crew scheduling is zero for smaller problem instances, but it increases for the larger instances prob113 (7.14%), prob148 (13.64%) and prob238 (11.11%). As mentioned before, for the larger problem instances we perform more iterations of the subgradient optimization than for the smaller instances. By doing so, we obtain improved lower bounds for these problems. For example, for prob148 we improved the lower bound from 17 to 19. This is possible, because only a small part of the computational effort is in

	prob24	prob42	prob72	prob113	prob148	prob238
nodes	48	84	144	226	296	476
arcs	132	312	460	846	1,376	2,260
iterations	5	6	9	9	15	20
average pieces	463	1,568	4,606	9,891	16,237	52,538
duties	529	1,689	6,026	9,329	28,208	70,307
cpu m.	0	0	3	7	1,261	1,845
cpu p.	1	8	41	136	6,907	14,422
cpu t.	1	8	44	178	8,169	16,274
lower	6	9	9	13	19	16
upper	6	9	9	14	22	18
gap (%)	0.00	0.00	0.00	7.14	13.64	11.11
cpu	0	0	0	50	53	52

Table 3.4: Results independent approach RET data

the master problem and the major part is in the pricing problem. In general, such large gaps in a column generation approach occur due to not generating columns during the search for integer solutions.

Recall from Section 3.5 that we can measure the potential benefit of integration by comparing solutions obtained by the sequential approach with solutions obtained by the independent approach. The minimum number of drivers resulting from model ICSP is a lower bound on the minimum number of drivers overall. Thus, if we compare the number of drivers resulting from the independent approach with the number of drivers in the CSP solution, taking the gap with the lower bound into account, we also have an indication of the potential benefit. That is, we have an indication of how many drivers could be saved from the sequential approach by an integrated one.

Integrated Approach

In Table 3.5 and Table 3.6 we show the computational results for the integrated approach with and without changeovers, respectively. We report the same information as in Table 3.4, except that we also report the number of combined duties in the case of "no changeovers" and the number of buses in the solution.

The gap between the lower and upper bounds is at most 3.57% for the approach with changeovers and 7.32% if changeovers are not allowed. However, we should note that we do not have a guaranteed lower bound in all cases, because of the tailing-off criterion. For prob72 we have computed an additional feasible solution, as described in Subsection 3.6.1. If we would not have done this, the best solution would have been 15 and the gap 6.67%. In the case where changeovers are allowed we have computed the feasible solution

	prob24	prob42	prob72	prob113	prob148	prob238
nodes	48	84	144	226	296	476
arcs	132	312	460	846	1,376	2,260
iterations	5	12	19	44	106	254
average pieces	463	1,568	4,606	9,891	16,237	52,538
duties	512	1,711	7,063	12,644	29,667	80,945
cpu m.	0	2	18	150	911	27,644
cpu p.	1	14	34	837	4,188	214,436
cpu t.	1	18	56	1,024	5,185	242,080
real_lower	12	18	14	23	33	27
upper	12	18	14	23	34	28
buses	6	9	5	8	11	9
drivers	6	9	9	15	23	19
gap (%)	0.00	0.00	0.00	0.00	2.94	3.57
cpu	4	3	37	212	351	1,907

Table 3.5: Results integrated approach on RET data - changeovers

for prob148 and prob238 with the integer programming solver of CPLEX instead of the heuristic of Caprara *et al.* (1999a), because for these problems the heuristic performed poorly. By doing this, we saved one vehicle and three drivers for prob148 and one vehicle and one driver for prob238.

Again, the major computational effort is in the pricing problem. For prob148 (with changeovers), the computation time of the lower bound phase is about 86 minutes. About 80% is spent on the pricing problem, where an average number of 16,237 pieces of work and a total number of 29,667 duties are generated. In the next subsection we give a comparison between the different approaches.

3.6.3 A Comparison of Different Approaches

In this subsection, we are interested in a comparison of the approaches discussed in this section. In Table 3.7 we present the results obtained by the sequential, the independent and the integrated approach for the RET data. For the sequential approach, the upper bounds (lower bounds) are obtained by the summation of the minimum number of vehicles obtained by solving the SDVSP and the (lower bound on the) number of drivers resulting from the Lagrangian heuristic for the CSP.

The number of buses is not mentioned for the independent approach because it is the same as for the sequential approach. The results for the integrated approach correspond to the best lower and upper bounds obtained using different column generation strategies (as mentioned in the previous subsection). For the case with changeovers, comparing

	prob24	prob42	prob72	prob113	prob148	prob238
nodes	48	84	144	226	296	476
arcs	132	312	460	846	1,376	2,260
iterations	3	4	12	9	15	14
average pieces	1,367	4,706	8,716	19,511	33,435	101,187
duties	2,201	3,780	5,092	3,150	8,984	16,536
combined duties	213	438	628	323	1,816	4,972
cpu m.	0	0	7	0	3	12
cpu p.	1	19	289	41	255	3,927
cpu t.	1	19	296	41	258	3,942
real_lower	12	18	14	26	38	30
upper	12	18	14	26	41	32
buses	6	9	5	8	12	9
drivers	6	9	9	18	29	23
gap (%)	0.00	0.00	0.00	0.00	7.32	6.25
cpu	0	0	42	0	29	35

Table 3.6: Results integrated approach on RET data - no changeovers

the sequential with the integrated approach shows that the savings when integrating bus and driver scheduling are at most one driver. This confirms the expectation based on the potential savings resulting from comparing the sequential with the independent approach. In 4 out of 6 test cases, the integrated solution was better than the sequential solution due to the saving of one driver. However, for the case without changeovers a significant saving is obtained with the integrated approach. This also confirms the expectation based on the potential savings resulting from comparing the sequential with the independent approach. In 4 out of 6 test cases, the integrated solution was better than the sequential solution due to the saving of up to 5 drivers at the cost of only one additional bus. Interestingly, for the small problems the integrated approach gives the same results for the cases with and without changeovers, but for the larger problems allowing changeovers can save many drivers and buses.

3.6.4 Conclusions

The results reported in this section show that we can get good solutions within reasonable computation times on a personal computer. We did not expect that it would be useful to integrate under every circumstance. Based on the results we can conclude that the benefit of integration may be significant when changeovers are not allowed. In practice, it often occurs that either changeovers are not possible due to long distances or changeovers are not allowed for legal or technical reasons. When changeovers are allowed in our test

problems, one driver may be saved by considering an integrated approach. Of course, the saving of one driver is considerable when taking into account that we investigated instances for one bus line each. The interpretation of the computational results depends on the ratio between the fixed vehicle and crew costs. If fixed vehicle costs are much higher as compared to crew costs it becomes less attractive to apply the integrated approach. On the other hand, if crew costs are higher the integrated approach becomes more attractive.

Furthermore, we have seen that the results of the independent approach give a good indication of the potential benefit of integration.

Finally, we would like to remark the following. Although, from an employment point of view, it may be interpreted negatively when we say that drivers can be saved by using integration, a different point of view is that the cost savings may allow for an increase in service while no driver needs to loose his or her job.

	changeovers											no changeovers							
	independent			sequential				integrated				sequential				integrated			
	low	upp	driv	low	upp	bus	driv	low	upp	bus	driv	low	upp	bus	driv	low	upp	bus	driv
prob24	12	12	6	12	12	6	6	12	12	6	6	12	12	6	6	12	12	6	6
prob42	18	18	9	19	19	9	10	18	18	9	9	19	20	9	11	18	18	9	9
prob72	14	14	9	15	15	5	10	14	14	5	9	15	15	5	10	14	14	5	9
prob113	21	22	14	23	24	8	16	23	23	8	15	26	26	8	18	26	26	8	18
prob148	30	33	22	34	34	11	23	33	34	11	23	45	45	11	34	38	41	12	29
prob238	25	27	18	27	29	9	20	27	28	9	19	30	33	9	24	30	32	9	23

Table 3.7: Comparison between different approaches for RET data

3.7 Application: RET

In the preceding sections some models and algorithms for integration of vehicle and crew scheduling were discussed and applied to problems of practical size. The test problems were based on real-life problems from the RET, the public transport company in the city of Rotterdam, the Netherlands. In these test problems, we omitted certain complicating constraints that are specific to the RET. In this section, we will apply our approach to the actual RET problems. Although the RET also provides service on tram and metro lines, only bus lines will be considered. For bus lines we can expect a larger benefit of the integration, because the relative difference between crew and vehicle costs is higher than in tram and metro scheduling. Moreover, tram and metro scheduling problems are more restricted because of constraints with respect to the capacity of the rail track and the capacity at the endpoints of the lines.

We will compare our integrated approach with a sequential approach. Furthermore, we will investigate the impact of allowing drivers to change vehicle during a break. Currently, the rule at the RET is that such *changeovers* are only allowed in split duties; they are never allowed in other types of duties. *Split duties* have a long break between the two parts of the duty such that the driver can go home during this break. We show that it is already possible to save crews if for the non-split duties, restricted changeovers are allowed.

The main objective of this study is threefold.

1. We show that our integrated approach can be applied to a practical situation.
2. We make a comparison between the results of the sequential and the integrated approach.
3. We look at the effect of allowing (restricted) changeovers.

3.7.1 Problem Description at the RET

At the RET the whole planning process is solved line-by-line, so there is no inter-lining and the size of the problems is relatively small (up to 259 trips). Almost all lines have two locations (lets call them A and B) and all the trips are from A to B or from B to A. In this section we only consider lines with this property.

For every line there is one given depot and all buses must start and end in that depot. An *unloaded trip* is the driving of a vehicle without passengers. There are three kinds of unloaded trips: from the depot to the start of a trip, from the end of a trip to the depot and between two trips (from one endpoint of the line to the other one). The workday of one driver is called a *(crew) duty*. There are many different types of duties, for example early duties, late duties and split duties. Every duty consists of two pieces with in between a

break. A *piece* consists of consecutive *tasks* (trips, layovers and unloaded trips) performed by one driver on the same bus. A *layover* is the time between two consecutive trips that a driver with bus waits at a start or end location of the line. For the layover there are several restrictions:

- after every trip and unloaded trip there is a layover of at least two minutes;
- there is a minimal layover in a *round-trip*, where a round-trip consists of two consecutive trips from location A to B and back to A; this is the so-called *round-trip-condition*, that will be discussed in more detail below;
- the total layover time in a duty is at least 10% of the length of the duty;
- if the longest piece in a split duty is more than 4 hours, then there is at least once a layover of at least 10 minutes.

Notice that the term *deadheads* as defined in Section 3.1 is not used here. However, unloaded trips and layovers together form these deadheads.

All duties start or end at the depot or at so-called relief locations. A *relief location* is the start or the end location of the line. In some cases both the start and the end location is a relief location and in other cases only one of them is a relief location. Duties starting at the depot always start with a *sign-on* time and the other duties start with a *relief* time. Duties ending at the depot end with a *sign-off* time and the other duties end after a layover. These times are fixed and known in advance. The breaks are only allowed at relief locations. A break is only necessary in duties that have a length greater than a certain minimum.

Furthermore there are restrictions for all 8 types of duties with respect to the earliest and latest starting time, the latest ending time, the minimum length of the break, the maximum spread and the maximum working time. On Sundays there are no split duties. For more details we refer to Huisman (1999).

Currently changeovers are only allowed in split duties and under no circumstances in other duties. In this section we compare this situation with the variant where a changeover is also allowed during a break of a non-split duty, provided that there is enough time to change vehicle. In that case the first piece ends with a layover and the second piece starts with the relief time. Of course, the length of the last layover of the first piece is then at least the relief time, because the driver who takes over the first bus needs this amount of time. As a consequence there is continuous attendance of the vehicle, i.e. there is always a driver present when the bus is outside the depot.

Round-trip-condition

The most important and complicated constraint of the RET is the round-trip-condition. As mentioned before, it is a restriction on the total layover in a round-trip (two consecutive

trips such that a driver drives from location A via location B back to location A). The round-trip-condition requires that the total layover at the locations A and B in one round-trip is at least 10 minutes if the round-trip is 60 minutes or longer and 15% of the travel time if the round-trip is shorter than 60 minutes with a minimum of 5 minutes. It is allowed to violate this condition at most once in every crew duty, but only if the total layover is at least 5 minutes.

In advance, we can construct combinations of three trips that violate the round-trip-condition. We show this in Figure 3.5. We use these combinations in the sequential approach as well as in the integrated approach.

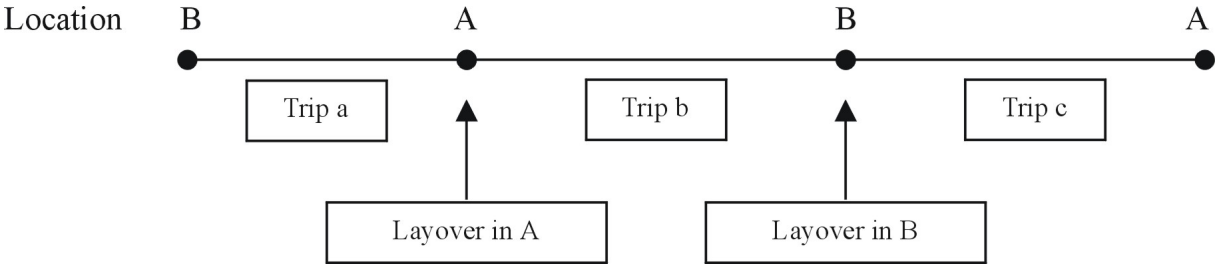

Figure 3.5: Violation of round-trip-condition

In a sequential approach it is not possible to incorporate the possibility of violating the round-trip-condition at most once in every crew duty. The reason is that we have to deal with the round-trip-condition in the vehicle scheduling problem (VSP) and at that stage we do not know anything yet about crew duties. Hence, if we violate the round-trip-condition more than once in the VSP, we may later on end up with an infeasible solution to the crew scheduling problem, because some crew duty has more than one violation of the round-trip-condition.

As we will see in Subsection 3.7.3, in our integrated approach, we can exploit the possibility that the round-trip-condition may be violated once in every crew duty. This is actually the main reason why we can save vehicles by using an integrated approach instead of a sequential one.

3.7.2 Sequential Approach

In this subsection, we discuss the differences between the sequential approach described in Chapter 2. These differences only occur in the VSP due to the round-trip-condition. Let $N = \{1, 2, ..., n\}$ be the set of trips, numbered according to increasing starting time, and let $E = \{(i, j) \mid i < j,\ i, j \text{ compatible},\ i \in N, j \in N\}$ be the set of arcs corresponding to unloaded trips and layovers. The nodes r and t both represent the depot. We define the vehicle scheduling network $G = (V, A)$, which is an acyclic directed network with nodes

$V = N \cup \{r, t\}$, and arcs $A = E \cup (r \times N) \cup (N \times t)$. Let R be the set of combinations of trips that violate the round-trip-condition. Every combination consists of three trips, that we will refer to as a, b and c like in Figure 3.5.

Let c_{ij} be the vehicle cost of arc $(i, j) \in A$. Fixed costs for using a vehicle are included in the cost of arcs (r, i) for all $i \in N$. For the remainder of this section, we assume that the primary objective is to minimize the number of vehicles. This means that the fixed costs are high enough to guarantee that this minimum number will be achieved.

Using decision variables y_{ij}, with $y_{ij} = 1$ if a vehicle covers trip j immediately after trip i, $y_{ij} = 0$ otherwise, the VSP with round-trip-condition can be formulated as follows.

(VSP*):

$$\min \quad \sum_{(i,j)\in A} c_{ij} y_{ij} \tag{3.15}$$

$$\text{s.t.} \quad \sum_{\{j:(i,j)\in A\}} y_{ij} = 1 \qquad \forall i \in N, \tag{3.16}$$

$$\sum_{\{i:(i,j)\in A\}} y_{ij} = 1 \qquad \forall j \in N, \tag{3.17}$$

$$y_{ab} + y_{bc} \leq 1 \qquad \forall (a, b, c) \in R, \tag{3.18}$$

$$y_{ij} \in \{0, 1\} \quad \forall (i, j) \in A. \tag{3.19}$$

Constraints (3.16) and (3.17) are the standard constraints. Recall that they assure that each trip is assigned to exactly one predecessor and one successor, that is, these constraints guarantee that the network is partitioned into a set of disjoint paths from r to t. Constraints (3.18) assure that the round-trip-condition is met, because the trips a, b and c cannot be assigned to the same vehicle.

We solve this model with subgradient optimization and Lagrangian relaxation, where we relax constraints (3.18). Then the remaining subproblem is a quasi-assignment problem, which is solved in every iteration of the subgradient optimization with an auction algorithm (see Freling *et al.* (2001b)). In this way we get a lower bound on the optimal solution. Furthermore, we also compute a feasible solution in every iteration by changing the solution of the subproblem such that the round-trip-condition is not violated anymore. This requires at least one more vehicle. We terminate the subgradient optimization if the gap between the lower bound and the value of the best feasible solution indicates that we have found an (almost) optimal solution.

3.7.3 Integrated Approach

In this subsection we discuss the integrated approach which we applied to the problems of the RET. Since changeovers are sometimes allowed, we use the model and algorithm of the general case described in Subsection 3.3. Due to several restrictions of the RET

modifications in the algorithm are necessary. These modifications occur in the generation of pieces, such that we also have to implement the round-trip-condition in the generation of pieces. Recall that if we want to generate duties with at most one violation of the round-trip-condition, we have to generate pieces with no violation and pieces with at most one violation. Therefore, we explain both situations. Finally, we discuss how we implement the restrictions with respect to relieving.

Generation of Pieces Without Violation of the Round-trip-condition

We generate pieces using a network as described in Subsection 3.3.5. Figure 3.6 illustrates such a network with six trips, six unloaded trips from and six to the depot and eight layovers. The trips (2,3,4) and (4,5,6) are combinations that violate the round-trip-condition.

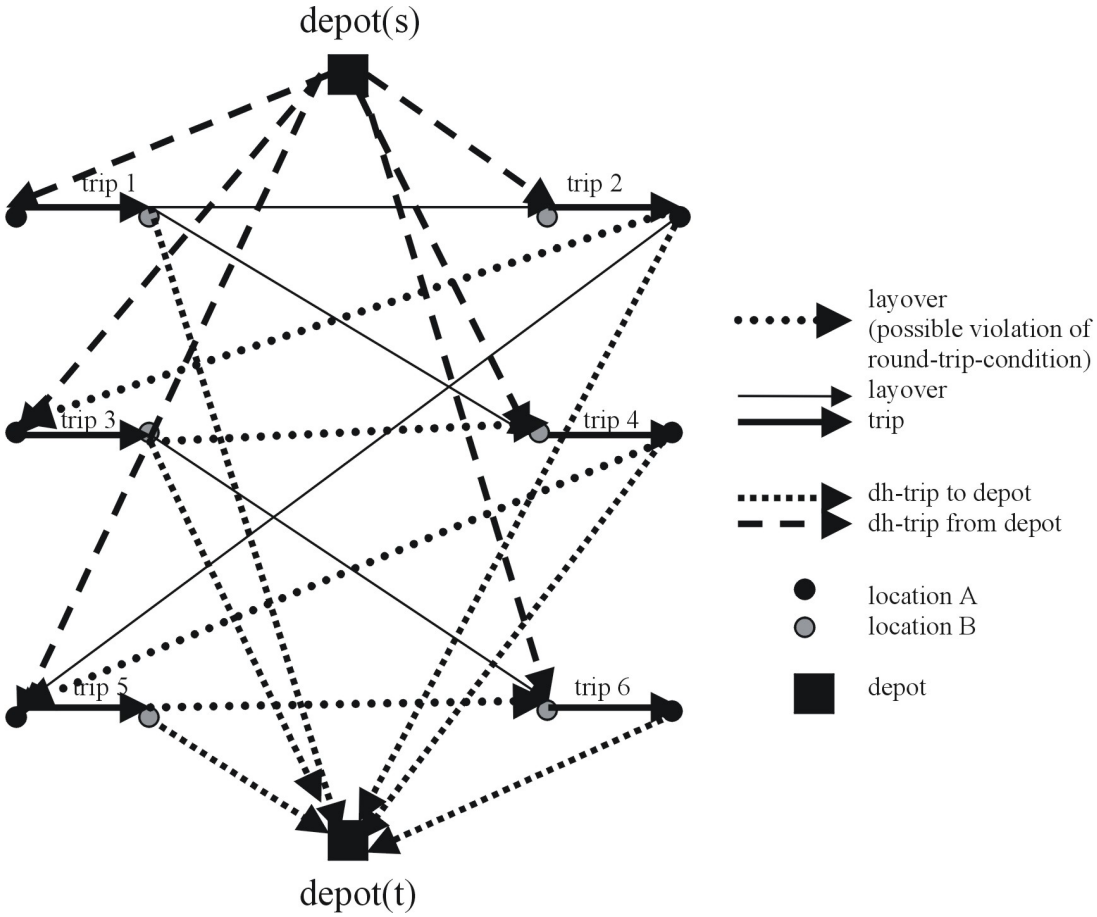

Figure 3.6: Piece network

If we would not have the problem of the round-trip-condition, finding all shortest paths is very easy (see Subsection 3.3.5), but in our case solving this restricted shortest

path problem is not very straightforward. We have considered two ways of solving this problem, namely by an exact procedure or by a fast heuristic.

Exact Shortest Path Algorithm

Because we want to generate pieces where the round-trip-condition is never violated, we define a new acyclic network $G" = (N", A")$. The set of nodes consists of two nodes for every trip (one which we can reach from the depot and one where we can go to the depot) and we have a node for every unloaded trip. The source and the sink correspond again to the depot and the set of arcs is now defined by unloaded trips from and to the depot and a combination of a trip and a layover or only a trip. For the same example as in Figure 3.6 we have the network $G"$ in Figure 3.7.

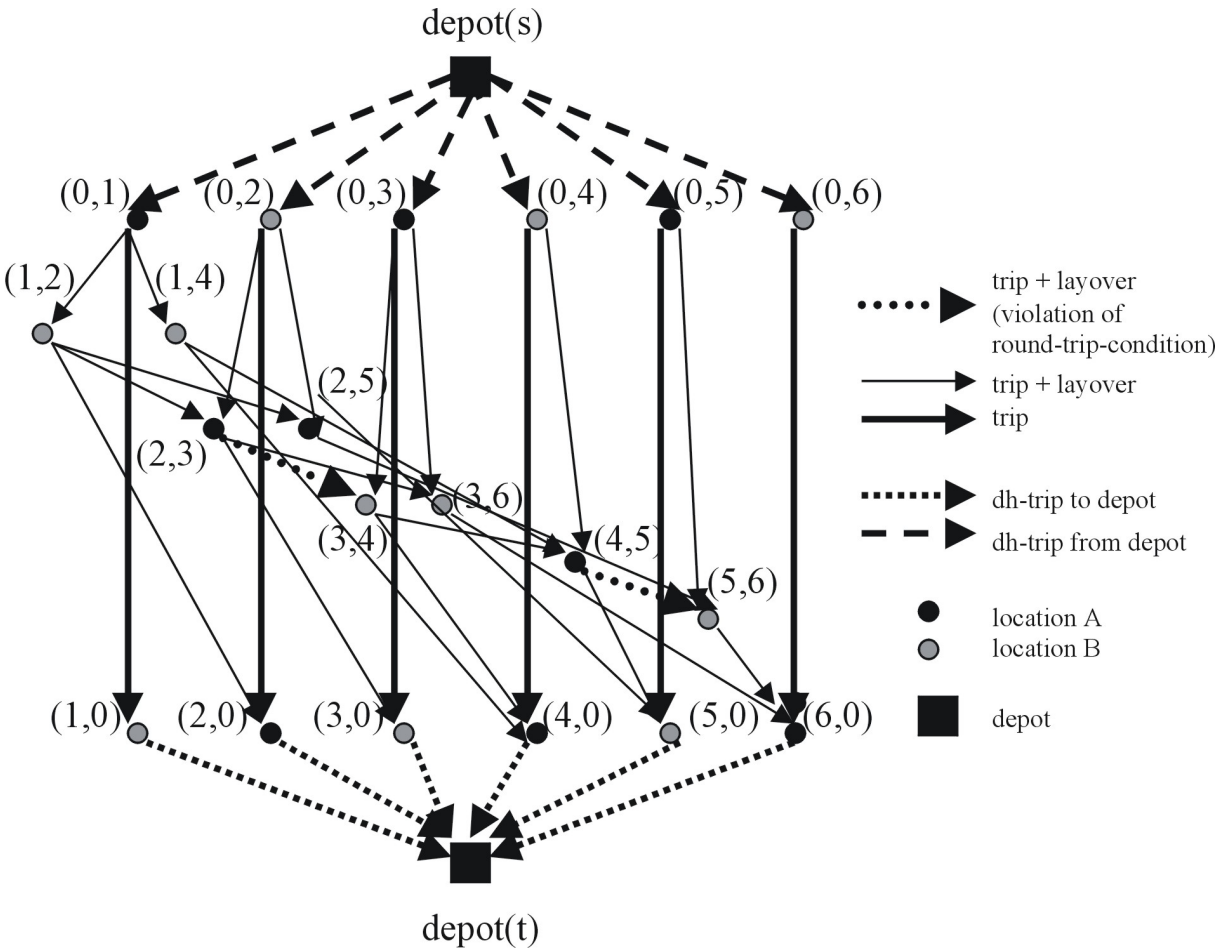

Figure 3.7: Better representation of piece network

We can generate the pieces now by just deleting the arcs corresponding to violations of the round-trip-condition (in the example the arcs from (2,3) to (3,4) and from (4,5) to (5,6)) and solving the shortest path problem between every node u of type (0,..) and

every possible end node v. For all feasible paths from u to v, three additional paths are considered like before, namely path $r, u, ..., v$, path $u, ..., v, t$ and path $r, u, ..., v, t$.

Heuristic for Generating Pieces

Because we have to generate the pieces many times, we have developed a heuristic for generating the pieces. We do this by solving two "shortest" path problems in the network G' between every pair of nodes. We explain this heuristic via the example in Figure 3.6. For solving the first "shortest" path problem we delete the arcs from 2 to 3 and from 4 to 5 (the parts of the violation of the round-trip-condition at location A) and compute after that the shortest path. For the other "shortest" path we delete the arcs from 3 to 4 and from 5 to 6 (parts corresponding to location B). Finally, we choose the best of the two "shortest" paths. In this way, we always get a piece that never violates the round-trip-condition, although it is possible that we do not get the piece with the lowest reduced cost. In the example, the piece with the lowest reduced cost can be a piece consisting of trips 3, 4 and 5, which cannot be the solution of our heuristic. However, from computational experiments we have concluded that this heuristic works very well for the problems at the RET.

Generation of Pieces with at Most One Violation of the Round-trip-condition

In this case, the pieces can have at most one violation of the round-trip-condition. Of course, we can generate here also pieces with no violation, for example, if there is no violation possible or if it is not attractive to violate the round-trip-condition. We can easily adapt the exact algorithm and the heuristic for this case. Previously, we computed the shortest path in an acyclic network from u to v by just computing the shortest path from u to every node u' connected with u and so on. In this case, we have to remember two paths to u', namely the shortest path with at most one violation of the round-trip-condition and the shortest path without violation of the round-trip-condition. We do this for all nodes between u and v such that we can add a violation at the moment we want.

Relief Restrictions

Recall that the RET has several restrictions with respect to relieving:

1. a duty that does not start at the depot, starts with a relief time and at a relief location;

2. a duty that does not end at the depot, ends after a layover and at a relief location.

These restrictions also hold for the start/end of every piece in the variants where changeovers are not always allowed. The difficulty of these restrictions is that a duty

cannot end at every relief location, because the time between the departure and the arrival must be at least the relief time. We solve this problem in the following way: first we construct a set of all possible layovers where a duty cannot end. We can do this in advance and when constructing the pieces we assure that a piece never ends at such a layover. Of course, we also introduce this restriction now for the pieces before the break, although that is not required. This is not a problem, because if a duty ends with a layover before the break, there is another duty with the same properties that ends with a trip.

3.7.4 Computational Results on RET Data

This subsection deals with a computational study of the integration of bus and driver scheduling. We have used data of two individual bus lines from the RET (line 35 and 38) of the winter-timetable 1998/1999. The routes of these lines are shown in Figures 3.8 and 3.9, respectively. In these figures, the location of the depot is denoted by a circle.

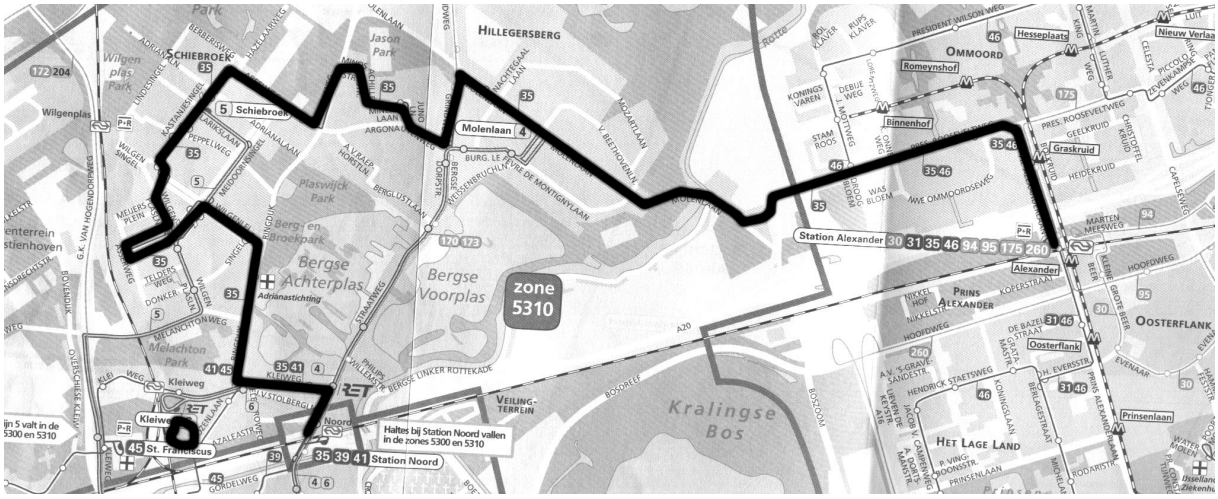

Figure 3.8: Route of bus line 35 (winter-timetable 1998/1999)

These lines are used to compare the different approaches for bus and driver scheduling, that is, we investigate the effectiveness of integration as compared to the traditional sequential approach and we compare the different variants of allowing changeovers. The following points are relevant for all the different approaches proposed in this section.

1. The objective is to minimize the number of drivers and buses in the schedule, therefore we use fixed costs. At the RET, the fixed cost for a driver is about equal to the fixed cost for two buses. As an indication, we take as fixed cost for the bus 600 cost units and for the driver 1,200 cost units. For a fair comparison between the sequential and integrated approach, we also need variable vehicle costs per minute for every unloaded trip and layover, because if we do not add variable vehicle costs,

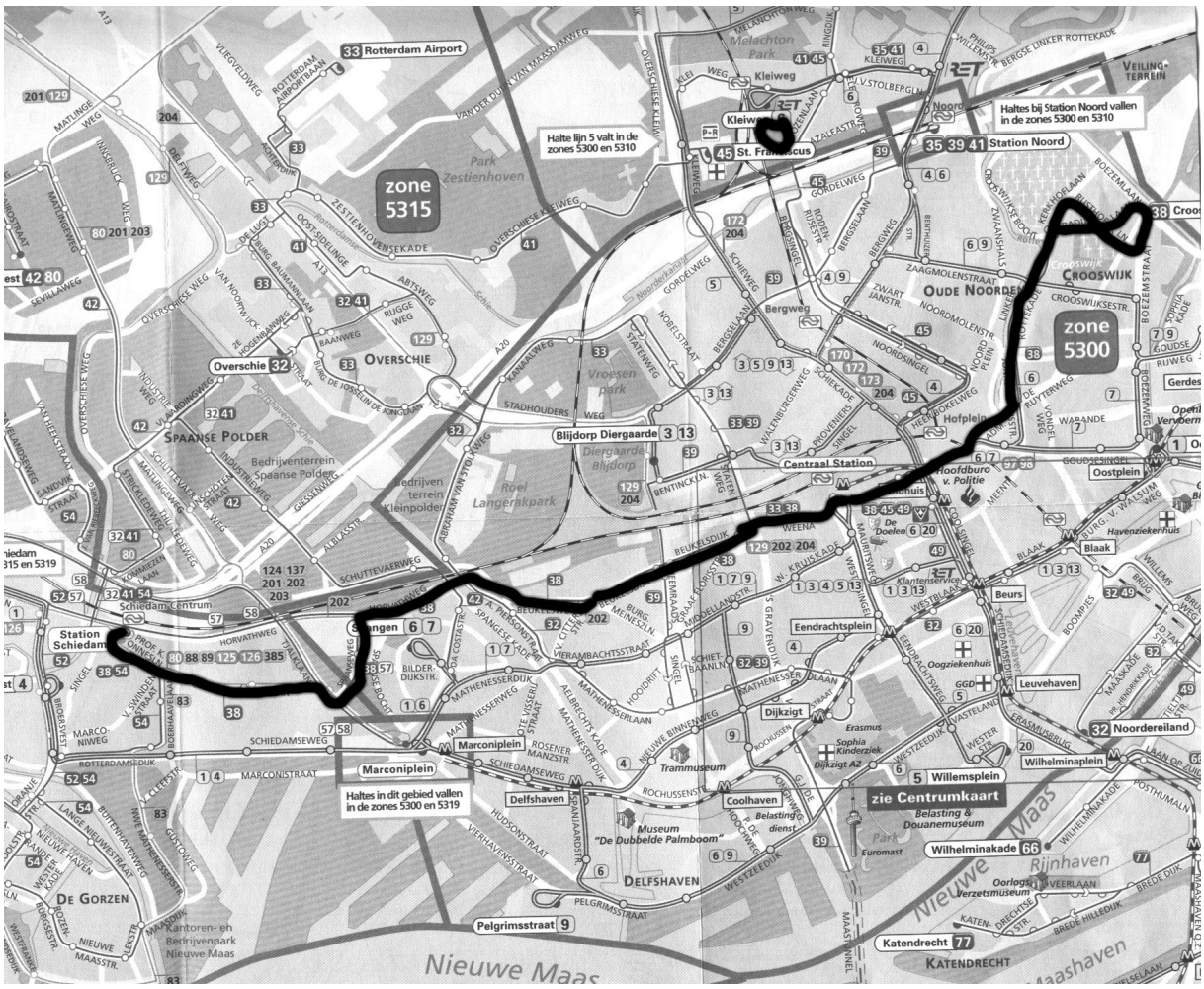

Figure 3.9: Route of bus line 38 (winter-timetable 1998/1999)

the VSP will only minimize the number of buses during peak hours and the number of buses during off-peak hours may be much too large. Because every bus needs a driver, in the off-peak hours the number of drivers is far from optimal. Therefore the total number of drivers is much higher than if we add variable vehicle costs. We take as variable vehicle cost one cost unit per minute.

2. The different variants, as described in Subsection 3.7.1, are called a and b, where a represents that changeovers are only allowed if there is enough time to relief one driver by another and b represents the current RET rule that changeovers are allowed for split duties and not for other types of duties.

3. The column generation is terminated once the difference between the real and the current lower bound is less than 0.1% or if no significant improvement in the current lower bound is obtained for a certain number ($max\_tailing\_off$) of iterations; this is called *tailing-off criterion*. We considered an improvement as significant if it was at

least 5%.

4. For the integrated approach we generate 10 feasible solutions, where 5 follow from the last iterations of the column generation and the other 5 follow from the last iterations of the last subgradient optimization.

5. The strategies for generating columns in the pricing problem are obtained after extensive testing and tuning (see Freling (1997)).

6. All tests are executed on a Pentium II 350 pc with 64Mb of computer memory.

In Table 3.8 we summarize the size of the problem and the important properties for lines 35 and 38. We have only considered the problems for weekdays.

	Line 35	Line 38
number of trips	131	259
number of round-trip-condition restrictions	6	39
number of relief restrictions	48	93
types of duties	8	8
number of relief locations	1	1
unloaded trips between two end locations allowed	yes	no
max_tailing_off	50	15

Table 3.8: Important properties of line 35 and 38

The lines 35 and 38 are representative for the RET, where line 35 is of middle size and line 38 is of large size (e.g. line 38 is the largest line of the RET if we look at the number of trips). In Table 3.9, we show the results of lines 35 and 38. For these results we used the heuristic for generation the pieces described in Subsection 3.7.3.

The table consists of four parts, two for the sequential and two for the integrated approach. In the first part we summarize the results for the lower bound phase of the VSP (number of buses), the CSP (number of drivers) and of the total vehicle and crew costs. In the second part we show the characteristics of the feasible solution found by the sequential approach. We do not give the computation times for the sequential approach, because they can be neglected. For the integrated approach we give for line 35 the lower bound after convergence and for line 38 the lower bound of the last iteration. The lower bound phase for line 38 terminated, because the tailing-off criterion is satisfied. We also give the number of column generating iterations and the total computation time for the lower bound phase. In the last part we give the number of buses, drivers and the total costs for the best of the ten feasible solutions. For line 35 we also give the relative gap between the lower and the upper bound. We cannot do this for line 38, because there we do not have a lower bound for the total problem but only for a subproblem, since the

		Line 35		Line 38	
		variant a	variant b	variant a	variant b
	lower: buses	10	10	14	14
	lower: drivers	16	17	21	23
	lower: total costs	26,634	27,834	35,653	38,053
sequential	number of buses	10	10	14	14
	number of drivers	16	17	23	24
	total costs	26,638	27,838	38,073	39,273
	lower: total costs	24,945	24,968	35,553	36,040
	iterations	24	20	19	19
	cpu (sec.)	279	232	2,433	2,502
integrated	number of buses	10	10	13	13
	number of drivers	15	15	22	23
	total costs	25,495	25,573	36,438	37,755
	gap (%)	2.16	2.37	-	-

Table 3.9: Results line 35 and 38

algorithm terminated before convergence. This is the reason why we have also computed a "real" lower bound, obtained when the tailing-off criterion was not used. These results are given in Table 3.10.

In Table 3.10 one can see that we can even compute for the largest RET problem a "real" lower bound within 5 hours. Furthermore, in both variants the solutions are better than the solution that was obtained when the tailing-off criterion was used.

For all the variants the integrated approach gave better results than the sequential approach and the gap for the integrated approach is always lower than 10%. For example in Table 3.10, we saved two drivers and one bus (about 8% reduction in costs) for the variants a and b and we still had a gap of 5.39% and 8.37%, respectively. Because of the possibility to violate the round-trip-condition once in every crew duty, we can even save buses in the integrated approach. (Without this possibility only drivers could have been saved.) Another important result is that we find solutions with less drivers if more flexibility for changeovers is allowed. We have to be careful with interpreting these results, however, because the difference in the lower bounds is negligible. Hence, it is possible that our approach simply works better in these variants, while the optimal solutions are actually the same.

In Huisman (1999) one can find more results with some other variants and cost structures. For example, we took different costs for split duties and we added an extra restriction on the maximum length of the break. We also looked at a problem where no changeovers are allowed at all and where changeover are always allowed. For the first situation we applied model VCSP2 (see Subsection 3.4).

		variant a	variant b
sequential	lower: buses	14	14
	lower: drivers	21	23
	lower: total costs	35,653	38,053
	number of buses	14	14
	number of drivers	23	24
	total costs	38,073	39,273
integrated	lower: total costs	33,152	33,273
	iterations	105	113
	cpu (hour)	4.7	4.9
	number of buses	13	13
	number of drivers	21	22
	total costs	35,040	36,311
	gap (%)	5.39	8.37

Table 3.10: Results line 38 after convergence

Exact Computation of the Lower Bound

When interpreting the above results, one should keep in mind that we used a heuristic for generating the pieces, i.e. we did not necessarily take the pieces with the lowest reduced cost. So it is possible that we found that there were no duties left with negative reduced cost with the pieces we generated, whereas a combination of two pieces with negative reduced cost did exist. In that case, we did not find a real lower bound for the total problem, but only an approximation of this lower bound. Therefore we have compared this approximation with the lower bound obtained by using the exact shortest path algorithm, described in Subsection 3.7.3, to generate the pieces. This was done for several lines and variants. A typical example of our findings is given in Table 3.11, which summarizes the results for line 35, variant b.

	heuristic	exact (normal)	exact (equality)
lower: total costs	24,968	24,846	25,052
iterations	20	40	66
cpu (sec.)	232	2,198	4,138

Table 3.11: Comparison of the heuristic and exact method to compute the lower bound

The column “exact (normal)” refers to the situation that the method and all parameters are the same as for the heuristic. This means that the exact lower bound is lower than the lower bound computed with the heuristic, although not much. Moreover, we can obtain a better exact lower bound if we do not compute the Lagrangian relaxation with “greater-than-or-equal-to” signs in constraints (3.4) - (3.7) in model (VCSP1), but

with equality signs instead. The results of this approach are given in the column "exact (equality)". We see that the lower bound of the heuristic is actually a real lower bound.

3.7.5 Conclusions

In this section we applied an integrated approach to vehicle and crew scheduling at the RET. We handled complicating constraints, which have not been considered in the literature before. We showed the results for two individual bus lines, including the RET bus line with the largest number of trips. For these lines the integrated problem could be solved in a reasonable amount of time, where the gap between the lower bound and the best feasible solution was less than 10% in all cases. The main conclusion is that we can save vehicles and/or crews by integrating the vehicle and crew scheduling problem, which may lead to a big decrease in costs. Especially, in the case that (almost) no changeovers are allowed, integration is very attractive.

Another important result is that sometimes it is indeed possible to reduce the total costs by allowing changeovers more often.

3.8 Summary

In this chapter we discussed models, relaxations and algorithms for an integrated approach to vehicle and crew scheduling for an urban mass transit system with a single depot. We proposed new mathematical formulations for integrated vehicle and crew scheduling problems and we discussed corresponding Lagrangian relaxations and Lagrangian heuristics. To solve the Lagrangian relaxations, we used column generation applied to set partitioning type of models.

Furthermore, we addressed the effectiveness of integration in different situations. We showed that integration is especially effective in the situation where changeovers are not allowed, i.e. a driver is not allowed to change from vehicle to another one during his break.

Finally, a computational study using real life data showed the applicability of the proposed techniques to practical problems. We could handle some complicating constraints and the problems could solved in still reasonable computation times.

Chapter 4

Integrated Vehicle and Crew Scheduling (Multiple-Depot)

In this chapter, partly based on Huisman *et al.* (2003), we consider the suburban/extra-urban transit system with multiple depots, where we investigate the savings of using an integrated approach instead of a sequential one. It is generally expected that the savings of using an integrated approach in a suburban/extra-urban transit system are much more significant than in an urban mass transit system, since there are much less opportunities to relief one driver for another one in such a way that both drivers can enjoy their break or start/finish their duty. These reliefs are only allowed at depots and certain other specified locations, which are much further away from each other than in the urban context. If first an optimal vehicle schedule is constructed, there can be vehicles which do not pass a relief location for hours. Therefore, it is very well possible that there does not exist a feasible crew schedule at all, or more probably, that the crew schedule will be very inefficient.

In this chapter we extend the mathematical model and the solution approach which we developed for the single-depot case in the previous chapter to the multiple-depot setting. However, in contrast to Chapter 3 we only consider the general case where changeovers are allowed. The solution approach is thus again based on Lagrangian relaxation in combination with column generation. The column generation is used to generate a set of duties, while Lagrangian relaxation is used to solve the master problem. Finally, Lagrangian heuristics are used to compute feasible solutions. Furthermore, we formulate another model which is an extension of the model for the single-depot case proposed by Haase *et al.* (2001) and we show the relation between both models. We also develop an algorithm for this model, which is based on the same ideas as the algorithm for the first model. However, an important difference between the single-depot and multiple-depot case is that in the latter one the underlying vehicle scheduling problem is NP-hard (see Bertossi *et al.* (1987)), while in the former it can be solved in polynomial time. Of course, the underlying crew scheduling problem is NP-hard in both cases (see Fischetti *et al.* (1989)).

For a literature review on integrated vehicle and crew scheduling we refer to Chapter 3. Recall that there is only one paper in the literature, Gaffi & Nonato (1999), dealing with integration in the multiple-depot case. The main difference with Gaffi & Nonato (1999) is that the two proposed algorithms in this chapter are very general and can thus be used for different applications in the area of suburban and extra-urban mass transit systems. It is even possible to use them for urban transit systems, although there the benefits of using an integrated approach are much smaller as explained earlier. To test the algorithms, we have generated some random data instances for the suburban/extra-urban mass transit system, which are based on our experience with real-world problems. These instances are much more realistic for a suburban/extra-urban mass transit system than randomly generated test instances that have been proposed in the literature before. Furthermore, we compare the two algorithms and we also provide lower bounds for small and medium-sized problem instances. Although we were not able to compute lower bounds for large problem instances, the algorithms can still be used to get feasible solutions, which can be compared with the sequential approach.

The chapter is organized as follows. In Section 4.1 we give a comprehensive problem definition and discuss some of the basic assumptions we make. In Sections 4.2 and 4.3, we discuss a mathematical model and an algorithm for the integrated multiple-depot vehicle and crew scheduling problem, respectively. Notice that these sections are in some sense extensions of Chapter 3. However, it is possible to read this chapter without having read the previous one, because some definitions and notation will be recalled. The second formulation and algorithm is given in Section 4.4. Furthermore, we provide some computational results on real-world and randomly generated data instances in Section 4.5. Finally, in Section 4.6, we discuss methods to split large instances into several smaller ones such that a large real-world instance can be solved. This section is based on joint work with De Groot and Wagelmans (see De Groot (2003)). The chapter is concluded with a short summary (Section 4.7)

4.1 Problem Definition

The *multiple-depot vehicle and crew scheduling problem* (MD-VCSP) combines the *crew scheduling problem* (CSP) and the *multiple-depot vehicle scheduling problem* (MDVSP). Given a set of *trips* within a fixed planning horizon, it minimizes the total sum of vehicle and crew costs such that both the vehicle and the crew schedule are feasible and mutually compatible. Each trip has fixed starting and ending times and can be assigned to a vehicle and a crew member from a certain set of depots. Of course, if every trip can only be assigned to a vehicle and a crew member from one depot, the problem decomposes to a number of single-depot vehicle and crew scheduling problems. Furthermore, the travelling times between all pairs of locations are known.

A vehicle schedule is feasible if:

- all trips are assigned to exactly one vehicle;
- each trip is assigned to a vehicle from a depot that is allowed to drive this trip.

The vehicle costs consist of a fixed component for every vehicle and variable costs for idle and travel time. It is allowed that a vehicle returns to a depot between two trips if there is enough time to do this.

From a vehicle schedule it follows which trips have to be performed by the same vehicle and this defines so-called vehicle *blocks*. The blocks are subdivided at *relief points*, defined by location and time, where and when a change of driver may occur and drivers can enjoy their break. A *task* is defined by two consecutive relief points and represents the minimum portion of work that can be assigned to a crew. These tasks have to be assigned to crew members. The tasks that are assigned to the same crew member define a crew *duty*. Together the duties constitute a crew schedule. Like before we only consider the basic crew scheduling problem, where a schedule is feasible if (1) each task is assigned to one duty, and (2) each duty is a sequence of tasks that can be performed by a single crew, both from a physical and a legal point of view. In particular, each duty must satisfy several complicating constraints corresponding to work load regulations for crews. Typical examples of such constraints are maximum working time without a break, minimum break duration, maximum total working time, and maximum duration. These constraints can differ between different types of duties, e.g. early, split and late duties. The cost of a duty is usually a combination of fixed costs such as wages, and variable costs such as overtime payment. Finally, a *piece (of work)* is defined as a sequence of tasks on one vehicle block without a break that can be performed by a single crew member without interruption.

We make five assumptions, which are discussed one by one below.

1. Each vehicle has its own depot, which means that a vehicle starts and ends in the same depot. The number of vehicles used per depot is unlimited.
2. All crew have their own depot, which means that a duty of a single crew member has only tasks on vehicles from that depot. However, it is not necessary that every duty starts and ends in this depot.
3. The feasibility of a piece only depends on its duration of this sequence, which is limited by a minimum and maximum piece length.
4. There is *continuous attendance*, i.e. there is always a driver present if the bus is outside the depot. However, vehicle attendance at the depot is not necessary.
5. *Changeovers*, which is the change of vehicle of a driver during his break, are allowed.

The last two assumptions imply that if a driver has no changeover, i.e. before and after the break he drives the same vehicle, there should be another driver on this vehicle during the break of the former driver, since there is otherwise nobody attending this vehicle.

We distinguish between two types of tasks, viz. *trip tasks* corresponding to trips, and *dh-tasks* corresponding to deadheading. A *deadhead* is a period that a vehicle is moving to or from the depot, or a period between two trips that a vehicle is outside of the depot (possibly moving without passengers). All trip tasks need to be covered by a crew member, while the covering of dh-tasks depends on the vehicle schedules and determines the compatibility between vehicle and crew schedules. In particular, each dh-task needs to be assigned to a crew if and only if its corresponding deadhead is assigned to a vehicle.

4.2 Mathematical Formulation

In this section, we propose a mathematical formulation for the MD-VCSP under the assumptions stated in Section 4.1.

Let $N = \{1, 2, ..., n\}$ be the set of trips, numbered according to increasing starting time, and let $E = \{(i, j) \mid i < j,\ i, j \text{ compatible},\ i \in N, j \in N\}$ be the set of deadheads. Define D as the set of depots and let r^d and t^d both represent depot d. We define the vehicle scheduling network $G^d = (V^d, A^d)$, which is an acyclic directed network with nodes $V^d = N^d \cup \{r^d, t^d\}$, and arcs $A^d = E^d \cup (r^d \times N^d) \cup (N^d \times t^d)$. Note that N^d and E^d are the parts of N and E corresponding to depot d, since it is not necessary that all trips can be served from every depot. Let c^d_{ij} be the vehicle cost of arc $(i, j) \in A^d$, which is usually some function of travel and idle time. Furthermore, a fixed cost for using a vehicle can be added to the cost of arcs (r^d, i) or (j, t^d) for all $i, j \in N^d$.

To reduce the number of constraints, we assume that a vehicle returns to the depot if it has an idle time between two consecutive trips which is long enough to let it return. In that case the arc between the trips is called a *long arc*; the other arcs between trips are called *short arcs*. Denote $A^{sd} \subset A^d$ and $A^{ld} \subset A^d$ as the sets of short and long arcs, respectively.

Furthermore, K^d denotes the set of all feasible duties corresponding to depot d and f^d_k denotes the crew cost of duty $k \in K^d$, respectively. We assume that deadheads to and from the depot correspond to one dh-task each. Suppose $(i, j) \in A^{ld}$ and let el_i and bl_j denote the ending and starting location of trips i and j, respectively. Then we let $K^d(i, t^d)$ and $K^d(r^d, j)$ denote the set of duties covering the dh-task from el_i to depot d and from depot d to bl_j, respectively. Furthermore, $K^d(i)$ denotes the set of duties covering the trip task corresponding to trip $i \in N^d$, which means that we assume that a trip corresponds to exactly one task. $K^d(i, j)$ denotes the set of duties covering dh-tasks corresponding to deadhead $(i, j) \in A^{sd}$. Decision variable y^d_{ij} indicates whether an arc (i, j) is used and assigned to depot d or not, while x^d_k indicates whether duty k corresponding to depot d

is selected in the solution or not. The MD-VCSP can be formulated as follows.

(MD-VCSP1):

$$\min \quad \sum_{d\in D}\sum_{(i,j)\in A^d} c_{ij}^d y_{ij}^d + \sum_{d\in D}\sum_{k\in K^d} f_k^d x_k^d \tag{4.1}$$

$$\text{s.t.} \quad \sum_{d\in D}\sum_{\{j:(i,j)\in A^d\}} y_{ij}^d = 1 \qquad \forall i \in N, \tag{4.2}$$

$$\sum_{d\in D}\sum_{\{i:(i,j)\in A^d\}} y_{ij}^d = 1 \qquad \forall j \in N, \tag{4.3}$$

$$\sum_{\{i:(i,j)\in A^d\}} y_{ij}^d - \sum_{\{i:(j,i)\in A^d\}} y_{ji}^d = 0 \qquad \forall d \in D, \forall j \in N^d, \tag{4.4}$$

$$\sum_{k\in K^d(i)} x_k^d - \sum_{\{j:(i,j)\in A^d\}} y_{ij}^d = 0 \qquad \forall d \in D, \forall i \in N^d, \tag{4.5}$$

$$\sum_{k\in K^d(i,j)} x_k^d - y_{ij}^d = 0 \qquad \forall d \in D, \forall (i,j) \in A^{sd}, \tag{4.6}$$

$$\sum_{k\in K^d\left(i,t^d\right)} x_k^d - y_{it^d}^d - \sum_{\{j:(i,j)\in A^{ld}\}} y_{ij}^d = 0 \qquad \forall d \in D, \forall i \in N^d, \tag{4.7}$$

$$\sum_{k\in K^d\left(r^d,j\right)} x_k^d - y_{r^dj}^d - \sum_{\{i:(i,j)\in A^{ld}\}} y_{ij}^d = 0 \qquad \forall d \in D, \forall j \in N^d, \tag{4.8}$$

$$x_k^d, y_{ij}^d \in \{0,1\} \quad \forall d \in D, \forall k \in K^d, \forall (i,j) \in A^d. \tag{4.9}$$

The objective is to minimize the sum of total vehicle and crew costs. The first three sets of constraints, (4.2)-(4.4), correspond to the formulation of the MDVSP. In fact, it is not necessary to have both constraints (4.2) and constraints (4.3), since constraints (4.2) and (4.4) imply constraints (4.3). However, it is useful to have both sets of constraints when we relax constraints (4.4), as will be done in the next section. Constraints (4.5) assure that each trip task will be covered by a duty from a depot if and only if the corresponding trip is assigned to this depot. Furthermore, constraints (4.6), (4.7) and (4.8) guarantee the link between dh-tasks and deadheads in the solution, where deadheads corresponding to short and long arcs in A^d are considered separately. In particular, constraints (4.6) guarantee that each deadhead from i to j is covered by a duty in the set $K^d(i,j)$ if and only if the corresponding short arc is in the vehicle solution. The other two constraint sets, (4.7) and (4.8), ensure that the dh-tasks from el_i to t^d and from r^d to bl_j, possibly corresponding to long arc $(i,j) \in A^d$, are both covered by one duty if and only if the corresponding deadheads are in the solution. Note that the structure of these last three sets of constraints is such that each constraint corresponds to the possible selection of one duty from a large set of duties, where the fact whether or not a duty has to be selected depends on the values of the corresponding y variables.

4.3 Algorithm

The algorithm we propose to solve model MD-VCSP1, is a combination of column generation and Lagrangian relaxation and is an extension of the algorithm proposed in Section 3.3.3 for VCSP1. An outline of the algorithm is shown in Figure 4.1.

Step 0: Initialization
Solve MDVSP and CSP for every depot and
take as initial set of columns the duties in the CSP-solution.
Step 1: Computation of dual multipliers
Solve a Lagrangian dual problem with the current set of columns.
This gives a lower bound for the current set of columns.
Step 2: Deletion of columns
If there are more columns than a certain minimum amount, then delete columns with positive reduced cost greater than a certain threshold value.
Step 3: Generation of columns
Generate columns with negative reduced cost.
Compute an estimate of a lower bound for the overall problem.
If the gap between this estimate and the lower bound found in Step 1 is small enough (or another termination criterion is satisfied), go to Step 4; otherwise, return to Step 1.
Step 4: Construction of feasible solution
Solve a second Lagrangian dual problem with the set of columns generated in Step 3, where the optimal solution of the subproblem gives feasible vehicle schedules. Solve for each depot the crew scheduling problem corresponding to the feasible vehicle schedules.

Figure 4.1: Solution method for MD-VCSP1

First, we compute a feasible solution by using the sequential approach, which means we compute the optimal solution of the MDVSP and afterwards, we solve for each depot a CSP given the vehicle schedule for that depot. To solve the MDVSP, we use the model described in Section 2.2 and the all-purpose solver CPLEX. The approach we used to solve the CSP, is described in Section 2.3.

The main part of the algorithm is used to compute a lower bound and we use therefore a column generation algorithm. Like in the previous chapter we solve the *master problem* with Lagrangian relaxation. The details will be discussed in Subsection 4.3.1.

Furthermore, we generate the duties in the column generation subproblem (*pricing problem*), which is the topic of Subsection 4.3.2. Since we do not want to get a very large master problem, columns with high positive reduced costs will be removed. This only happens if there are more columns than a certain minimum number. The deletion

of columns is an important difference with the algorithm for the single-depot case, where it was not necessary to delete columns, since the resulting Lagrangian subproblem could be solved faster and the number of generated columns was lower.

Finally, in Step 4 we compute feasible solutions, which will be discussed in detail in Subsection 4.3.3.

4.3.1 The Master Problem

To solve the master problem, we use the relaxation of model MD-VCSP1, where the equality signs in the constraints (4.4)-(4.8) are first replaced by "greater-than-or-equal" signs, which are subsequently relaxed in a Lagrangian way. That is, we associate non-negative Lagrangian multipliers κ_j^d, λ_i^d, μ_{ij}^d, ν_i^d and ξ_j^d with constraints (4.4), (4.5), (4.6), (4.7) and (4.8), respectively. Then the remaining Lagrangian subproblem can be solved by pricing out the x variables and solving a large *single-depot vehicle scheduling problem* (SDVSP) for the y variables.

Furthermore, we need an additional procedure to update the Lagrangian multipliers after solving the Lagrangian relaxation. This is necessary to assure that all duties in the current master problem have non-negative reduced costs so that these duties will not be generated again in the pricing problem. We refer to Subsection 1.4.3 for an explanation of this procedure.

An alternative and slightly different approach is to relax constraints (4.2) instead of (4.4). Constraints (4.3) are redundant in this approach. Then the remaining Lagrangian subproblem for the y variables corresponds with solving $|D|$ small SDVSP's instead of a large one. We have compared both approaches and concluded that the initial approach gives slightly better computation times to get the same lower bound (viz. the value of the LP lower bound). Therefore, in the remaining of the chapter, we do not consider the alternative again.

4.3.2 The Column Generation Subproblem

For the MD-VCSP, vehicle blocks are not known and a huge number of feasible pieces of work may exist. Therefore, we propose a two phase procedure for the column generation pricing problem: in the first phase, a *piece generation network* is used to generate a set of pieces of work which serves as input for the second phase where duties are generated. Since there is no interaction between the different depots in the column generation subproblem, we can solve them separately for every depot.

In every iteration i we compute an estimate LBT_i of the lower bound for the overall problem. Let LBS_i denote the value of the Lagrangian lower bound in iteration i, then the estimate is computed as

$$LBT_i = LBS_i + \sum_{d \in D} \sum_{k \in K_i^d} \overline{f}_k^d, \tag{4.10}$$

where K_i^d is the set of duties added in iteration i and $\overline{f}_k^d$ is the reduced cost of duty $k \in K^d$, which is defined below.

$$\overline{f}_k^d = f_k^d - \sum_{i \in N(k,d)} \lambda_i^d - \sum_{(i,j) \in A^s(k,d)} \mu_{ij}^d - \sum_{i \in N^t(k,d)} \nu_i^d - \sum_{j \in N^r(k,d)} \xi_j^d, \tag{4.11}$$

where
$N(k,d)$: set of trips in duty k from depot d,
$A^s(k,d)$: set of short arcs A^{sd} in duty k from depot d,
$N^t(k,d)$: set of trips that have a corresponding task in duty k from depot d with the end of this trip as starting location and the depot as ending location,
$N^r(k,d)$: set of trips that have a corresponding task in duty k from depot d with the depot as starting location and the start of this trip as ending location.

LBT_i is a lower bound for the overall problem if all duties with negative reduced cost are added to the master problem. We terminate Step 3 if the relative difference between LBT_i and LBS_i is small or if the maximum computation time is reached.

Generation of Pieces of Work

Recall that we have defined a piece of work as a continuous sequence of trip tasks and dh-tasks corresponding to (a part of) one vehicle block, and that this sequence of tasks is only restricted by its duration. The network for piece generation is an extension of the network G^d for vehicle scheduling (see Section 4.2). Let a *start point* (*end point*) be defined as the relief point corresponding to the start (end) of a vehicle trip. We define the network $G^{d\prime} = (V^{d\prime}, A^{d\prime})$, where nodes correspond to the relief points on each trip that can be assigned to a vehicle from depot d, and the source r^d and the sink t^d represent the depot. Arcs in $A^{d\prime}$ correspond to dh-tasks and trip tasks. Notice that $G^{d\prime}$ is acyclic.

Let b_{mn}^d be the cost associated with each arc $(m,n) \in A^{d\prime}$. Recall from Section 4.3.1 that we associate Lagrangian multipliers λ_i^d, μ_{ij}^d, ν_i^d and ξ_j^d with constraints (4.5), (4.6), (4.7) and (4.8), respectively. The reduced cost is then defined as

$$\bar{b}^d_{mn} = \begin{cases} b^d_{mn} - \lambda^d_i, & \text{for each arc } (m,n) \text{ with } m \text{ the start point and } n \text{ the end point of trip } i, \\ b^d_{mn} - \mu^d_{ij}, & \text{for each arc } (m,n) \text{ with } m \text{ the end point of trip } i \text{ and } n \text{ the start point of trip } j, \\ b^d_{mn} - \xi^d_j, & \text{for each arc } (m,n) \text{ with } m = r^d \text{ and } n \text{ the start point of trip } j, \\ b^d_{mn} - \nu^d_i, & \text{for each arc } (m,n) \text{ with } n = t^d \text{ and } m \text{ the end point of trip } i. \end{cases}$$

Thus, the reduced costs on the arcs are defined such that the reduced cost of a path is equal to the reduced cost of the corresponding piece of work. Each path between two nodes u and v in network $G^{d\prime}$ corresponds to a feasible piece of work if its duration is between the minimum and maximum allowed duration of a piece of work. However, it is not necessary to generate all pieces, since we only have to satisfy the column generation optimality condition, that is, there are no duties left with negative reduced costs. The sufficient subset is generated by solving a shortest path problem between each pair of nodes in network $G^{d\prime}$ that satisfy the constraint on the piece duration. For all feasible paths from u to v, three additional paths are considered, namely $r^d, u, ..., v$; $u, ..., v, t^d$ and $r^d, u, ..., v, t^d$. It is easy to see that by generating only this subset of pieces, we assure that the column generation optimality condition is satisfied.

Generation of Duties

Duties have to satisfy certain feasibility conditions. In particular, they consist of a certain maximum number of pieces. In our case this maximum is equal to 2. This is the reason why we simply enumerate all possible combinations of pieces and check if such a combination is feasible, until we find a specified number of duties with negative reduced costs (or all combinations have been checked). The reduced cost of a duty can be easily computed when the reduced cost of a piece is already known: the reduced cost of a duty is equal to the sum of the reduced costs of the pieces it is built from under the assumption of continuous attendance (as described in Section 4.1).

4.3.3 Feasible Solutions

At the end, in Step 4, we only relax constraints (4.5)-(4.8), which is again done in a Lagrangian way. Therefore, the solution of the remaining subproblem gives a feasible vehicle schedule. Notice, that this subproblem is a MDVSP, which is an NP-hard problem.

However, we need to solve only a few iterations of the subgradient algorithm to get good solutions, since we start with already good multipliers (the best one from the last iteration in Step 1). After that, for every depot, we compute feasible crew schedules given these (feasible) vehicle schedules. We do this by solving the CSP in the same way as in the initial step. Of course, it is also possible to compute more feasible solutions by solving the CSP not only for the vehicle solution from the last iteration, but also for vehicle solutions which were encountered earlier on. A reason to actually do this could be that the gap between the lower and upper bound is quite large, which is an indication that the upper bound could be improved upon.

4.4 Alternative Approach

In this section we propose another mathematical formulation for the MD-VCSP which has only variables related to crew duties. The vehicle schedule can be obtained implicitly from the crew schedule. This formulation can be derived from the one previously presented in this chapter, but is also equivalent to the formulation of Haase *et al.* (2001) in the case of a single depot. An alternative algorithm based on this model is suggested in Subsection 4.4.2.

4.4.1 Mathematical Formulation

The alternative mathematical formulation for the MD-VCSP can be obtained from model (MD-VCSP1) by substituting for the y variables using constraints (4.6), (4.7) and (4.8). That is, we replace $\sum_{\{j:(i,j)\in A^d\}} y_{ij}^d$ by $\sum_{\{j:(i,j)\in A^{d*}\}} \sum_{k\in K^d(i,j)} x_k^d$, where A^{d*} denotes the part of A^d without long arcs (which are not relevant for the duty related variables), in constraints (4.2), (4.4) and (4.5), since

$$\begin{aligned}
\sum_{\{j:(i,j)\in A^d\}} y_{ij}^d &= \sum_{\{j:(i,j)\in A^{sd}\}} y_{ij}^d + \sum_{\{j:(i,j)\in A^{ld}\}} y_{ij}^d + y_{it^d}^d \\
&= \sum_{\{j:(i,j)\in A^{sd}\}} \sum_{k\in K^d(i,j)} x_k^d + \sum_{k\in K^d\left(i,t^d\right)} x_k^d \\
&= \sum_{\{j:(i,j)\in A^{d*}\}} \sum_{k\in K^d(i,j)} x_k^d.
\end{aligned}$$

The problem can then be formulated as follows.

(MD-VCSP2):

$$\min \quad \sum_{d\in D}\sum_{k\in K^d} g_k^d x_k^d \tag{4.12}$$

$$\text{s.t.} \quad \sum_{d\in D}\sum_{\{j:(i,j)\in A^{d*}\}}\sum_{k\in K^d(i,j)} x_k^d = 1 \qquad \forall i \in N, \tag{4.13}$$

$$\sum_{\{i:(i,j)\in A^{d*}\}}\sum_{k\in K^d(i,j)} x_k^d - \sum_{\{i:(j,i)\in A^{d*}\}}\sum_{k\in K^d(j,i)} x_k^d = 0 \qquad \forall d \in D, \forall j \in N^d, \tag{4.14}$$

$$\sum_{k\in K^d(i)} x_k^d - \sum_{\{j:(i,j)\in A^{d*}\}}\sum_{k\in K^d(i,j)} x_k^d = 0 \qquad \forall d \in D, \forall i \in N^d, \tag{4.15}$$

$$x_k^d \in \{0,1\} \quad \forall d \in D, \forall k \in K^d. \tag{4.16}$$

In this formulation g_k^d is the sum of the costs of duty $k \in K^d$ and the variable vehicle costs corresponding to the arcs in this duty. However, in the proposed formulation we cannot deal with fixed vehicle costs. We can only introduce them by adding an extra decision variable B to count the number of vehicles and by adding the following set of constraints:

$$\sum_{d\in D}\sum_{k\in K^d(h)} x_k^d \leq B \quad \forall h \in H, \tag{4.17}$$

where $K^d(h)$ is the set of duties corresponding to depot d where time point h is between the start and end time of one of the tasks of this duty. H is defined as the set of time points at which a vehicle may leave a depot to drive to the start location of a trip, i.e. the start time of the trip minus the driving time from the depot to the start location. It suffices to consider only these time points, since only at these time points the number of vehicles can increase, i.e. a departure may occur. Moreover, if there are two consecutive time points in H between which no arrival at a depot can occur, then the number of vehicles at the latest time point is at least the number of vehicles at the earlier one. This means that the constraint for the earlier time point can be left out.

The restrictions in the above formulation can also be interpreted directly from the problem description. Constraints (4.13) assure that each trip is assigned to exactly one duty from one of the depots. Therefore, each trip is implicitly assigned to exactly one vehicle too. Furthermore, constraints (4.14) and (4.15) guarantee feasibility of the underlying vehicle schedules. That is, they ensure that each trip has a successor and a predecessor assigned to the same depot. This predecessor (successor) is possibly the source (sink), i.e. then it is the first (last) trip of that vehicle.

The main advantage of the formulation above compared to the one in Section 4.2 is that the number of constraints is much less. In the case that $|N^d| = N$ and $|A^{sd}| = A^s, \forall d \in D$, the number of constraints reduces from $(4|D| + 2)|N| + |D||A^s|$ to $(2|D| + 1)|N| + |H|$. However, we do not have the vehicle schedules explicitly in the model anymore, which

means it is less straightforward how to construct feasible solutions by using a Lagrangian heuristic. Therefore, we propose an algorithm that consists of two phases, in which we use both formulations presented in this chapter.

4.4.2 Algorithm

In this subsection we discuss a second algorithm to solve the MD-VCSP which consists of two phases. In the first phase, we compute a lower bound using model MD-VCSP2 by again combining column generation and Lagrangian relaxation. We use the columns generated during the first phase in the second one to find a feasible vehicle schedule and a corresponding crew schedule. The second phase is similar to Step 4 of the previous algorithm described in Section 4.3. The important differences are thus in computing the lower bound, where we use model MD-VCSP2 instead of MD-VCSP1. These differences are described below.

The first difference is in the computation of the lower bound. Before we describe how we calculate the lower bound, we discuss some important observations. If we include only one of the constraints (4.17) at a time, the Lagrangian subproblem can still be solved in polynomial time. In the case we do not include any of these constraints, the subproblem can even be solved by only pricing out the x variables. Furthermore, most of the constraints (4.17) are not binding at all. Therefore, the general idea is to include one of the constraints (4.17) at a time. We start without constraints (4.17), which means we do not consider fixed vehicle costs explicitly. However, we can easily get a lower bound on these costs, since an optimal solution of the MDVSP with only fixed costs, provides a lower bound on the fixed vehicle costs. A lower bound for the total problem is thus given by the sum of the Lagrangian lower bound and the value of the optimal solution of the MDVSP with only fixed costs. This lower bound can be improved by adding one of the constraints (4.17) and by calculating a new lower bound with the same set of multipliers. If we obtain an improvement, then we use the subgradient algorithm to get a better lower bound. Otherwise, we include the next constraint. In this way, we can subsequently improve the lower bound. However, we cannot guarantee that we find the best lower bound.

Another important difference compared to the first algorithm is in the definition of the reduced costs of the arcs in the piece generation network. Let μ_i, υ_j^d and λ_i^d be the Lagrangian multiplier corresponding to constraints (4.13), (4.14) and (4.15) in model MD-VCSP2, respectively. The reduced cost of the arcs are then defined as follows:

$$\bar{b}^d_{mn} = \begin{cases} b^d_{mn} - \lambda^d_i, & \text{for each arc } (m,n) \text{ with } m \text{ the start point and } n \text{ the endpoint of trip } i, \\ b^d_{mn} - \mu_i - \upsilon^d_i + \upsilon^d_j, & \text{for each arc } (m,n) \text{ with } m \text{ the end point of trip } i \text{ and } n \text{ the start point of trip } j, \\ b^d_{mn} + \upsilon^d_j, & \text{for each arc } (m,n) \text{ with } m = r^d \text{ and } n \text{ the start point of trip } j, \\ b^d_{mn} - \mu_i - \upsilon^d_i, & \text{for each arc } (m,n) \text{ with } n = t^d \text{ and } m \text{ the end point of trip } i. \end{cases}$$

However, notice that we can still use the all-pairs shortest path algorithm for the generation of the pieces as described in Subsection 4.3.2.

4.5 Computational Results

In this section we test our algorithms on some real-life data sets from Connexxion and on some random data instances. All tests are executed on a Pentium III 450MHz personal computer (128MB RAM) with the following parameter settings. Notice that all computation times are denoted in seconds.

1. The objective is to minimize the total sum of vehicles and drivers. For solving the MDVSP in the sequential approach and in the initial step for the integrated approach we use an additional fictitious cost in the variable vehicle costs, viz. for every minute a vehicle is empty outside the depot a cost equal to 1 is incurred. This is necessary to make a fair comparison between a sequential and an integrated approach.

2. The pricing problems are solved independently for each depot and each type of duty. Moreover, we generate at most 1500 duties for each combination of a depot and type of duty.

3. The maximum number of iterations in the subgradient algorithm to solve the master problem (Step 1) is $500+3k$ in the k-th iteration of the column generation algorithm. However, for constructing the feasible solutions in Step 4, the number of iterations is only 10, since in that case the subproblem is NP-hard. Such a small number of iterations is sufficient, since we already start with good multipliers, namely the best ones of the last iteration in the previous step. We construct 10 feasible solutions from which the best one will be selected.

4. The column generation algorithm is stopped if the difference between the current and estimated lower bound is smaller than 0.1% or if the computation time of the lower bound phase is more than 3 hours. Notice that in the latter case we do not have a proven lower bound.

In Subsection 4.5.1 we describe some properties of the real-world data instances. The results can be found in Subsection 4.5.2. Furthermore, we propose a new way of generating random data instances to simulate problem instances for an extra-urban bus network in Subsection 4.5.3. We choose for a different way of generating these random instances, since generators defined in the literature before (see e.g. Dell'Amico *et al.* (1993), Freling (1997) and Haase *et al.* (2001)) do not take into account specific properties of an extra-urban bus network. First of all, they choose to take the start times of the trips completely random, which means that the intervals between two trips can vary a lot, e.g. between the first two trips it may be five minutes, while it can be two hours between the second and the third trip. However, in the real world these frequencies are fixed to some extent, e.g. during the peak hours it is 20 minutes and outside the peak hours 30 minutes. Secondly, they have very long travel times such that vehicles can only drive a few trips, while in our real-world instances a vehicle drives about 10 trips on average.

4.5.1 Properties of the Real-World Data Instances

We test our algorithms on some subsets of a large data set from Connexxion, which is the largest bus company in the Netherlands. The total set consists of 1104 trips and 4 depots in the area between Rotterdam, Utrecht and Dordrecht, three large cities in the Netherlands. However, it is important to note that not all trips are allowed to be driven by a vehicle from every depot. In fact, almost half of the trips can only be assigned to one depot and only a very small number can be assigned to all depots. On average, a trip can be assigned to 1.71 depots.

The restrictions that we have taken into account, are as follows. A driver can only be relieved by another driver at the start or end of a trip at certain specified locations or at the depot. There are 8 of these locations, which are all major bus stations. If a driver starts/ends his duty at the depot, there is a sign-on/sign-off time of 10 and 5 minutes, respectively. If a driver starts/ends his duty at another relief location, an extra time of 15 minutes plus the deadhead time between this location and the depot is added to the length of the duty. There are five different types of duties, one tripper type consisting of one piece with a length between 30 minutes and 5 hours, and four normal types consisting of two pieces with the properties described in Table 4.1.

type	1 (early)		2 (day)		3 (late)		4 (split)	
	min	max	min	max	min	max	min	max
start time			8:00		13:15			
end time		16:30		18:14				19:30
piece length	0:30	5:00	0:30	5:00	0:30	5:00	0:30	5:00
break length	0:45		0:45		0:45		1:30	
duty length		9:45		9:45		9:45		12:00
work time		9:00		9:00		9:00		9:00

Table 4.1: Properties of the different duty types

4.5.2 Results on Real-World Data Instances

We consider 8 different problem instances for which the number of trips varies between 194 and 653 trips, which have been derived from the large set described in the previous subsection. In Table 4.2 an overview of the results of the different algorithms is provided for these test problems. For each instance, we give the number of trips and the average number of depots to which a trip may be assigned. Furthermore, we give the number of vehicles, drivers and the total sum of these two for the sequential approach and the two integrated approaches presented earlier in this chapter (Section 4.3 and 4.4, respectively). Finally, we report the best lower bound given by the two algorithms.

	instance	1	2	3	4	5	6	7	8
	trips	194	210	220	237	304	386	451	653
	depots/trip	1.60	2.47	1.52	2.38	2.48	1.27	1.67	1.74
	vehicles	19	33	27	34	40	32	47	67
seq.	drivers	33	56	51	62	74	59	86	123
	total	52	89	78	96	114	91	133	190
	vehicles	19	33	27	34	40	32	47	67
int. 1	drivers	30	50	41	55	65	58	77	117
	total	49	83	68	89	105	90	124	184
	vehicles	19	33	27	34	40	32	47	67
int. 2	drivers	28	50	41	54	67	58	77	117
	total	47	83	68	88	107	90	124	184
	lower	44*	77	64*	81	95*	-	-	-

Table 4.2: Results Connexxion data instances

As can be seen from Table 4.2 both integrated approaches give much better solutions than the sequential one. The number of vehicles does not change, while the number of drivers is reduced significantly. The largest relative improvement is obtained for instance 3, where both algorithms save 10 out of 51 drivers. Furthermore, we can see that it

is difficult to conclude which of the algorithms for the integrated approach is better. Sometimes the first one gives a better solution and sometimes the second one.

We were only able to compute a lower bound for two of these instances given the maximum computation time of 3 hours for the lower bound phase, namely for instances 2 and 4. For instances 1, 3 and 5 the lower bound has been computed without taking a maximum computation time as stop criterion into consideration. For all instances, the best lower bound is obtained by the first algorithm. The computation times to compute these lower bounds vary a lot, e.g. for instance 1 it takes almost 6 hours while for instance 2 it takes only 45 minutes (both for the first algorithm), although instance 2 has more trips and the average number of depots per trip is significantly higher. This difference can be explained by the completely different structure of these instances, namely that the average length of the trips in instance 2 is much higher than in instance 1. This can also be seen from the table, since the number of vehicles and drivers is much lower for instance 1 than for instance 2.

4.5.3 Generation of Random Data Instances

In this subsection, we give a detailed description of the way the random data instances have been generated. These instances are available at the web page

http://www.few.eur.nl/few/people/huisman/instances.htm.

The coordinates of the different locations, either depots or start/end points of the lines, are integers generated from a uniform distribution in a 50 by 50 kilometers square. However, there is a minimum distance between each pair of depots and between each pair of start/end points of 10 kilometers.

We consider two different types of instances, which vary in the travel speed. If the travel speed is lower, trips are longer and therefore, less trips will be assigned to one vehicle as well as to one driver. For each of the two types, we have six cases, three in which we have four lines (from A to B, C and D and from B to C) and again three with five lines (the same lines as in the first case plus a line from C to E). All lines are driven in both directions and have the same frequency. Furthermore, we define four different intervals with respect to frequency and travel speed. In Table 4.3, we denote these frequencies (in the case of 10, 20 and 40 trips per line/direction, respectively) and travel speeds (for type A as well as B) for the different intervals.

The start time of the first trip for each line/direction is uniformly drawn between 06:00 and 07:19, between 06:00 and 06:39 and between 06:00 and 06:19, respectively. The end times are computed as the start time plus the travel time between the locations rounded up to the nearest integer. The travel speed for deadhead trips is 50 kms/hour. We choose this significantly higher than the operational travel times since the bus does not have to stop for passengers entering or leaving, which means the shortest route can be taken.

interval	frequency (min.)			speed (kms/hour)	
	10-trips	20-trips	40-trips	type A	type B
06:00 - 08:59	80	40	20	28	20
09:00 - 12:59	120	60	30	32	24
13:00 - 18:59	80	40	20	30	23
19:00 - 23:59	240	120	60	35	26

Table 4.3: Frequencies and travel speeds per interval

Finally, we have to choose the relief locations, where a driver can take a break and one driver can be replaced by another one. These locations need some relief facilities like a canteen to take a meal break. It is most likely that these facilities are at the start and end points of the lines. Therefore, we choose A, B, C and D as relief locations in the cases with four lines and these locations plus E in the other ones. We use the same restrictions with respect to the feasibility of the duties as before (see Subsection 4.5.1).

Notice that in contrary to the real-world instances we assume that vehicles from each depot can carry out all trips.

4.5.4 Results on Random Data Instances

We have tested our algorithms by generating 10 random instances for the six different cases described in Subsection 4.5.3 (10, 20 and 40 trips per line/direction with 4 and 5 lines). As a consequence, the total number of trips varies from 80 to 400. These tests are executed with 2 as well as 4 depots. However, with 4 depots we do not consider the two largest cases (with 320 and 400 trips, respectively).

In Tables 4.4 and 4.5 we give an overview of the results for instances of type A with 2 and 4 depots, respectively. We give the solution of the traditional sequential approach and of the two integrated approaches described in the Sections 4.3 and 4.4, respectively.

	trips	80	100	160	200	320	400
	vehicles	9.2	11.0	14.8	18.4	26.7	32.9
seq.	drivers	23.8	29.0	35.9	44.5	60.8	74.9
	total	33.0	40.0	50.7	62.9	87.5	107.8
	vehicles	9.2	11.0	14.8	18.4	26.7	32.9
int. 1	drivers	20.6	24.8	33.5	40.7	60.1	73.2
	total	29.8	35.8	48.3	59.1	86.8	106.1
	vehicles	9.2	11.0	14.8	18.4	26.7	32.9
int. 2	drivers	20.6	24.6	33.5	41.0	60.0	74.2
	total	29.8	35.6	48.3	59.4	86.7	107.1

Table 4.4: Average results random data instances - 2 depots - type A

	trips	80	100	160	200
	vehicles	9.2	11.0	14.8	18.4
seq.	drivers	25.8	29.9	38.8	47.1
	total	35.0	40.9	53.6	65.5
	vehicles	9.2	11.0	14.8	18.4
int. 1	drivers	20.5	25.3	34.1	41.6
	total	29.7	36.3	48.9	60.0
	vehicles	9.2	11.0	14.8	18.4
int. 2	drivers	20.4	25.2	34.7	42.0
	total	29.6	36.2	49.5	60.4

Table 4.5: Average results random data instances - 4 depots - type A

As can be seen from Tables 4.4 and 4.5, the total sum of vehicles and drivers can be reduced significantly by using an integrated approach. Furthermore, the difference between the two algorithms is quite small on average for problem instances up to and including 320 trips. Only in the case of 400 trips (2 depots) and 160 and 200 trips (4 depots), the first algorithm performs significantly better than the second one. Since the maximum computation time for the lower bound phase is fixed, we conclude that only the first algorithm finds good feasible solutions after a few column generation iterations. Finally, notice that the number of drivers in most cases with 4 depots is more than in the corresponding case with 2 depots. This is conspicuous since all data are the same except the fact that there are two extra depots, which means that the solution in the 2 depot case is also a feasible one in the 4 depot case. Therefore, we can conclude that our algorithms perform worse if there are more depots.

In Tables 4.6 and 4.7 we present detailed results for both algorithms for 2 and 4 depots, respectively. In the upper part of the tables we give some statistics with respect to the first algorithm. We denote the average number of iterations of the column generation algorithm and the average computation times for the master problem (cpu m.) and the pricing problem (cpu p.), respectively. Furthermore, we give the total average computation time for computing the lower bound (cpu t.) and the average computation time for solving the last subgradient algorithm in Step 4 (cpu f.). These averages are computed over the instances for which a lower bound is found. Therefore, we also denote the number of instances (out of 10) for which we actually found a lower bound. In the second part of the tables we give the same statistics for the second algorithm. Although, we use a crew scheduling algorithm several times to compute feasible solutions in the final step of the algorithm, we do not mention these cpu times here, since any crew scheduling algorithm can be used and it is not necessary to use the one we suggested.

In the third part, we compare the upper bounds (best feasible solutions) of both algorithms with the best lower bound of the two algorithms, which results in “gap 1”

and "gap 2". Notice that sometimes the first algorithm gives the better lower bound and sometimes the second one. Finally, we denote in the bottom part the number of instances (out of 10) that the first algorithm gives a better lower and upper bound, respectively. Between brackets, we also indicate the number of instances, where these bounds for the two algorithms are equal. By definition, the second algorithm gives the best bound in the remaining cases.

	# trips	80	100	160	200	320	400
	# iter.	17.4	25.2	36.8	39.5	-	-
	cpu m.	154.7	403.9	982.8	1,641.5	-	-
int. 1	cpu p.	148.7	510.7	3,529.8	4,769.5	-	-
	cpu t.	317.5	942.3	4,721.3	6,675.0	-	-
	cpu f.	3.6	5.0	15.3	40.0	-	-
	# found	10	10	4	2	0	0
	# iter.	24.2	22.8	55.5	67.5	-	-
	cpu m.	212.7	224.0	832.3	1,251.0	-	-
int. 2	cpu p.	133.7	201.8	4,159.5	6,128.0	-	-
	cpu t.	354.0	439.3	5,140.8	7,562.5	-	-
	cpu f.	13.3	20.3	42.0	237.0	-	-
	# found	10	9	4	2	0	0
	upper 1	29.8	35.8	52.2	69.0	-	-
	upper 2	29.8	35.6	52.4	68.5	-	-
	best lower	28.2	33.9	49.2	64.5	-	-
	gap 1 (%)	5.37	5.31	5.75	6.52	-	-
	gap 2 (%)	5.37	4.78	6.11	5.84	-	-
	# lower 1	0 (10)	2 (8)	3 (6)	1 (9)	0 (10)	0 (10)
	# upper 1	1 (8)	1 (6)	3 (4)	4 (3)	3 (4)	5 (4)

Table 4.6: Detailed results random data instances - 2 depots - type A

From Tables 4.6 and 4.7, we can conclude that only for small instances a lower bound is computed within 3 hours computation time. Furthermore, it is more difficult to find a lower bound in the case of 4 depots. The first algorithm finds the lower bounds more often than the second one. In all cases except two, namely 160 trips for 2 as well as 4 depots, the first algorithm gives the best lower bound of the two or both algorithms give the same lower bound.

If we look at the quality of the solutions found, the first algorithm sometimes performs better, while other times the second one performs better. For instance, in the case of 160 trips and 2 depots the first and the second algorithm give both three times the best solution. However, in the cases with a large number of trips the first algorithm performs significantly better, e.g. in the case of 400 trips and 2 depots the first algorithm gives 5

	# trips	80	100	160	200
	# iter.	27.8	34.0	37.3	-
	cpu m.	316.9	363.2	1,278.3	-
int. 1	cpu p.	532.3	853.4	7,063.0	-
	cpu t.	875.7	1,272.4	8,574.7	-
	cpu f.	30.4	172.2	341.7	-
	# found	10	9	2	0
	# iter.	28.6	34.0	52.0	-
	cpu m.	366.4	379.3	1,077.0	-
int. 2	cpu p.	285.9	566.4	6,276.0	-
	cpu t.	667.0	973.3	7,562.5	-
	cpu f.	120.9	392.3	1,021.5	-
	# found	7	8	2	0
	upper 1	29.7	37.0	56.7	-
	upper 2	29.6	36.8	57.7	-
	best lower	27.8	34.0	52.7	-
	gap 1 (%)	6.40	8.11	7.06	-
	gap 2 (%)	6.08	7.55	8.67	-
	# lower 1	3 (7)	2 (8)	2 (7)	0 (10)
	# upper 1	1 (7)	4 (2)	5 (5)	4 (4)

Table 4.7: Detailed results random data instances - 4 depots - type A

out of 10 times the best solution, while the second algorithm gives only once the best solution of the two.

The average gaps between the feasible solutions and the best known lower bound varies between 4 and 9% for the first as well as the second algorithm. These gaps are slightly higher in the cases with 4 depots than in the corresponding case with 2 depots, which confirms our earlier suggestion that the algorithms perform better in the 2 depot case. Furthermore, these gaps do not vary significantly between the two algorithms. Although, the savings of using an integrated approach are quite high, the gaps suggest that there may be some room for further improvement. However, in our opinion the main reason of these gaps is the fact that the lower bounds are not very strong.

For instances of type B we obtain similar results (see Tables 4.8, 4.9, 4.10 and 4.11). The main conclusions discussed above about the effectiveness of using an integrated approach and the differences between the two algorithms still hold. The difference between the two types is that the instances of type B are easier to solve than those of type A, since the number of instances for which we found a lower bound is higher, the computation times are lower and the gaps are smaller. Notice that this is also what we expected beforehand.

	trips	80	100	160	200	320	400
	vehicles	11.3	13.8	19.3	24.4	35.8	44.2
seq.	drivers	26.9	32.9	44.4	54.7	79.0	96.8
	total	38.2	46.7	63.7	79.1	114.8	141.0
	vehicles	11.3	13.8	19.3	24.4	35.8	44.2
int. 1	drivers	24.9	29.1	42.3	51.4	77.8	95.0
	total	36.2	42.9	61.6	75.8	113.6	139.2
	vehicles	11.3	13.8	19.3	24.4	35.8	44.2
int. 2	drivers	24.7	29.1	42.6	52.2	78.0	95.6
	total	36.0	42.9	61.9	76.6	113.8	139.8

Table 4.8: Average results random data instances - 2 depots - type B

	trips	80	100	160	200
	vehicles	11.3	13.8	19.3	24.4
seq.	drivers	28.3	34.1	45.9	56.8
	total	39.6	47.9	65.2	81.2
	vehicles	11.3	13.8	19.3	24.4
int. 1	drivers	25.1	30.3	42.9	52.1
	total	36.4	44.1	62.2	76.5
	vehicles	11.3	13.8	19.3	24.4
int. 2	drivers	24.8	30.0	44.0	53.6
	total	36.1	43.8	63.3	78.0

Table 4.9: Average results random data instances - 4 depots - type B

	# trips	80	100	160	200	320	400
	# iter.	15.9	194	40.0	42.3	-	-
	cpu m.	136.2	243.8	1,196.4	1,753.0	-	-
int. 1	cpu p.	121.0	266.5	4,106.3	4,489.0	-	-
	cpu t.	267.6	530.9	5,487.3	6,321.0	-	-
	cpu f.	4.1	4.3	14.7	20.3	-	-
	# found	10	10	7	3	0	0
	# iter.	19.9	19.1	55.9	55.5	-	-
	cpu m.	162.2	165.1	1,044.1	779.5	-	-
int. 2	cpu p.	90.8	116.3	3,641.5	1,962.5	-	-
	cpu t.	258.8	290.8	4,823.3	2,809.5	-	-
	cpu f.	12.2	16.6	54.8	86.5	-	-
	# found	10	9	8	2	0	0
	upper 1	36.2	42.9	64.3	86.7	-	-
	upper 2	36.0	42.9	64.6	87.7	-	-
	best lower	34.1	40.9	60.6	82.3	-	-
	gap 1 (%)	5.80	4.66	5.61	5.00	-	-
	gap 2 (%)	5.28	4.66	6.24	6.08	-	-
	# lower 1	0 (10)	1 (8)	3 (6)	2 (8)	0 (10)	0 (10)
	# upper 1	2 (4)	2 (6)	3 (5)	7 (2)	3 (4)	6 (3)

Table 4.10: Detailed results random data instances - 2 depots - type B

	# trips	80	100	160	200
	# iter.	24.3	27.6	43.0	-
	cpu m.	287.9	606.1	1,548.0	-
int. 1	cpu p.	403.9	1,103.7	6,538.8	-
	cpu t.	709.9	1,753.4	8,236.5	-
	cpu f.	36.5	84.2	310.3	-
	# found	10	10	3	0
	# iter.	22.1	43.6	40.0	39.0
	cpu m.	240.1	835.7	1,034.8	970.0
int. 2	cpu p.	171.4	534.8	3,557.3	4,376.0
	cpu t.	421.6	1,388.1	4,750.8	5,583.0
	cpu f.	62.2	217.9	1,673.5	3,622.0
	# found	9	9	4	1
	upper 1	36.4	44.1	71.5	102.0
	upper 2	36.1	43.8	73.3	103.0
	best lower	33.9	40.6	66.5	92.0
	gap 1 (%)	6.87	7.94	6.99	9.80
	gap 2 (%)	6.09	7.31	9.22	10.68
	# lower 1	2 (7)	2 (8)	3 (6)	0 (9)
	# upper 1	2 (4)	3 (4)	8 (1)	9 (0)

Table 4.11: Detailed results random data instances - 4 depots - type B

4.5.5 Discussion and Conclusion

Before we state our conclusions, we first would like to make several remarks on the limitations of the proposed models and algorithms, since they are only valid under certain assumptions. Especially, the second assumption that all crew have their own depot and are only allowed to perform tasks on vehicles from their own depot, is a crucial one. If this assumption is not valid, the crew scheduling problem cannot be solved independently for each depot anymore. Moreover, in the integrated models and algorithms the constraints linking the vehicles and the crews for the different depots will not be independent anymore. Such a situation can occur in practice, where one can think of a situation where the driver first has a few tasks on a vehicle from his own depot and then on a vehicle from another depot, where he will be relieved at the end of his duty on a location close to his own depot. However, it is unlikely that such a situation occurs frequently, since not all trips are allowed to be assigned to each depot and if it is allowed, it is often possible to change vehicles between the depots. Furthermore, drivers often have only knowledge about a part of the bus network and a particular type of vehicle (different vehicle types corresponds to different depots in the model).

The results reported in the previous section indicates that medium-sized problem instances with multiple depots can be solved by using an integrated approach for the vehicle and crew scheduling problem. Furthermore, there are significant savings compared to the traditional sequential approach, where first the vehicle scheduling and afterwards the crew scheduling problem is solved. However, it should be mentioned that these comparisons do not say anything about the performance of the actual planners, since they use neither a sequential approach nor an integrated one.

We have suggested two different algorithms which are both Lagrangian heuristics based on column generation. The major difference between these two algorithms is that in the second one the computation of the lower bound is based on a model with only variables related to the crew schedules. However, the lower bounds obtained by this algorithm are rarely stronger than the bounds obtained by the first one and regularly weaker. If the solutions of both algorithms are compared, it is difficult to conclude which algorithm is better, since sometimes the first one gives the best solution and sometimes the second one. However, for the larger random problem instances of both types, the first algorithm performs better.

Finally, we have introduced a new and better way to generate random problem instances to simulate an extra-urban bus network. We hope that also other researchers will use these instances in future experiments.

4.6 Application: Solving Large Real-World Instances

In the literature on vehicle and crew scheduling, there has not been much attention to the problem of splitting up large instances into several smaller ones such that a good overall solution is obtained. Algorithms are developed to solve a certain problem, either optimally or heuristically, and they are tested on self made problem instances, or on (small) instances from practice which the algorithm can still solve. If a real-world instance has to be solved and it seems to be too large for the algorithm to solve it, the problem is just split up into several smaller instances. The algorithm is used to solve those smaller instances and the results are combined such that there is an overall solution. This solution is then feasible, but of course, even if the algorithm itself provides an optimal solution, optimality of the overall problem is likely to be lost. Moreover, in practice, exotic constraints can exist such that even feasibility cannot be guaranteed. The way the instance has been divided up, is almost never an issue in the literature. However, different divisions can result in completely different final outcomes; one splitting can result in a much better solution than another one. Therefore, the instances are mostly divided according to some logical rules.

For example, in the field of crew scheduling, Fores *et al.* (2001) describe this problem. In 1998, they subdivided a large instance of ScotRail into two smaller instances according to a geographic division. Since this resulted in some strange outcomes, several tasks were exchanged between the different divisions. After several days of trial and error, they found a reasonable splitting of the instance such that the optimal solutions of both smaller instances seemed to give a reasonable overall solution. In 2000, they were able to solve the large instance optimally. They checked the performance of the splitting and indeed the optimal solution of the complete instance was the same as the solution, which they obtained by splitting up the instance several years before.

Haghani *et al.* (2003) describe a comparative analysis of different vehicle scheduling problems with route time constraints. This can be seen as a special case of the integrated vehicle and crew scheduling problem, namely where a duty exactly coincides with a vehicle and the only constraint is a maximum duty length. They compared several approaches on a large real-world instance in Baltimore which consists of multiple depots. Since they could not solve this problem exactly, they considered three approaches. The first approach used CPLEX to solve a reduced problem instance, i.e. several variables in the large IP were just omitted. In the second and third approach, they solved several smaller, single-depot instances with an exact algorithm. The difference between both approaches is the way in which the problem is split up. One is based on the current solution of the public transport company, the other on the outcome of the first approach. They showed that this last approach outperformed the first one.

For the integrated vehicle and crew scheduling problem only small and medium-sized instances have been solved. In fact, the instances in Section 4.5 are among the largest

ones that have ever been solved in the case of multiple depots. Furthermore, we showed that a significant improvement can be obtained by applying such an integrated approach. Therefore, we try to answer the following questions in this section.

1. How can large instances be split up into several smaller ones such that applying an integrated approach on those instances can be done in a reasonable computation time?

2. Does such a splitting outperform the sequential approach when this is used to solve the large instance at once?

3. Does it outperform the integrated approach when this is terminated after a certain computation time?

Furthermore, we compare different ways of splitting the problem and we give some results on several real-world instances from Connexxion, which have been described in Subsection 4.5.1. Finally, we use these ideas to find a solution for the large problem of 1104 trips which we could not solve before.

4.6.1 Different Ways of Splitting

In this subsection we describe several approaches of splitting a large instance of the MD-VCSP into several smaller ones. The different approaches can be divided into two categories:

1. splitting the problem into several SD-VCSP's, i.e. assign each trip to a depot;

2. splitting an instance into a predetermined number of smaller ones.

We will start the discussion with the first category. The most simple way is a random assignment of the trips to the depots. Although, this is not interesting in itself, a more sophisticated rule should always beat this trivial one. The more interesting assignments of trips to depots are the following:

- assign each trip to the depot closest to its start location;
- assign each trip to the depot closest to its end location;
- assign each trip to the depot closest to a combination of its start and end location;
- solve the MDVSP and assign each trip to the depot where it is assigned to in the MDVSP.

The first three rules are based on the geographical structure of the problem and can be based on distances or travel times. However, the last rule requires solving of another, much simpler, optimization problem, namely the multiple-depot vehicle scheduling problem, and uses that solution. Notice hereby that even the MDVSP is a NP-hard problem. Moreover, recall that the solution approach on the MD-VCSP starts with solving the MDVSP to obtain an initial feasible solution. Therefore, the extra effort is very low. Of course, it is possible to recombine certain smaller SD-VCSP's again to large MD-VCSP's. This is especially attractive if certain subproblems are so small that recombining does not result in a too large problem again. Another possibility is to use this assignment only as a splitting of the instance and to consider more depots again during the optimization.

The second category is dividing the trips instead of the depot(s) into several small subproblems. We assume here that we have given a maximum number of trips per subproblem. This leads to a certain minimum number of subproblems. Below, we give an overview of such divisions.

- Assign each trip arbitrary to a subproblem such that the maximum number of trips in a subproblem is not exceeded.
- Solve the MDVSP and assign all trips executed by the same vehicle block to the same subproblem. However, the vehicle blocks themselves are assigned arbitrary to a subproblem.
- Solve the MDVSP and assign all trips executed by the same vehicle block to the same subproblem. Moreover, assign the vehicle blocks in consecutive order to the subproblems.
- Solve the MDVSP and assign all trips executed by the same vehicle block to the same subproblem. Moreover, assign the vehicle blocks with the highest correlation to the same subproblem.

The first three ways of dividing speak for themselves. The fourth one needs some further explanation. We calculate the correlation w_{ij} between two vehicle blocks with the algorithm suggested in Figure 4.2.

$w_{ij} := 0$.
For each different line number l in vehicle block i:
    $\delta_i :=$ number of trips in block i with line number l;
    $\delta_j :=$ number of trips in block j with line number l;
    if $\delta_j > 0$, then $w_{ij} := w_{ij} + \delta_i + \delta_j - 1$;
    otherwise, $w_{ij} := w_{ij}$.

Figure 4.2: Algorithm to compute w_{ij}

It can be easily seen that the weight is only positive if the two vehicle blocks have both at least one trip from the same bus line.

We define a weighted graph $G = (V, E)$ with V as the set of nodes, where a node corresponds to a vehicle block and E as the set of edges. There is an edge (i, j) between each pair of nodes with its weight equal to w_{ij}. The assignment of the vehicle blocks to different subproblems corresponds now to the partitioning of the graph in certain subgraphs such that the total weight of the cuts is minimal and the different parts have an (almost) equal size, where the size of a part is defined as the sum of the number of trips executed by each vehicle block in that part. For more details about different methods which can be used to partition such a graph, we refer to De Groot (2003).

After the problem has been divided into several subproblems and they have been solved with an integrated approach, we can still recombine some parts of the problem such that the solution can be improved. Since the last step of the algorithm consists of solving a CSP for a certain vehicle schedule, we can recombine all vehicle schedules for each depot and solve one large CSP. Notice that this is possible, since the bottleneck of solving an integrated approach is not the CSP. We will see in the next subsection that this recombining significantly improves the solutions.

4.6.2 Results

In this subsection we first discuss the results of the suggested methods of assigning a trip to a depot. Afterwards, we discuss the splitting of single-depot problems. All tests in this subsection are executed on a Pentium IV 1.8GHz personal computer (512MB RAM). Notice that this pc is much faster than the one used in the previous section. Furthermore, another version of CPLEX is used (6.5 instead of 7.1). Therefore, the results of the sequential as well as the integrated approach can be slightly different than before. Moreover, as stopping criterium for the lower bound phase of the integrated approach a maximum computation time of 4 hours is used. If the problem is divided, this time is maximal 2 hours per subproblem.

Assigning Trips to Depots

In the previous subsection we suggested four different methods to assign a trip to a depot. These approaches have been tested to split real-world instance 2 (see Subsection 4.5.1), containing 4 depots, into two 2-depot instances. Notice that this can be done in three different ways. Table 4.12 provides the results of these divisions where the trips are assigned to a depot at random (average results over three runs), or using one of the four methods, i.e. closest to the start location, closest to the end location, closest to a combination of start and end location or according to the solution of the MDVSP. Notice that, for example, 12-34 means that depots 1 and 2 are in one subdivision, while 3 and 4

are in the other one.

	12-34	13-24	14-23
random	91.7	101.7	95.3
start	89	110	102
end	91	101	97
start-end	88	99	92
MDVSP	84	87	86

Table 4.12: Results splitting depots - instance 2

From this table we can immediately conclude that dividing based on the MDVSP is much better than on one of the geographical rules. Some of those do not even outperform a random assignment. We refer to De Groot (2003) for similar results on other instances. Therefore, we will only consider these types of divisions of the depots in the remainder of this section.

Splitting of the Trips

The different methods for the second category introduced in the previous subsection, have been tested on the eight real-world problem instances discussed in Subsection 4.5.1. We refer to De Groot (2003) for a detailed overview of the results of these tests. Here, we only provide an overview of those methods that performed well. These are the following methods.

- Solve the MDVSP and assign each trip to the depot where it is assigned to in the MDVSP. Afterwards divide the trips into two sets: one set with the trips assigned to the largest depot, i.e. the one with most trips assigned to it and the other sets with the remainder of the trips. Divide those sets again into sets of at most 200 trips such that the trips executed by the same vehicle block (resulting from the earlier solved MDVSP) should be in the same subproblem and the vehicle blocks are assigned to the different subproblems in consecutive order. (**method A**).
- Same as Method A. However, the vehicle blocks are now divided such that the ones with high correlation are as much as possible in the same subproblem (**method B**).
- Same as Method A. However, the depots are not split first (**method C**).
- Same as Method B. However, the depots are not split first (**method D**).

Before we continue our discussion on methods of the second category, we first look at the effect of recombining the different crew scheduling problems per depot at the end.

instance	1	2	3	4	5	6	7	8
with	49	86	70	89	105	91	122	182
without	49	87	71	91	108	91	126	188

Table 4.13: Sum of vehicles and drivers with and without recombining CSP's - method C

Since the effect on the computation time of this step can be neglected, we only compare the solution values. In Table 4.13, we provide this comparison for method C .

As can be seen from the table the saving of recombining can be quite large (up to 6 drivers). Therefore, we recommend to use this option always and thus we take this option also into account for methods A, B and D as well.

In Table 4.14, we report the total number of vehicles and drivers (denoted as V+D), the maximum computation time for one subproblem (cpu max.) and the total computation time (cpu tot.), both in seconds for the methods A to D. Furthermore, we provide similar statistics for the sequential as well as the integrated approach with a maximum computation time (see also Subsection 4.5.2). Notice that since these algorithm do not use splitting, we only mention one computation time.

	instance	1	2	3	4	5	6	7	8
	trips	194	210	220	237	304	386	451	653
	depots/trip	1.60	2.47	1.52	2.38	2.48	1.27	1.67	1.74
seq.	V+D	54	89	76	96	115	93	133	192
	cpu	63	29	21	28	33	144	146	168
int.	V+D	48	85	67	89	106	91	122	184
	cpu	9,304	1,945	5,610	2,616	14,630	15,616	15,265	16,527
	V+D	50	84	70	91	106	91	122	184
A	cpu max.	992	432	327	410	1,172	4,322	2,643	1,800
	cpu tot.	1,656	479	637	483	1,513	6,283	5,048	4,478
	V+D	50	84	70	91	106	90	124	184
B	cpu max.	992	432	327	410	1,172	4,717	2,845	2,182
	cpu tot.	1,656	479	637	483	1,513	7,430	5,320	5,602
	V+D	48	86	70	89	105	91	122	182
C	cpu max.	9,304	151	567	114	1,645	3,557	2,035	1,344
	cpu tot.	9,304	245	728	254	1,904	6,311	5,432	3,298
	V+D	48	86	70	90	106	90	121	181
D	cpu max.	9,304	150	436	175	1,942	7,627	2,525	2,434
	cpu tot.	9,304	245	619	359	2,338	8,950	4,913	4,659

Table 4.14: Results splitting on the Connexxion data instances

If we look at the results we need to make a distinction between instance 1, instances 2-4, instances 5 and 6, and instances 7 and 8. For instance 1, methods C and D provide the same results as the standard integrated approach, since there is no splitting at all. Furthermore, methods A and B are the same. That is, the problem is divided into two subproblems, which reduces the computation time significantly but needs two drivers more. For the instances 2-4 the methods A and B are the same. Here, we can see that the solutions are mostly slightly worse if we split the problems. However, the computation times reduce significantly. The instances 5 and 6 could not be solved integrated without introducing a maximum computation time. Here, we already see an important benefit of the splitting idea, since the solutions of some of the methods are better, while the others are equal. Moreover, the computation times are reduced dramatically. For the instances 7 and 8, we can even see that most of the splitting methods provide better results. Moreover, the computation times become reasonable small. If we would run the subproblems on parallel machines, the computation time would be less than one hour on each machine. For all instances, we can see that splitting the problem leads to much better results than the, fast and simple, sequential approach. If we compare the different methods with each other, we can conclude that method D provides the best results.

Without splitting we were not able to solve the largest problem instance of 1104 trips with an integrated approach. However, it is possible with the different splitting methods. Since method D performed as best one, we use this method here. The results are shown in Table 4.15.

method	seq.	D
vehicles	109	109
drivers	185	176
total	294	285
cpu max.	66	4,895
cpu tot.	66	19,655

Table 4.15: Results of splitting on the largest problem instance

From this table we can see that we can solve this instance in a reasonable time, since we can solve each subproblem within one hour and a half, while we save 9 drivers compared with the sequential approach. Of course, since we cannot give a lower bound on the optimal solution, it is impossible to say anything about the quality of the solution in absolute terms.

4.6.3 Conclusions

In this section we discussed several methods to split large problem instances of the integrated vehicle and crew scheduling problem into several smaller instances. Since we

first applied these approaches to small instances, where we were able to calculate lower bounds on the optimal solutions and a feasible solution with the integrated approach on the complete instance, we showed that on these instances the effect of dividing the instances did not deteriorate the quality of the solutions a lot. Later on, we applied these ideas on large instances and showed that those problems could be solved now, which was not possible before. Furthermore, we showed that the saving compared with the simple, sequential approach is large. Finally, we recommend the use of such splitting methods to solve practical instances instead of dividing it in a 'logical' way.

4.7 Summary

In this chapter we extended the models and algorithms from the previous chapter to solve the integrated vehicle and crew scheduling problem with multiple depots. Furthermore, we discussed a new way of generating random data instances for this problem. We showed the performance of our approaches on these instances and on real-world instances. Since we were not able to solve all instances, we came up with the idea to split instances into smaller ones. We discussed several methods in this chapter how to do this.

Finally, we like to remark that we are happy to see that, based on our success to tackle the integrated vehicle and crew scheduling problems, several other research groups in the world look at this problem. Some of them (see e.g. Borndörfer *et al.* (2002)) even try to implement parts of our ideas (with some modifications) in planning systems to solve real-world problems.

Chapter 5

Dynamic Vehicle and Crew Scheduling

In this chapter we consider a new way of looking at the vehicle and crew scheduling problem. Traditionally, the four operational planning problems as described in Chapter 1 are solved once for every timetable period, but in reality travel times are not fixed which means that the vehicle and crew schedules cannot be executed exactly. This may result in trips starting late.

In recent years, it has become much more important for public transport companies to provide an adequate service level to their customers. This is due to privatization and the growing competition in the public transport market. For instance, in the Netherlands, public transport companies (will) sign a contract with the government to provide transport in a certain area that is only valid for a limited period. The contract specifies minimum service levels. In case these are not met, a penalty is due and the contract may not be renewed. For example, this can be the case if there are too many delays. So it is very important for public transport companies to build robust schedules that limit the number of possible delays.

Connexxion, the largest bus company in the Netherlands, provides services for suburban and interregional transport, especially in highly populated areas with a lot of traffic jams. The company experiences a significant number of trips starting late. Therefore, it is studying the possibility of using a dynamic planning process. This means that if we consider the planning process as described in Section 1.1, the timetabling problem remains the same, but the other planning problems will be solved in a dynamic environment. This has motivated us to develop algorithms for dynamic vehicle and crew scheduling. Furthermore, we answer the question whether the delays will be reduced or not and if yes, at which costs. For this purpose, we will compare the dynamic approach with the static one, where fixed *buffer times* are introduced. That is between every pair of trips assigned to the same vehicle (driver), there needs to be a certain minimum amount of time. Notice that if such a dynamic approach would be used in practice, it would also have some effects

on the rostering problem. These effects fall outside the scope of this thesis.

Of course, delays are not only an important issue in bus transport. In the airline world, a related problem, disruption management, is one of the major issues nowadays (see Horner (2002) about the success of Operations Research after September 11). In the disruption management (or recovery) problem, all aircrafts and crew need to be recovered to their actual schedule after a disruption. An interesting reference to this problem is Stojković & Soumis (2001). The authors propose a method to recover the aircraft routing and crew schedules simultaneously. Their method is based on a Dantzig-Wolfe decomposition combined with a branch-and-bound method. Another related idea is to make more robust crew schedules where the costs of disruptions are taken into account. Schaefer *et al.* (2001) discuss some algorithms to take the expected crew costs into account during the optimization and they show that their approach outperformed the traditional one.

To the best of our knowledge, there is no literature about the dynamic vehicle (and crew) scheduling problem. Therefore, we only discuss here briefly some literature on solving other optimization problems using dynamic models and solution approaches for these models. For a general survey about dynamic and stochastic models, we refer to Powell *et al.* (1995), who explain why it may be useful to use dynamic models instead of static ones for many problems in the field of transportation and logistics. Furthermore, these authors discuss a lot of different methods to deal with uncertainty.

Powell *et al.* (2000) explain that because of randomness in travel times, optimal solutions of a sequence of deterministic crew scheduling and vehicle routing problems, may in reality lead to overall non-optimality. They argue that it is better to use algorithms that are more local in nature, e.g. greedy heuristics.

The literature about dynamic optimization problems can be divided in two parts. One part deals with online approaches as introduced by Sleator & Tarjan (1985). This is especially used in the field of machine scheduling problems. In the last decade, also in the vehicle routing area this type of approaches got more attention. For instance, Ichoua *et al.* (2000) discuss a tabu search heuristic for real-time vehicle dispatching. In the field of passenger transport, this type of approach is used in dial-and-ride problems, see e.g. Madsen *et al.* (1995).

The other part deals with explicit stochastic models. Most often, stochastic programming is used to tackle the uncertainty (see e.g. Laporte & Louveaux (1998) for an application in the area of vehicle routing). Recently, Yen & Birge (2000) have used stochastic programming to solve airline crew scheduling problems to get more robust schedules.

Online approaches can also be used to solve dynamic vehicle (and crew) scheduling problems, but we have a lot of information about the future, which we may use in the approach. The start time, start location and end location of all trips are known. We even know the end times of the trips to some extent. Therefore, our solution approach is more

based on the ideas of stochastic programming than on the ones of online algorithms.

Compared with the previous chapters we will use slightly different abbreviations, instead of VSP for the static vehicle scheduling problem we will use S-VSP, while D-VSP is used for the dynamic vehicle scheduling problem. In this way there is a clear distinction between the static and dynamic case. A similar remark holds for SDVSP, MDVSP, CSP, VCSP and MD-VCSP. Moreover, notice that some definitions and notation will be recalled, to enable the reader to follow this chapter without having read the previous ones.

The chapter is organized as follows. Section 5.1 discusses the potential benefit of using dynamic scheduling. In Section 5.2, based on Huisman *et al.* (2001), we discuss dynamic vehicle scheduling, i.e. we do not consider the crews and focus only on vehicles. Furthermore, we present results of computations with real life data from Connexxion and state our conclusions. Based on (the success of) those ideas we extend the approach to dynamic vehicle and crew scheduling in Section 5.3. To be more precise, we include the ideas on dynamic vehicle scheduling within the models and algorithms on (integrated) vehicle and crew scheduling in the previous chapters of this thesis. In Section 5.4, we apply some ideas to the application which has been discussed in Section 4.6. Finally, we conclude this chapter with some general conclusions on dynamic scheduling of buses and drivers in Section 5.5.

5.1 Potential Benefit of Dynamic Scheduling

Traditionally, the different planning problems (vehicle scheduling/crew scheduling) are solved a few months before the new timetable starts, and it will not be changed for the whole period that the timetable is valid. The disadvantage of this approach is that when the schedules are executed and if there is a delay at a certain moment, the trip following a delayed trip may start late. Of course, one may try to guard against this problem by adding a fixed buffer time to the travel times, but then this buffer will also be present on days that it is not needed, which may cause inefficiencies. The following simple example shows that it may not be obvious how we should choose the buffer times.

Example 5.1 *We have four trips, 1, 2, 3 and 4, and three locations, A, B and C. Table 5.1 gives the start time, end time, start location and end location of these trips. We minimize the number of vehicles, while we have only one depot.*

There are two static optimal solutions. Suppose we choose the one that the trips 1 and 3 are assigned to vehicle 1 and vehicle 2 does trips 2 and 4.

Furthermore, suppose that trip 1 has a delay at arrival of 10 minutes, which means that it arrives at 10:10. Then trip 3 would start 5 minutes late if we use the schedule above. Even if we had added a buffer time of 5 minutes (or less) to every trip, the static optimal solution would not change, which means that trip 3 would still start late. If we

trip	start time	end time	start location	end location
1	9:00	10:00	B	A
2	9:15	10:00	C	A
3	10:05	11:05	A	B
4	10:15	11:00	A	C

Table 5.1: Data of the trips

had introduced a buffer time of 6 minutes (or more), we would need three vehicles for completing these trips. Also note that there is another optimal solution for this example, where 2 vehicles are used and there are no trips starting late: vehicle 1 does the trips 1 and 4 and vehicle 2 trips 2 and 3.

Notice that in the previous example we only focussed on the vehicles and not on the drivers. However, assuming that both vehicles need one driver, it is obvious that the example does not change and the remarks made still hold if also the drivers were taken into account.

If the VSP and CSP (either sequentially or integrated) are solved dynamically, which means that we reschedule a few times per day, i.e. we reassign vehicles and crews to trips, we may be able to prevent the delays at the start of a trip in many cases. In the above example we could reschedule at the moment we know trip 1 has a delay and trip 2 arrives on time such that trip 3 can start at exactly the right time. However, there are also some disadvantages of such an approach. For example, the schedules are only known just before they should be executed, which asks a different way of working and thinking from the drivers, planners and managers of the company. Furthermore, a fast and reliable communication and information system is needed between the vehicles/drivers and the planners. Finally, the underlying optimization problems should be solved within a couple of minutes.

5.2 Dynamic Vehicle Scheduling

In this section we only focus on the vehicles, i.e. we assume there are no crews involved. There are three measures which we consider throughout this section, namely the number of vehicles used, the percentage of trips starting late and a "virtual" measure for delay costs. For the models we will specify objective functions containing these different measures.

We consider the following solution approach to the dynamic vehicle scheduling problem: at certain moments in time, we construct a schedule for the next l time units, where we take into account those decisions already made earlier that cannot be changed anymore, and we also take into account in some way the future after the next l time units. In Figure 5.1, we show the same example as in Figure 2.1 (see Section 2.2), where T is

a point in time at which we construct a schedule for the period $[T, T + l)$. Note that we have drawn two nodes for every trip, namely one corresponding to the start and the end time of the trip. We have already constructed the vehicle schedule up to time point T, so for that part of the schedule we only draw the arcs corresponding to the choices made. The decisions we have to make for the current period are which vehicle will do trip 4 and which one trip 5. Of course, we can only choose vehicles from depot 1 for trip 4. When making these decisions, we already take into account, in some way, the network after $T + l$.

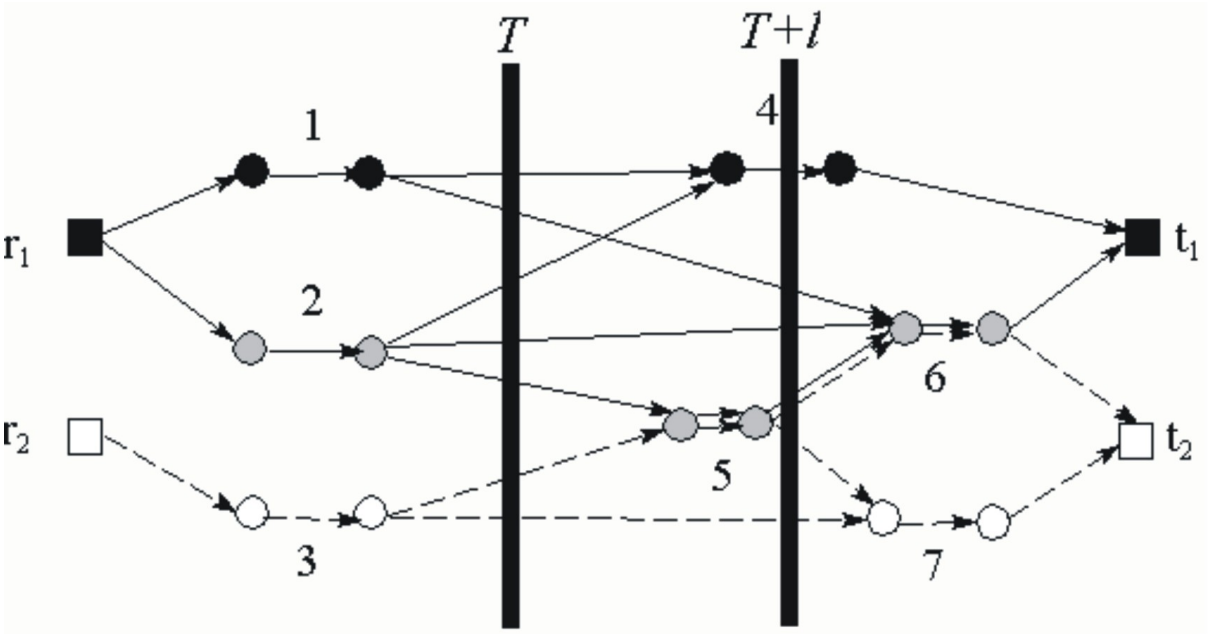

Figure 5.1: Example - Network G^1 and G^2 including the time points T and $T + l$

Our crucial assumption is that the travel times in the period $[T, T + l)$ are known with complete certainty. For the travel times after this period, we assume that we have information in the form of a number of possible scenarios, each with a certain probability of occurrence. These scenarios and the associated probabilities could be based on historical data, on subjective expert opinions or a combination of both. Note that one may choose to aggregate the scenarios into a single average scenario. Also note that in case no scenarios are available at all, one can still apply this approach in which the travel times after the next l periods are simply taken equal to the standard times that one also uses in the static problem.

In our current implementation these scenarios are based on historical data, which means that every scenario corresponds to a day in the past. Therefore, the scenarios and the probabilities of these scenarios will not vary over the day. Of course, in principle, it is possible that the planners modify these probabilities or the scenarios themselves during

the day. For instance, on a rainy day, they can give higher probabilities to scenarios corresponding to rainy days in the past.

After $k \leq l$ time units have passed, we repeat the above procedure, which means we construct a schedule for the period $[T + k, T + k + l)$. In our implementation the length of the different periods are all equal except for the first one. However, for the approach itself the length of the periods can also vary over the day.

The optimization problem that we have to solve repeatedly is a stochastic programming problem if we explicitly consider multiple scenarios for the future travel times. In the case that we consider only a single scenario with average or standard travel times, we have to solve a sequence of static vehicle scheduling problems.

5.2.1 Mathematical Formulation (Single-Depot)

We will formulate the problem that we want to solve at time point T, where we schedule for the period $[T, T + l)$ and we use scenarios for the period after $T + l$. Let S denote the set of scenarios (where possibly $|S| = 1$, i.e. we also consider the case with a single scenario). We again denote by N the set of trips and we define for every scenario $s \in S$ a network $G^s = (N \cup r \cup t, A)$ where r and t are the source and the sink corresponding to the depot, respectively, and A is the set of arcs between two trips, from r to every trip and from every trip to t. Recall from Subsection 2.2.2 that st_i, $trav(r, i)$ and $trav(i, t)$ are defined as the start time of trip i, the deadhead travel time from the depot to the start location of trip i and from the end location of trip i to the depot, respectively. For the end time of trip i, we have to make a distinction between trips that end in the period $[T, T + l)$ and after $T + l$ for every scenario s. Define these end times as et_i^0 and et_i^s, respectively. Let $trav(i, j)$ be the deadhead travel time from the end location of trip i to the start location of trip j. Furthermore, denote by A^* the set of arcs without long arcs, by A_1 the subset of A^* that corresponds to the period $[T, T + l)$, which is the following set:

- (r, j), if $st_j - trav(r, j) \in [T, T + l)$;
- (i, j), if $st_j - trav(i, j) \in [T, T + l)$;
- (i, t), if $et_i^0 \in [T, T + l)$.

In the same way, we can define A_2 as the subset of arcs corresponding to the period after $T + l$. Furthermore, we define p_s as the probability of scenario s occurring and c'_{ij} and c^s_{ij} as the cost of arc (i, j) in respectively period $[T, T + l)$ and the period after $T + l$ in scenario s. Here, the costs associated with an arc is a function of travel and idle time if the time between the trips i and j is nonnegative; otherwise it is a function of the delay or a sufficiently large number if delays are not allowed. Notice that by defining the cost

in this way, we can use the same set of arcs for all scenarios. As before, we define c as the fixed vehicle cost.

Similar as in Section 2.2, let tp^{sh} be the time points at which a vehicle may leave the depot to drive to the start location of a trip in scenario s and define H^s as the corresponding set. Let

$$a_{ij}^{sh} = \begin{cases} 1, \text{ if } et_i^s \leq tp^{sh} < st_j, \\ -1, \text{ if } st_j \leq tp^{sh} < et_i^s, \\ 0 \text{ otherwise,} \end{cases} \quad \text{for each arc } (i,j) \text{ with } i,j \in N,$$

$$a_{rj}^{sh} = \begin{cases} 1, \text{ if } st_j - trav(r,j) \leq tp^{sh} < st_j, \\ 0 \text{ otherwise,} \end{cases} \quad \text{for each arc } (r,j) \text{ with } j \in N, \text{ and}$$

$$a_{it}^{sh} = \begin{cases} 1, \text{ if } et_i^s \leq tp^{sh} < et_i^s + trav(i,t), \\ 0 \text{ otherwise,} \end{cases} \quad \text{for each arc } (i,t) \text{ with } i \in N.$$

For arcs $(i,j) \in A_1$ similar definitions hold, but then with et_i^0 instead of et_i^s. Furthermore, let b^{sh} be the number of trips carried out at tp^{sh}, i.e. we count all trips for which $st_i \leq tp^{sh} < et_i^0$ for trips ending in $[T, T+l)$ and $st_i \leq tp^{sh} < et_i^s$ otherwise . Note that if a trip starts late the corresponding vehicle is counted twice in b^{sh}. The problem of double counting is solved by the definition of the a_{ij}^{sh} parameters, since these are -1 in that case.

We use decision variables z_{ij} and y_{ij}^s, where $z_{ij} = 1$, if arc (i,j) is chosen in period $[T, T+l)$, $z_{ij} = 0$ otherwise and $y_{ij}^s = 1$, if arc (i,j) is chosen after $T+l$ in scenario s, $y_{ij}^s = 0$ otherwise. Furthermore, we also use B^s as decision variable for the number of buses in scenario s. Then we get the following 0-1 program, where we minimize the expected vehicle and delay costs.

(D-SDVSP-T):

$$\min \quad c \sum_{s \in S} p^s B^s + \sum_{(i,j) \in A_1} c'_{ij} z_{ij} + \sum_{s \in S} p^s \sum_{(i,j) \in A_2} c_{ij}^s y_{ij}^s \tag{5.1}$$

$$\text{s.t.} \quad \sum_{\{i:(i,j) \in A_1\}} z_{ij} + \sum_{\{i:(i,j) \in A_2\}} y_{ij}^s = 1 \qquad \forall s \in S, \forall j \in N, \tag{5.2}$$

$$\sum_{\{j:(i,j) \in A_1\}} z_{ij} + \sum_{\{j:(i,j) \in A_2\}} y_{ij}^s = 1 \qquad \forall s \in S, \forall i \in N, \tag{5.3}$$

$$b^{sh} + \sum_{(i,j) \in A_1} a_{ij}^{sh} z_{ij} + \sum_{(i,j) \in A_2} a_{ij}^{sh} y_{ij}^s \leq B^s \qquad \forall s \in S, \forall h \in H^s, \tag{5.4}$$

$$z_{ij} \in \{0,1\} \quad \forall (i,j) \in A_1, \tag{5.5}$$

$$y_{ij}^s \in \{0,1\} \quad \forall s \in S, \forall (i,j) \in A_2. \tag{5.6}$$

Constraints (5.2) and (5.3) assure that every trip has exactly one predecessor and one successor in every scenario. Furthermore, constraints (5.2) guarantee that if a trip i has

a successor j and (i, j) is in set A_1, this holds for all scenarios. A similar remark holds for constraints (5.3) and the predecessor of a trip. Finally, constraints (5.4) and the fact that $c > 0$ guarantee that B^s is the number of vehicles in scenario s.

5.2.2 Solution Method (Single-Depot)

For the single-depot vehicle scheduling problem, a solution with possibly some trips starting late can be obtained by solving a sequence of problems (D-SDVSP-T) for different values of T to optimality. In Figure 5.2, we give a schematic overview of our solution method.

Step 0: Choose initial parameters T, k and l ($\geq k$).
Step 1: Solve problem (D-SDVSP-T) for period $[T, T + l)$
Update T: $T := T + k$.
Repeat step 1 until the end of the day.

Figure 5.2: Solution method for D-SDVSP

We use the CPLEX MIP solver to compute an optimal solution for problem (D-SDVSP-T).

Notice that if we take a larger value for l the assumptions are less realistic, because we assume that the travel times are known for the period $[T, T+l)$ at moment T. Furthermore, we found that, in our experiments, solving the LP-relaxation often resulted in an integer solution. Of course, if we have only one scenario this problem is equivalent to the static SDVSP, which is known to be solvable in polynomial time.

5.2.3 Multiple-Depot Dynamic Vehicle Scheduling

For the dynamic approach to multiple-depot problems, we can formulate a stochastic programming problem that is a combination of (S-MDVSP) in Section 2.2 and (D-SDVSP-T) in Subsection 5.2.1. This formulation will be given later. In case of multiple scenarios, it turns out, however, that because of its size, it is hard to solve this formulation exactly in a straightforward way, for instance, by a commercial MIP solver. Therefore we use a so-called *cluster-reschedule heuristic*. In Figure 5.3, we give a schematic overview of the cluster-reschedule heuristic.

Step 1: Assign the trips to depots by solving (S-MDVSP).
Step 2: For each depot, apply the dynamic approach of Subsection 5.2.2.

Figure 5.3: Solution method for D-MDVSP

In the first step, we assign the trips to a certain depot and in the second step, we solve the (D-SDVSP) for all depots. This means that a trip is assigned to a depot for

the whole timetable period - recall from Chapter 1 that this is mostly half a year or one year - and that this cannot be changed anymore. For this reason, the heuristic is very attractive from a practical point of view, since it is already known on which subset of the trips vehicles from a certain depot may have to drive. In reality, where there are also drivers, this means that these drivers should only be educated about a subset of the trips.

Lower bound

To evaluate the quality of the cluster-reschedule heuristic, we compute a lower bound using Lagrangian relaxation. Therefore, we use the following formulation, which is a straightforward combination of (S-MDVSP) and (D-SDVSP-T).

(D-MDVSP-T):

$$\min \quad \sum_{d\in D}(c\sum_{s\in S}p^s B^{ds} + \sum_{(i,j)\in A_1^d} c_{ij}^{d'} z_{ij}^d + \sum_{(i,j)\in A_2^d}\sum_{s\in S} p^s c_{ij}^{ds} y_{ij}^{ds}) \tag{5.7}$$

$$\text{s.t.} \quad z_{ij}^d - y_{ij}^{ds} = 0 \qquad \forall s \in S, \forall d \in D, \forall (i,j) \in A_1^d, \tag{5.8}$$

$$\sum_{\{i:(i,j)\in A^{d*}\}} y_{ij}^{ds} - \sum_{\{k:(j,k)\in A^{d*}\}} y_{jk}^{ds} = 0 \qquad \forall s \in S, \forall d \in D, \forall j \in N, \tag{5.9}$$

$$\sum_{d\in D}\sum_{\{j:(i,j)\in A^{d*}\}} y_{ij}^{ds} = 1 \qquad \forall s \in S, \forall i \in N, \tag{5.10}$$

$$\sum_{d\in D}\sum_{\{i:(i,j)\in A^{d*}\}} y_{ij}^{ds} = 1 \qquad \forall s \in S, \forall j \in N, \tag{5.11}$$

$$\sum_{i\in N} b_i^{hs} \sum_{\{j:(i,j)\in A^{d*}\}} y_{ij}^{ds} + \sum_{(i,j)\in A^{d*}} a_{ij}^{dhs} y_{ij}^{ds} \leq B^{ds} \qquad \forall s \in S, \forall d \in D, \forall h \in H^{ds}, \tag{5.12}$$

$$z_{ij}^d \in \{0,1\} \qquad \forall d \in D, \forall (i,j) \in A_1^d, \tag{5.13}$$

$$y_{ij}^{ds} \in \{0,1\} \qquad \forall s \in S, \forall d \in D, \forall (i,j) \in A^{d*}. \tag{5.14}$$

To compute a lower bound, we associate Lagrangian multipliers λ_{ij}^{ds} and μ_j^s with constraints (5.8) and (5.9), respectively. Furthermore, we define the corresponding vectors λ and μ. The Lagrangian subproblem becomes:

$$\Phi(\lambda,\mu) = \Phi_y(\lambda,\mu) + \Phi_z(\lambda)$$

with

$$\Phi_y(\lambda,\mu) = \min \sum_{d\in D}(c\sum_{s\in S} p^s B^{ds} + \sum_{(i,j)\in A^{d*}}\sum_{s\in S} \overline{c}_{ij}^{ds} y_{ij}^{ds})$$
$$\text{s.t. (5.10)-(5.12), (5.14),}$$

and

$$\begin{aligned}\Phi_z(\lambda) &= \min \sum_{d\in D}\sum_{s\in S}\sum_{(i,j)\in A_1^d}(c_{ij}^d + \lambda_{ij}^{ds})z_{ij}^d \\ &\quad \text{s.t. (5.13)},\end{aligned}$$

where

$$\overline{c}_{ij}^{ds} = \begin{cases} p^s c_{ij}^{ds} - \lambda_{ij}^{ds} + \mu_j^{ds} - \mu_i^{ds} & \text{if } (i,j) \in A_1^d, i \in N, j \in N, \\ p^s c_{ij}^{ds} - \lambda_{ij}^{ds} + \mu_j^{ds} & \text{if } (i,j) \in A_1^d, i = r^d, j \in N, \\ p^s c_{ij}^{ds} - \lambda_{ij}^{ds} - \mu_i^{ds} & \text{if } (i,j) \in A_1^d, i \in N, j = t^d, \\ p^s c_{ij}^{ds} + \mu_j^{ds} - \mu_i^{ds} & \text{if } (i,j) \in A_2^d, i \in N, j \in N, \\ p^s c_{ij}^{ds} + \mu_j^{ds} & \text{if } (i,j) \in A_2^d, i = r^d, j \in N, \\ p^s c_{ij}^{ds} - \mu_i^{ds} & \text{if } (i,j) \in A_2^d, i \in N, j = t^d, \end{cases}$$

Let $\overline{y}(\lambda,\mu)$ and $\overline{z}(\lambda)$ denote optimal solutions corresponding to $\Phi_y(\lambda,\mu)$ and $\Phi_z(\lambda)$, respectively, for given λ and μ. Then $\overline{y}(\lambda,\mu)$ is obtained by solving a S-SDVSP for every scenario, and $\overline{z}(\lambda)$ is obtained by pricing out each variable, that is, for each $d \in D$ and $(i,j) \in A_1^d$, $\overline{z}_{ij}^d = 1$ if $c_{ij}^d + \lambda_{ij}^{ds} \leq 0$ and $\overline{z}_{ij}^d = 0$ otherwise.

We use subgradient optimization to solve the Lagrangian dual problem $\max_{\lambda,\mu} \Phi(\lambda,\mu)$ approximately. In the subgradient optimization, we use as upper bound the value of the solution produced by the cluster-reschedule heuristic. It is worthwhile to note that we found that the gap between the upper and lower bound is already small when we did not use the subgradient algorithm, but just choose the Lagrangian multipliers equal to 0, which is equivalent to deleting the relaxed constraints. Notice that, for the static MDVSP, Löbel (1997) also found that by deleting constraints (5.9) already a good lower bound can be obtained.

5.2.4 Computational Experience

We have evaluated our approach by using data from Connexxion, the largest bus company in the Netherlands. In this subsection, we discuss the data and the results of the static vehicle scheduling problem. Furthermore, we discuss the results of our dynamic approach. All tests reported in this subsection are executed on a Pentium III 450MHz personal computer (128MB RAM).

Data Description and Results S-VSP

The data set consists of 1104 trips and 4 depots in the area between Rotterdam, Utrecht and Dordrecht, three large cities in the Netherlands. On a typical workday, there are a lot of traffic jams in this area, especially during rush hours (in the morning towards Rotterdam and Utrecht and in the afternoon in the opposite direction). Of course, most trips are also during these hours, which can be seen in Figure 5.4.

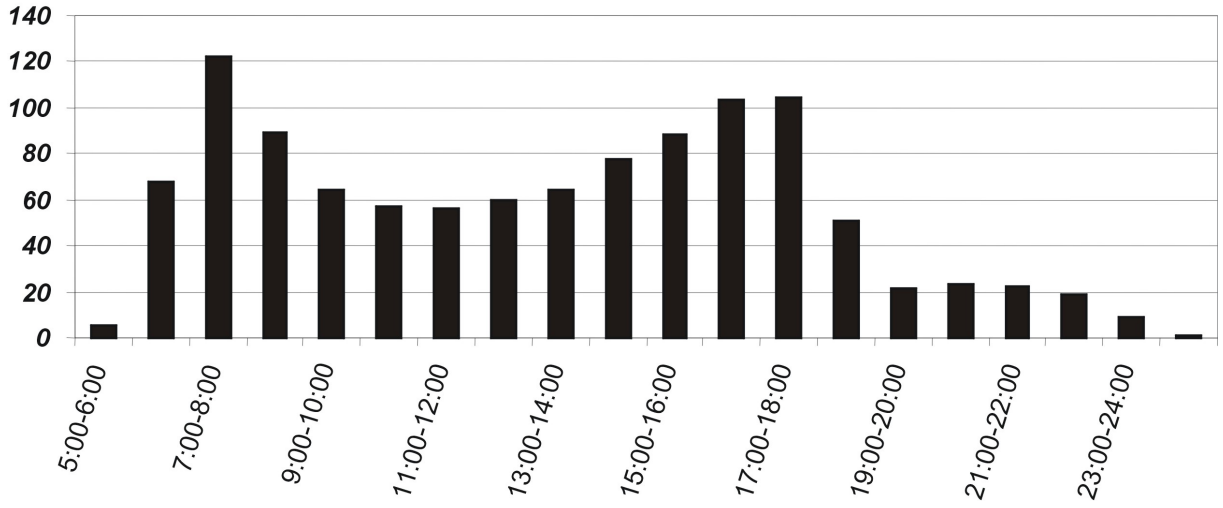

Figure 5.4: Distribution of the trips over the day

In this figure one can also see that the minimum number of vehicles will be determined during one hour in the morning peak (from 7 am until 8 am). Furthermore, it is important to note that not all trips are allowed to be driven by a vehicle from every depot. In fact, almost half of the trips can only be assigned to one depot and only a very small number can be assigned to all depots. On average, a trip can be assigned to 1.71 depots. Furthermore, we have historical data concerning the travel times for a period of 10 days (2 weeks from Monday to Friday). These are only the travel times for trips and not for deadheads. Therefore, we implicitly consider the travel times for deadheads as fixed. Notice, however, that a similar approach can also be used if this is not the case. Furthermore, since we only have information about delays in these data, we implicitly assume that the actual travel time of a trip is never less than the travel time in the timetable. Therefore, delays are always nonnegative and a bus is never too early at the end location. In practice, this will never happen if a driver just waits at each stop until it is time to depart.

We assume that a bus will leave exactly at the start time of a trip if this is possible. Furthermore, we assume that the difference between the realized and actual travel time of each trip is independent of its initial delay, i.e. the one at the start of the trip. This is realistic, because the frequencies at the different lines are quite low (e.g. every half hour or hour), which means that the number of passengers does not increase significantly if the trip starts late. This is in contrast with urban transport, where the frequencies are typically much higher, e.g. every 10 minutes. Then, if a trip starts more than 5 minutes late, it gets an additional delay that depends on the actual starting time, since there are more passengers at the different bus stops taking this trip who would have taken the next one if this trip had no delay.

We use 10,000 as fixed costs per vehicle and variable vehicle costs of 1 per minute time that a vehicle is without passengers. If we solve this problem by using the static vehicle

scheduling problem, we get the solution in Table 5.2.

vehicle costs	1,102,538
number of vehicles	109 (9-8-37-55)
cpu (sec.)	64

Table 5.2: Optimal solution of the S-VSP

The numbers between brackets show how the optimal number of vehicles is split over the depots. The computation time is about one minute when using the MIP solver of CPLEX, version 6.5, to solve model (S-MDVSP) in Section 2.2. Unfortunately, if we apply this optimal schedule on the 10 days for which we have historical data (realized travel times) available, a large number of trips will start late. In Table 5.3, we show, for these 10 days, the percentage of trips that start late and the cost of these delays, when we use as cost function $10x^2$, where x is the time in minutes that the trip starts late. We take a quadratic cost function, because a few small delays are preferred above one large delay. Furthermore, we scale it by a factor 10 to get the delay costs in the same order of magnitude as the vehicle costs. This means that it is better to introduce one vehicle more if a delay of 32 minutes is prevented. As can be seen in the last column, on average 17.2% of the trips are starting late. Furthermore, one can see that there are much less delays on Friday's (day 5 and 10) compared to the other days.

day	1	2	3	4	5	
trips late (%)	20.5	19.2	21.9	18.1	12.3	
costs	146,350	116,290	89,490	51,080	107,550	

day	6	7	8	9	10	av.
trips late (%)	16.9	14.2	20.3	17.0	11.7	17.2
costs	134,940	88,520	112,870	171,660	59,550	107,830

Table 5.3: Results when using the optimal solution of the S-VSP

Of course, we can reduce the number of trips starting late and the delay costs by introducing fixed buffer times. The problem is still static, but we can take care of small delays. Table 5.4 shows the optimal solution, the average number of trips starting late and the average costs of these delays by using a fixed buffer time of respectively 2, 5, 7 and 10 minutes after every trip. The detailed results for all different days are shown in Table 5.28 in the appendix.

It is obvious that the number of trips starting late and the delay costs can be reduced in this way, but on the other hand the number of vehicles and the vehicle costs increase enormously. But even by introducing 5 or 10 minutes buffer time and thus using 8 respectively 16 vehicles more, there is still a significant number of trips starting late.

buffer time	no	2 min.	5 min.	7 min.	10 min.
vehicle costs	1,102,538	1,135,605	1,189,867	1,232,984	1,277,535
number of vehicles	109	112	117	121	125
trips late (%)	17.2	7.6	3.3	2.1	1.2
delay costs	107,830	55,174	28,424	18,683	14,254

Table 5.4: Results of introducing fixed buffer times

Moreover, these are most of the time the trips starting very late, which means that the delay costs are still quite high.

Results D-VSP

We show the results of our approach to the dynamic vehicle scheduling problem and compare it with the results of the static case. We have used the following parameter settings.

- The first period starts when the first vehicle leaves the depot and ends at 7 am. The length of the other periods are equal to l, where we vary the value of l (1, 5, 10, 15, 30, 60 and 120 minutes).
- Rescheduling only takes place at the start of a period ($k = l$), which means we follow a time-driven approach. Notice, however, that if $k = 1$ the approach coincides with an event-driven approach, since the rescheduling takes place every time a delay occurs.
- With respect to the vehicle costs, the cost structure is the same as in the static case.
- Since our first goal is to minimize the number of delays, a cost of 10,000 is incurred for every trip starting late (independent of the size of the delay). However, we still use the cost function defined in the previous subsection for evaluation purposes.
- We consider each of the 10 days for which we have historical data separately. In the case of using multiple scenarios (referred to as I), we took the realizations of the other 9 days as scenarios, where we gave one scenario (the same day but in the other week) a probability of 0.2 and the others a probability of 0.1. So if we optimize day 1, we took as scenarios the realizations of day 2 until day 10, where day 6 has a probability of 0.2 and the others 0.1.
- In the case of a single average scenario (referred to as II), we computed this average scenario by the weighted average of the 9 scenarios as described above.

The average results over all days of the cluster-reschedule heuristic for D-VSP described in Subsection 5.2.3 are shown in Table 5.5. We show the results for the cases I

and II, as well for the case where we do not use the historical data (column 0) for different settings of l. In this table #V means the number of vehicles used, %L the percentage of trips starting late and DC the cost of these delays.

	0			I			II		
l	#V	%L	DC	#V	%L	DC	#V	%L	DC
120	111.7	2.22	11,615	113.2	0.53	1,672	113.8	0.85	8,630
60	110.9	3.84	19,567	114.2	0.76	1,827	114.9	1.39	5,981
30	110.4	5.76	29,628	113.8	1.03	3,714	115.3	2.17	10,971
15	109.7	9.22	51,551	113.5	1.73	9,224	115.7	3.60	25,365
10	109.6	9.58	49,126	113.8	1.83	6,835	115.9	3.87	22,577
5	109.3	11.12	61,424	114.0	1.97	7,788	116.1	4.40	23,939
1	109.2	12.38	65,785	114.2	2.14	8,960	115.9	4.99	25,716

Table 5.5: Average results D-VSP

The case where we do not use historical data and we take k and l equal to 1 can be seen as an online approach where we know at the start of the day the travel times until 7 am and we modify the schedules every time a delay occurs. One can see that there are still a lot of delays in the case, where we do not consider historical data. Therefore, we will focus in the remaining of the paper on the cases I and II. In the appendix (Tables 5.26 and 5.27), we give a more detailed overview of the results for these cases.

It is obvious that the quality of the results decreases if l decreases, because we assume that we know the realizations of the travel times l minutes in advance. Notice that in reality it is very difficult to predict these travel times a long time before their actual realization. Furthermore, we can see that using more scenarios (case I) leads to less delays and most often to less vehicles than case II. So case I clearly outperforms case II. If we compare the results of case I with the results in the previous subsection, one can see that for $l = 1$ the average number of vehicles used is 114.2, the average number of trips starting late is 2.1% and the average delay costs are 8,960. These numbers are much better than the average results of the static case with a fixed buffer time of 5 minutes (see Table 5.4), which means that we clearly outperform the traditional static solution. Only by introducing a buffer time of 10 minutes, we get on average less trips starting late (1.2% instead of 2.1%), but 125 vehicles are needed every day instead of 114.2 on average, which is a very significant difference. Although, we did not optimize on the size of the delays, also the delay costs are clearly lower than in the static case.

In Table 5.6, we show the computation time in seconds (column cpu), the number of iterations (it.) and the maximum computation time of one iteration in seconds (max.) for the largest depot. The computation time for the other depots is much smaller than for depot 4.

	I			II		
l	cpu	it.	max.	cpu	it.	max.
120	117	10	47	8	10	1
60	164	19	39	13	19	1
30	270	37	37	22	37	1
15	496	71	37	41	71	1
10	702	104	39	60	104	1
5	1,687	204	55	122	204	1
1	6,088	861	55	535	857	1

Table 5.6: Computation times for D-VSP (depot 4)

Of course, the computation time for case I is much higher than for case II, but it is important to note that the computation time per iteration and per depot is always less than l, which means that this approach makes sense in practice.

Lower bound We evaluate our cluster-reschedule method for solving the D-VSP by comparing its results to lower bounds per iteration and to the situation where we have perfect information.

Because we use a heuristic in every iteration, it is interesting to compare its results with a lower bound on the optimal solution in each iteration (see Subsection 5.2.3). Therefore, we give in Table 5.7 the relative gap between the lower and the upper bound in the first iteration, where the optimization problem is the most difficult one since it is the largest. Because the gaps are almost the same for the different days, we only show the minimum, maximum and average gap over all days. Since the gap is reasonably small in the first iteration, this gives an indication that the difference between an exact algorithm and our cluster-reschedule heuristic is small in all iterations.

	I	II
minimum	3.42%	5.11%
maximum	3.60%	5.95%
average	3.51%	5.70%

Table 5.7: Gap in the first period

Of course, in case of perfect information, which means that we know all realizations of the travel times in advance, we get a lower bound on our problem. In this case, we can ensure that no trips are starting late and thus we only have to minimize the total vehicle costs. This can be done by a heuristic or exact method. In the first case we first cluster the trips by solving S-MDVSP like in the cluster-reschedule heuristic and then schedule the trips to optimality. The results are shown in Table 5.8.

day	1	2	3	4	5	6	7	8	9	10	av.
heuristic	117	113	114	116	113	115	115	115	114	113	114.5
optimal	111	110	110	112	110	110	111	112	110	110	110.6

Table 5.8: Results of the heuristic and optimal solution in the case of perfect information

If we compare the solution of case I in Table 5.5 with the solution above in the case of using the same heuristic, it is very small. For example, on average we get 2.1% of the trips starting late while we even save 0.3 vehicles if we obtain the relevant information only 1 minute in advance compared to the situation where we have full information. Furthermore, we can see that the difference between the heuristic and the optimal solution with perfect information is 3.5 % on average, which is similar to the results shown in Table 5.7. This means that the solution of the cluster-reschedule heuristic is close to the optimal solution in every iteration.

Sensitivity analysis In Table 5.5, we see that the costs are lower for larger values of l. This is, of course, because we assume that the travel times are known (can be estimated without any error) at time point T for the period $[T, T+l)$. Especially for large values of l, this assumption may be unrealistic. Therefore, we have performed a sensitivity analysis by considering small deviations of the actual travel times from the estimated ones. In this analysis, we use the same schedules as generated in the experiments discussed before. These are considered to be based on estimated travel times. We evaluate the schedules with respect to actual travel times that may differ slightly from the estimated ones. Note that evaluation means the calculation of the number of trips starting late and the delay costs with the simulated travel times. The number of vehicles does not change. Furthermore, we note that the sensitivity analysis is carried out to test whether our approach is robust. This is different than in, for example, Daduna *et al.* (1993), where sensitivity analysis is carried out on static vehicle scheduling problems to show that small changes in the start times of the trips can improve the optimal solution significantly.

We have simulated four times 100 runs of actual travel times, where the actual travel time is drawn from a normal distribution with mean equal to the estimated travel time and variance σ^2. Furthermore, we still assume that the total delay is nonnegative. In Table 5.9, we show the average number of trips starting late (%L) and the average delay costs (DC) for case I and II for different values of l and σ. Note that we take the average results over all days.

It is obvious that the number of trips starting late and the delay costs increase, but they are still much less than in the static case. Of course, it is reasonable that the estimates of the travel times become better if l decreases. Recall that, for l equal to 1, 2.1% of the trips were starting late with a cost of 8,960, while we have better results for l equal to 30 and small deviations from the estimated travel times. Furthermore, if we

		I		II	
l	σ	%L	DC	%L	DC
120	2	13.0	12,626.4	13.8	21,724.2
120	1	6.8	3,876.5	7.3	11,435.1
120	0.5	3.3	2,219.9	3.7	9,371.1
120	0.25	0.9	1,745.9	1.3	8,742.6
60	2	12.5	12,864.2	13.9	18,757.2
60	1	6.7	4,106.7	7.6	9,151.6
60	0.5	3.3	2,400.7	4.1	7,310.6
60	0.25	1.1	1,883.5	1.8	6,766.9
30	2	12.7	12,229.9	13.8	22,252.2
30	1	6.7	5,374.2	7.7	13,267.0
30	0.5	3.4	4,161.8	4.5	11,629.6
30	0.25	1.4	3,794.7	2.5	11,071,7

Table 5.9: Average results with extra perturbation $\epsilon \sim N(0, \sigma^2)$

compare the results of the Tables 5.5 and 5.9, one can see that small perturbations have only a small impact on the performance of our method.

5.2.5 Conclusion

The results reported in the previous subsection show that we can reduce the number of trips starting late and the delay costs at the price of using only a few vehicles more if we use our dynamic method instead of the traditional static one. The case where we use multiple scenarios to describe the future travel times clearly outperforms the case with only a single average scenario. However, our method uses the fact that the travel times are known a certain time before realization. It is obvious that this is only realistic if this time is small, but even then our method clearly outperforms the static one. Furthermore, the impact of small deviations from the estimated travel time on the performance of our method is quite small.

It is very important that the optimization problem in every iteration of our method is solved quite fast. Therefore, we have not used an exact approach, but a cluster-reschedule heuristic, where we first cluster the trips using the static VSP and then we dynamically reschedule the trips per depot. We have shown that the gap between this cluster-reschedule heuristic and a lower bound on the overall problem is quite small. Finally, we have shown that also the gaps between the solutions generated by our method and the optimal solutions with perfect information are reasonable.

In the next section, we will integrate the dynamic vehicle scheduling with crew scheduling such that the whole process can be done dynamically, which is necessary if such a

dynamic approach will be used in practice.

5.3 Dynamic Vehicle and Crew Scheduling

In this section we generalize the idea of dynamic vehicle scheduling to dynamic vehicle and crew scheduling. Furthermore, we compare this again with the traditional approach of static vehicle and crew scheduling with buffer times. Notice from the previous chapters that solving the integrated vehicle and crew scheduling problem asks much more computational power than the vehicle scheduling problem. Therefore, dynamic approaches can only be used if the underlying problems are solved heuristically, which has of course a negative effect on the quality. The question is whether such a negative effect can be compensated by the positive effect of using a dynamic approach.

In this section we consider six measures to evaluate a solution: the total number of vehicles and drivers used, the percentage of trips starting late, the "virtual" delay costs, the percentage of duties violating at least one of the crew rules and a "virtual" cost measure for those violations. Notice that three of these measures were already used in the previous section.

Before we introduce some solution approaches, we start the discussion about dynamic vehicle and crew scheduling with some basic assumptions.

Assumptions

Due to the fact that we also consider crew scheduling aspects, an important difference with the previous section is that we can sometimes choose between trips starting late and violating crew rules. To explain this consider the following example with two trips, one trip ending at A at 10:00 and another starting from A at 10:15. Furthermore, there should be a break of at least 15 minutes in the duty. Moreover, suppose that both trips are in one duty and the break is between these two trips. However, if the first trip arrives late, we do not have a feasible duty anymore or the second trip starts late. Thus we have the possibility of choosing between violating crew rules and trips starting late. It is obvious that similar problems can occur for other crew rules. Therefore, we assume that passengers have a higher priority than drivers, which means that if there is a possibility to choose between a violation of one or more restrictions of a duty and a trip starting late, the first option is always chosen.

Furthermore, we assume that a trip can only start late due to a delay of the vehicle and thus not due to the driver. That is whenever a changeover occurs this cannot lead to an extra delay on the new vehicle. Since changeovers only occur during the break, we assume that the delays are not larger than the minimum break length. Later on we will see that in our test problems such a delay never occurs and therefore such an assumption is reasonable.

Similarly to the previous section, where we assumed that the number of vehicles is unlimited, we assume here that also the number of crews is unlimited. However, the models and solution approaches can be modified to relax this assumption. As a consequence of this assumption, the number of duties per type are not known beforehand. However, if a duty of a certain specified type has started, the type will not be changed anymore. Therefore, the driver has an indication at which time he will finish his duty, although, he does not know it exactly.

Finally, for evaluation of the schedules we assume that the start time of the two pieces is fixed. Since if a driver reliefs another one at the start of a piece and the bus has a delay, the driver is already there at the scheduled time and does not have the advantage that he could actually start later. Therefore, the total length of pieces, work time and so on, can only increase and not decrease.

5.3.1 Different Approaches

In the case of multiple-depots, all algorithms use again the *cluster-reschedule heuristic* shown in Figure 5.5. This algorithm has, as already mentioned in Subsection 5.2.3, an important advantage in practice, since drivers should only be educated about a subset of the trips.

Step 1: Assign the trips to depots by solving (S-MDVSP).
Step 2: For each depot, apply a dynamic approach for the single-depot case.

Figure 5.5: Solution method for D-MDVCSP

Since multiple-depot problems are solved as several single-depot problems, we only focus on the single-depot case in the remainder of this subsection. We developed two algorithms to solve the dynamic vehicle and crew scheduling problem with a single depot. There is an algorithm (called D-VSP-CSP) which uses the sequential approach, i.e. first vehicle scheduling and then crew scheduling, and the other one (called D-VCSP) uses the integrated approach. Furthermore, for both approaches we consider two cases (like in Section 5.2), one with one scenario and one with more scenarios for the travel times. A schematic overview of the first algorithm is provided in Figure 5.6. The second algorithm is postponed to the next subsection.

Notice that in algorithm D-VSP-CSP we solve problem (D-SDVSP-T), which is discussed in Subsection 5.2.1. Afterwards, we solve the corresponding crew schedule for the main scenario, i.e. the scenario with the highest probability (in case there is more than one such scenario, we choose one of them arbitrary). This means that we just have to solve a standard crew scheduling problem. However, we need to take into account that decisions made before time point T cannot be changed anymore. This can be handled in the construction of the set of duties K. In fact the requirement that there should

Step 0: Choose initial parameters T, k and l ($\geq k$).
Step 1: Solve problem (D-SDVSP-T) for period $[T, T+l)$.
Solve the corresponding CSP for the main scenario.
Update T: $T := T + k$.
Repeat step 1 until the end of the day.

Figure 5.6: Solution method for D-VSP-CSP

be no contradiction with decisions made earlier, is just an extra feasibility check when we construct this set. Therefore, this does not have any impact on the structure of the underlying set partitioning problem, which can still be solved as described in Section 2.3.

5.3.2 Algorithm D-VCSP

An overview of algorithm D-VCSP is given in Figure 5.7.

Step 0: Choose initial parameters T, k and l ($\geq k$).
Step 1: Solve problem (D-SDVCSP-T) for period $[T, T+l)$.
Update T: $T := T + k$.
Repeat step 1 until the end of the day.

Figure 5.7: Solution method for D-VCSP

The algorithm solves a sequence of integrated vehicle and crew scheduling problems. In this subsection we provide a mathematical formulation for this problem. Hereby, we assume again that one of the scenarios (s^*) is the main scenario and that the variables related to the crew scheduling part of the problem are defined on this scenario. We only consider the general case with a single depot, i.e. an extension of formulation (VCSP1) from Subsection 3.3.1. Furthermore, we describe the modifications on the corresponding algorithm of Subsection 3.3.3.

Mathematical Formulation (D-SDVCSP-T)

Before providing the mathematical formulation we need to recall some notation. For the notation with respect to the vehicle scheduling part of the formulation we refer to Subsection 5.2.1, since it is completely similar.

Recall from Section 3.3 that K denotes the set of duties and that f_k denotes the crew cost of duty $k \in K$, respectively. Furthermore, I_1 denotes the set of trip tasks and $K(p)$ is the set of duties covering trip task $p \in I_1$. $K(i,j)$ denotes the set of duties covering dh-tasks corresponding to deadhead $(i,j) \in A^*$. Finally, binary decision variables x_k are defined as 1 if duty k is selected in the solution and 0 otherwise.

The problem can be formulated as follows.

(D-SDVCSP-T):

$$\min \quad c\sum_{s\in S} p^s B^s + \sum_{(i,j)\in A_1} c'_{ij} z_{ij} + \sum_{s\in S} p^s \sum_{(i,j)\in A_2} c^s_{ij} y^s_{ij} + \sum_{k\in K} f_k x_k \tag{5.15}$$

$$\text{s.t.} \quad \sum_{\{i:(i,j)\in A_1\}} z_{ij} + \sum_{\{i:(i,j)\in A_2\}} y^s_{ij} = 1 \qquad \forall s \in S, \forall j \in N, \tag{5.16}$$

$$\sum_{\{j:(i,j)\in A_1\}} z_{ij} + \sum_{\{j:(i,j)\in A_2\}} y^s_{ij} = 1 \qquad \forall s \in S, \forall i \in N, \tag{5.17}$$

$$b^{sh} + \sum_{(i,j)\in A_1} a^{sh}_{ij} z_{ij} + \sum_{(i,j)\in A_2} a^{sh}_{ij} y^s_{ij} \leq B^s \qquad \forall s \in S, \forall h \in H^s, \tag{5.18}$$

$$\sum_{k\in K(p)} x_k = 1 \qquad \forall p \in I_1, \tag{5.19}$$

$$\sum_{k\in K(i,j)} x_k - z_{ij} = 0 \qquad \forall (i,j) \in A_1, \tag{5.20}$$

$$\sum_{k\in K(i,j)} x_k - y^{s^*}_{ij} = 0 \qquad \forall (i,j) \in A_2, \tag{5.21}$$

$$x_k, z_{ij} \in \{0,1\} \quad \forall k \in K, \forall (i,j) \in A_1, \tag{5.22}$$

$$y^s_{ij} \in \{0,1\} \quad \forall s \in S, \forall (i,j) \in A_2. \tag{5.23}$$

The objective function (5.15) minimizes the total sum of vehicle, crew and delay costs. The first three sets of constraints, (5.16) - (5.18), correspond to the formulation of the dynamic vehicle scheduling problem (see Subsection 5.2.1). Constraints (5.19) assure that each trip task is in a duty. Finally, the constraints (5.20) and (5.21) guarantee the link between dh-tasks and deadheads in the solution. Notice that it is not necessary here to distinguish between short and long arcs since the fixed vehicle costs are dealt with separately.

Algorithm (D-SDVCSP-T)

The algorithm for D-SDVCSP-T is shown in Figure 5.8.

The differences between this algorithm and the one for VCSP1 are in the steps 0, 1 and 3. Step 2 is exactly similar, since it only deals with the generation of columns and another vehicle scheduling problem does not influence this. The main difference is found in Step 1, where constraints (5.19)-(5.21) are first replaced by set covering constraints, which are subsequently relaxed in a Lagrangian way. That is, we associate non-negative Lagrangian multipliers λ_p, μ_{ij} and ν^s_{ij} with constraints (5.19), (5.20) and (5.21), respectively. Then the remaining Lagrangian subproblem can be solved by pricing out the x variables and the following problem for the y and z variables:

Step 0: Initialization
Solve D-SDVSP-T and CSP for the main scenario and
take as initial set of columns the duties in the CSP-solution.
Step 1: Computation of dual multipliers
Solve a Lagrangian dual problem with the current set of columns.
This gives a lower bound for the current set of columns.
Step 2: Generation of columns
Generate columns (duties) with negative reduced cost.
Compute an estimate of a lower bound for the overall problem.
If the gap between this estimate and the lower bound found in Step 1 is small enough (or another termination criterion is satisfied), go to Step 3; otherwise, return to Step 1.
Step 3: Construction of feasible solution
Based on feasible vehicle solution(s) for scenario s^* from Step 1,
construct corresponding feasible crew solution(s).

Figure 5.8: Solution method for D-SDVCSP-T

$$\min \quad c\sum_{s\in S} p^s B^s + \sum_{(i,j)\in A_1} (c'_{ij} - \mu_{ij}) z_{ij} + \sum_{s\in S}\sum_{(i,j)\in A_2} \overline{c}^s_{ij} y^s_{ij} \tag{5.24}$$

$$\sum_{\{i:(i,j)\in A_1\}} z_{ij} + \sum_{\{i:(i,j)\in A_2\}} y^s_{ij} = 1 \qquad \forall s \in S, \forall j \in N, \tag{5.25}$$

$$\sum_{\{j:(i,j)\in A_1\}} z_{ij} + \sum_{\{j:(i,j)\in A_2\}} y^s_{ij} = 1 \qquad \forall s \in S, \forall i \in N, \tag{5.26}$$

$$b^{sh} + \sum_{(i,j)\in A_1} a^{sh}_{ij} z_{ij} + \sum_{(i,j)\in A_2} a^{sh}_{ij} y^s_{ij} \leq B^s \qquad \forall s \in S, \forall h \in H^s, \tag{5.27}$$

$$z_{ij} \in \{0,1\} \quad \forall (i,j) \in A_1, \tag{5.28}$$

$$y^s_{ij} \in \{0,1\} \quad \forall s \in S, \forall (i,j) \in A_2, \tag{5.29}$$

where

$$\overline{c}^s_{ij} = \begin{cases} p^s c^s_{ij} - \nu^s_{ij} & \text{if } (i,j) \in A_2, s = s^*, \\ p^s c^s_{ij} & \text{if } (i,j) \in A_2, s \neq s^*. \end{cases}$$

Notice that this problem is equivalent to problem (D-SDVSP-T). Moreover, we will again use the CPLEX MIP solver to compute an optimal solution for this problem.

At the end we compute a feasible crew schedule given the (feasible) vehicle schedule for scenario s^* which resulted from solving the last Lagrangian subproblem. Of course, it is again possible to compute more feasible solutions by solving the CSP not only for the vehicle solution from the last iteration, but also for vehicle solutions which were encountered earlier on.

5.3.3 Computational Experience

We have evaluated our approach by using a small single-depot and medium-sized multiple-depot data set from Connexxion, which are derived from the large data set in the previous section (the second one is equivalent with instance 5 in Section 4.5). The problems, denoted as prob_A and prob_B, consist of 164 and 304 trips, respectively. Moreover, the restrictions w.r.t. the feasibility of a duty that have to be taken into account are the same as in Section 4.5. Furthermore, the assumptions about the realizations of the travel times as described extensively in Subsection 5.2.4 still holds, i.e. the deadhead times are fixed and all delays are nonnegative. Moreover, the maximum delay that occurs in these data is less than 45 minutes, the minimum break length. Therefore, the assumption made that a trip can only start late due to a delay of the vehicle holds. Finally, notice that all tests reported in this subsection are again executed on a Pentium III 450MHz personal computer (128MB RAM). However, in this subsection version 7.1 of CPLEX is used (like in Chapter 4), while version 6.5 was used in Subsection 5.2.4.

Results Static Approach

We use 1,000 as fixed costs per vehicle and variable vehicle costs of 1 per minute time that a vehicle is without passengers. Furthermore, we use 1,000 as fixed costs for each duty. To solve the static problem, we can choose between the sequential and the integrated approach. In the sequential approach we solve the static vehicle scheduling problem and afterwards the crew scheduling problem (see Chapter 2), while in the integrated approach we solve the whole problem at once. In Chapter 4, two algorithms are developed for the integrated approach in the multiple-depot case. Here, in this section we will use the first algorithm, i.e. the one discussed in Section 4.3, where we will use the same parameter settings as in the previous chapter (see Section 4.5 for a detailed description of these parameter settings).

Furthermore, we calculate, for the 10 days for which we have data of the realizations of the travel times, the percentage of trips that start late and the cost of these delays in the same way as in Subsection 5.2.4. That is, we use as cost function $10x^2$, where x is the time in minutes that the trip starts late. Finally, we check, for these 10 days, the percentage of duties that violate one or more of the restrictions and the cost of these violations. Here, we use for each duty the cost function $10y_1^2 + 2y_2^2 + 2y_3^2 + y_4^2 + y_5^2 + y_6^2$, where y_1 is the number of minutes that the break in the duty is shorter than the minimum break length, y_2 (y_3) is the number of minutes that the maximum duty length (working time) is exceeded, y_4 (y_5) is the number of minutes that the maximum length of the first (second) piece is exceeded and y_6 is the number of minutes that the latest end time is exceeded. We calculate the total violation costs as the sum of these costs of each duty.

In Tables 5.10 and 5.11, the best feasible solutions for prob_A and prob_B of the static

as well as the integrated approach are denoted, respectively. Furthermore, we give the following averages, which are explained above, over the 10 days: the percentage of trips starting late, the delay costs, the percentage of duties violated and the violation costs. We give these results without buffer times (standard problem) and with buffer times of 2, 5, 7 and 10 minutes, respectively. In this table #V and #D denote the number of vehicles and drivers used, respectively, %L the percentage of trips starting late, DC the cost of these delays, %DV the percentage of duties violated and VC the total cost of these violations.

buffer	no		2 min.		5 min.		7 min.		10 min.	
	seq.	int.	seq.	int.	seq.	int.	seq.	int.	seq.	int.
#V	17	17	20	20	24	24	24	24	25	25
#D	32	31	35	32	40	37	41	39	42	41
%L	16.2	15.1	8.8	7.6	3.9	3.8	2.4	3.0	2.0	2.2
DC	17,272	17,812	7,765	7,360	2,925	3,927	2,612	3,264	2,681	2,178
%DV	7.8	11.6	7.4	10.0	3.8	4.9	4.1	4.6	6.2	5.6
VC	825	569	538	1,224	934	69	97	164	322	191

Table 5.10: Average results of the S-VCSP - prob_A

buffer	no		2 min.		5 min.		7 min.		10 min.	
	seq.	int.	seq.	int.	seq.	int.	seq.	int.	seq.	int.
#V	40	40	41	41	43	43	44	44	44	44
#D	74	65	76	66	81	71	82	71	82	71
%L	9.9	9.2	5.6	4.9	2.9	2.3	2.1	1.9	1.8	1.5
DC	19,897	15,691	8,742	7,206	3,906	4,830	2,741	2,721	1,963	1,994
%DV	8.9	15.7	5.8	10.7	7.6	10.3	4.2	9.4	8.5	8.4
VC	1,499	6,935	1,153	6,865	3,612	2,211	1,621	5,320	4,495	5,080

Table 5.11: Average results of the S-VCSP - prob_B

It is obvious that the numbers of vehicles and drivers increase when we introduce buffer times, while the percentage of trips starting late (and its costs) decreases. Furthermore, it is noteworthy to mention that the effect of buffer times is much higher in prob_A than in prob_B. The numbers of vehicles and drivers increase with about 50% and 30%, respectively, by introducing a buffer time of 10 minutes in prob_A, while this is only about 10% in prob_B. This effect is caused by several differences in the data. First of all, prob_B has more trips and therefore, there are more options to combine trips. Secondly, the duration of the trips in prob_A is slightly smaller, which means that the same buffer time is relatively larger for prob_A.

Results Dynamic Approach

We show the results of our approach to the dynamic vehicle and crew scheduling problem and compare them with the results of the static case. We have used the following parameter settings.

- The first period starts when the first vehicle leaves the depot and ends at 7 am. The length of the other periods are equal to l. We only consider the case where l is equal to 120 and 60 minutes.
- Rescheduling only takes place at the start of a period ($k = l$).
- The vehicle and crew costs are the same as in the static case.
- A cost of 500 is incurred for every trip starting late (independent of the size of the delay). For evaluation purposes, we still use the cost function defined in the previous subsection, but our first goal is to minimize the number of delays. There are no costs taken into account for violating duties during the optimization. However, we still use the cost function from the previous subsection to evaluate the solutions.
- We consider each of the 10 days for which we have historical data separately. In the case of using multiple scenarios (referred to as I), we took the realizations of the other 9 days as scenarios, where we gave one scenario (the same day but in the other week) a probability of 0.2 and the others a probability of 0.1. So if we optimize day 1, we took as scenarios the realizations of day 2 until day 10, where day 6 has a probability of 0.2 and the others 0.1.
- In the case of a single average scenario (referred to as II), we computed this average scenario by the weighted average of the 9 scenarios as described above.
- Mostly, we solve the integrated problems (in case I as well as case II) with the same parameter settings as in the previous chapter (see Section 4.5 for an extensive overview). However, sometimes we deviate from those settings. We will explicitly mention those deviations.

We applied the four heuristics described in Subsection 5.3.1 to the dynamic vehicle and crew scheduling. In Tables 5.12 and 5.13, the average results over all days are shown for prob_A and prob_B, respectively, for the cases I and II using the sequential algorithm (D-VSP-CSP) as well as the integrated one (D-VCSP). Hereby, the lower bound phase of the integrated approach is stopped after 11 iterations such that the computation time is reasonable low, which is necessary in a dynamic environment. Moreover, we use as period length, l, 120 minutes. In this table #V, #D, %L, DC, %DV and VC have the same meaning as before.

	I		II	
	sequential	integrated	sequential	integrated
#V	18.7	18.7	18.6	18.6
#D	43.9	37.8	40.2	37.2
%L	3.4	2.9	3.2	3.8
DC	2,986	1,798	2,814	3,351
%DV	6.1	5.6	4.5	5.4
VC	445	316	361	637

Table 5.12: Average results D-VCSP - prob_A

	I		II	
	sequential	integrated	sequential	integrated
#V	41	40.4	41.1	40.6
#D	85.6	76.5	84.9	76.4
%L	2.4	4.2	2.8	4.2
DC	3,509	6,586	4,173	7,633
%DV	3.9	4.8	5.2	5.3
VC	598	1,201	1,567	633

Table 5.13: Average results D-VCSP - prob_B

If we compare the different approaches the most important criteria are, of course, the actual costs, i.e. the number of vehicles and crews. If these results are similar, we take the other criteria into account. Therefore, we can immediately conclude that the integrated approach significantly outperforms the sequential approach. This is also what we expected beforehand, since applying an integrated approach leads in each iteration to an improvement of the sequential solution and therefore, the overall improvement is dramatic. Furthermore, we can see that the difference between case I and II (in the sequential as well as the integrated approach) is small. However, introducing more scenarios (case I) leads to a slightly better solution. Finally, we have to conclude that for prob_B even the best dynamic approach performs worse than the static integrated approach with fixed buffer times. Possible reasons for this are the stopping criteria in the dynamic version of the integrated approach and the way the problem is split up. We will investigate this question later on, but we will now focus on the results of the integrated approach (case I) for prob_A. In Table 5.14, we provide some more results for other values of the period length, while we give the total computation time (denoted by cpu), the number of iterations (it.) and the maximum computation time of one iteration over all instances (max.) in Table 5.15.

We can see that the results are better than in the static case with buffer times. Consider for example the case with 5 minutes buffer time, where we needed 24 vehicles and 37

l	#V	#D	%L	DC	%DV	VC
120	18.6	37.2	3.8	3,351	5.4	637
60	18.3	36.4	5.1	4,272	5.7	568

Table 5.14: Detailed results D-VCSP - case I integrated - prob_A

l	cpu	it.	max.
120	13,578	10	5,612
60	26,370	19	17,336

Table 5.15: Computation times D-VCSP - case I integrated - prob_A

drivers. In the dynamic approach, with l equal to 120, we can save more than 5 vehicles, while the number of drivers increases only slightly, the percentage of trips starting late is the same, and the percentage of violated duties and its costs are only slightly higher. Moreover, the delay costs are even slightly lower. Of course, we have assumed here that we know all travel times two hours ahead, in fact two hours before we start the calculation which also takes about 1.5 hour in the worst case. Therefore, the travel times are assumed to be known 3.5 hours before their realization. Due to this large computation time we cannot consider too small values of l. Furthermore, we can see that if the period length is 60 minutes, the results only change slightly. A few less vehicles and drivers are needed and there are slightly more delays. However, the maximum computation of one iteration is much larger than l, which means that we cannot use it anymore in practice. We can even see that this time is much larger than in the case where $l = 120$. This is a coincidence since the average computation of an iteration is almost equal in both cases (1,358 compared to 1,388 seconds).

To explain the bad performance of the dynamic approaches for prob_B, we also considered a variant of the D-VCSP without a maximum number of iterations (or indirectly computation time) in the lower bound phase of the integrated approach. That is the lower bound phase is only terminated when no duties with negative reduced costs have been found or the difference between the lower bound on the overall set of duties and the lower bound in a certain iteration is very small (less than 0.1%). For this purpose we only considered case I. Furthermore, we looked at the results for smaller values of l, the length of the period. The average results over all days are denoted in Table 5.16. The computation times for these cases are shown in Table 5.17. Notice hereby that the problem has been split into one subproblem for each depot. The abbreviations in both tables have the same meaning as before.

We can see that the effect of allowing more computation time is quite small. On average, one driver can be saved, but on the other criteria the performance is slightly worse. The maximum computation time per iteration and per depot is slightly less than one hour, which is only reasonable for large values of l. Finally, we have to conclude that

l	#V	#D	%L	DC	%DV	VC
120	40.5	75.4	4.4	7,101	6.4	1,761
60	40.5	76.5	5.0	8,809	5.1	1,692

Table 5.16: Improved results D-VCSP - case I integrated

	depot 1			depot 2			depot 3			depot 4		
l	cpu	it.	max.	cpu	it.	max.	cpu	it.	max.	cpu	it.	max.
120	1,861	8	513	2,378	10	364	10,374	10	2,690	240	7	48
60	1,949	14	335	2,763	19	267	16,093	19	2,716	410	13	48

Table 5.17: Computation times D-VCSP - case I integrated

if we also consider crew scheduling aspects, the dynamic approach is not as good as we expected beforehand. It does not perform well under all circumstances.

Lower bound Since the results of the dynamic approach are not completely satisfactory, we consider the situation where we have perfect information and compare several variants such that we can see where things go wrong. In the case with perfect information, there are obviously no trips starting late and thus also no delay costs and violated duties. Therefore, we just calculate the best feasible solution for each day with as input the realization of the travel times of that particular day. We do this again with the sequential and integrated approach. Recall from Chapter 4 that for such large problems no lower bounds on the optimal (integrated) solution can be found in a reasonable time. Therefore, we do not mention them here. The average of the best feasible solutions over all days can be found in Tables 5.18 and 5.19 for prob_A and prob_B, respectively.

	sequential	integrated
#V	19.4	19.4
#D	33.4	32.3

Table 5.18: Results in the case of perfect information - prob_A

From this table, we can conclude that the number of vehicles and drivers is of course slightly higher than in the static case without buffer times, however, the difference is very small. A remaining question is now whether the poor quality of the dynamic approach for prob_B can be explained by the fact that we use a cluster-reschedule heuristic, i.e. we first split the problem into several smaller problems for each depot, or by the performance of the heuristics, which we used to solve a single-depot problem. This question can be answered partly by looking at the results in the case of perfect information by using the same splitting of the problem into several subproblems. These average results over all days are denoted in Table 5.20 for the sequential as well as the integrated approach. We

	sequential	integrated
#V	40.8	40.8
#D	75.5	66.9

Table 5.19: Results in the case of perfect information - prob_B

can see that especially the difference for the integrated approach is quite large, namely a total cost saving of 7.8%. Therefore, we can conclude that the division of the problem already leads to a significant loss in the quality of the solution.

	sequential	integrated
#V	43.3	43.3
#D	77.7	73.5

Table 5.20: Results in the case of perfect information by first splitting the problem

Sensitivity analysis The assumption that the travel times are known (can be estimated without any error) at time point T for the period $[T, T + l)$ may be unrealistic for the considered values of l. Therefore, we have performed a sensitivity analysis by considering small deviations of the actual travel times from the estimated ones in the same way as in Subsection 5.2.4. We have only done this analysis for prob_A, since the dynamic approach did not perform well for prob_B even if the travel times were exactly known. Recall that by using slightly different simulated travel times than the estimated ones, only the performance w.r.t the number of trips starting late, the delay costs, the number of violated duties and the violation costs can change. The number of vehicles and drivers does not change.

Like in Subsection 5.2.4, we have again simulated four times 100 runs of actual travel times, where the actual travel time is drawn from a normal distribution with mean equal to the estimated travel time and variance σ^2. Furthermore, we still assume that the total delay is nonnegative. The results can be found in Table 5.21.

As can be easily seen from the table, the effects of small deviations from the estimated travel times are small. In fact, the effects of disturbances are much smaller than in Subsection 5.2.4, where we only considered vehicles. This means that a dynamic approach can be used to solve prob_A.

5.3.4 Discussion and Conclusion

In this section we considered the dynamic vehicle and crew scheduling. The extension of our approach to the dynamic vehicle scheduling problem to the situation where we also considered crews did not always give the results which we expected beforehand. For

l	σ	%L	DC	%DV	VC
120	2	11.1	4,534	6.4	687
120	1	6.9	3,652	5.7	652
120	0.5	5.0	3,467	5.5	646
120	0.25	4.0	3,394	5.4	644
60	2	10.7	5,434	6.7	609
60	1	7.7	4,740	6.4	582
60	0.5	6.6	4,475	6.3	580
60	0.25	6.6	4,417	6.2	591

Table 5.21: Average results with extra perturbation $\epsilon \sim N(0, \sigma^2)$ - prob_A

the small instance with a single depot the dynamic approach performs well. However, computation times are still high for applying such an approach in practice. On the other hand for the medium-sized instance with multiple depots, the traditional static approach with buffer times performed much better. We discussed several reasons why our dynamic approach does not perform so well in this case. The first reason is that this approach used the cluster-reschedule heuristic, i.e. all trips are assigned beforehand to a certain depot. In other words, the overall result is dependent on the chosen assignment. Another reason could be that the computation times allowed to solve the approach dynamically were set too small, although by extending these times the results did not improve so much. The final mentioned reason is that the idea of dynamically solving itself does not work so well. Since the dynamic approach worked well for the small instance where the data set was not divided into several smaller ones, and since extending the computation times did not lead to significant improvement, we can conclude that the way the problem is split up, is the bottleneck. Therefore, we recommend to invest further research in speeding up the suggested algorithms. With faster computers and better algorithms the dynamic approach should outperform the static one with buffer times for larger problem instances as well.

Finally, we would like to make several remarks about the practical applicability of such a dynamic approach. First of all, it will be difficult to test the assumptions we made in a practical environment. For instance, how can one measure if the travel times of the trips are really independent of the actual chosen schedule? Secondly, it is important how drivers (but also planners and managers) react on such a way of working. It is very easy for them to frustrate such an approach. Therefore, we conclude this discussion with the fact that there is still a long way to go before such an approach can be used in practice!

5.4 Application: Buffer Times in a Large Real-World Problem

In this section, we consider the effect of introducing buffer times in the large real-world problem of Connexxion described in Sections 4.6 and 5.2. Recall that this problem has 1,104 trips and 4 depots. Before we consider the introduction of buffer times, we first look at the (best) solution of this problem by using the sequential as well as the integrated approach. Since the problem is very large, we cannot solve it with an integrated approach at once and we have to divide it into several (small) subproblems. We use therefore one of the methods suggested in Section 4.6, namely method C. We divide the trips into six subproblems based on the optimal solution of the MDVSP. That is all trips assigned to the same vehicle schedule will be in one subproblem. Furthermore, the vehicles are assigned in consecutive order to the subproblems. Moreover, the maximum number of trips in a subproblem is equal to 200. Then each subproblem is solved with the integrated approach. However, the best feasible solutions of all subproblems are recombined, i.e. the vehicle schedules of those solutions are combined per depot and for each depot a CSP is solved. The results are given in Table 5.22. Furthermore, we give the distribution of the vehicles and drivers over the depots.

	sequential	integrated
vehicles	109 (9-8-37-55)	109 (5-9-35-60)
drivers	187 (20-15-62-88)	175 (9-17-56-93)

Table 5.22: Sequential and integrated solution

Since the integrated approach performs much better, we only calculate the percentage of trips starting late, its virtual costs and the percentage of violated duties and its costs (denoted again by %L, DC, %DV and VC, respectively) for the integrated solution. In Table 5.23 we give a detailed overview of these performance measures for each day.

From the table we can see that there is a large difference between the days. On days 5 and 10 (both corresponding to a Friday) there are much less delays and violated duties than on the other days. Moreover, we can see that the delay costs differ almost a factor 10 (day 2 compared with day 5), which is caused by a few large delays. A similar remark holds for the violation costs.

In the previous section, we concluded that the way the problem is divided into different subproblems, is very important. Therefore, we consider now two different ways to introduce the buffer times. The first possibility we consider, is the one where the problem is split up by taking the buffer times into account in the division. That is, we split up the problem again by the same method but now with the buffer times in the data. Moreover, as alternative we look at the option of keeping the same division and introducing the

day	1	2	3	4	5	
%L	18.6	17.9	19.6	18.4	11.4	
DC	99,260	319,050	132,330	108,390	36,610	
%DV	9.9	16.5	13.5	14.2	6.2	
VC	4,783	8,962	5,039	4,070	669	

day	6	7	8	9	10	av.
%L	16.4	12.7	15.4	16.5	11.3	15.8
DC	141,720	83,860	71,100	136,660	47,710	117,669
%DV	12.8	11.7	10.1	7.7	7.5	11.0
VC	5,959	4,364	1,952	1,700	690	3,819

Table 5.23: Results when using the optimal solution of the S-VCSP

buffer times when solving the subproblems. We will now consider the first option. In Table 5.24 the results of the sequential and integrated approach can be found when we introduce buffer times and divide the subproblems again. We can see that the number of drivers increases more than the number of vehicles by introducing buffer times. The effect on the delays is similar to the situation where we only considered vehicles (see Subsection 5.2.4). Furthermore, introducing buffer times leads to less violated duties (and corresponding costs), although, the effect is not that large.

We will now look at the second option, where the problem is divided beforehand. Since splitting has only effect on the outcomes of the integrated approach, we will only mention those results in Table 5.25. Moreover, we mention in the last line of the table, the percentage of trips that moved from one subproblem to another one if the trips would be redivided in the same way as before.

buffer	no		2 min.		5 min.		7 min.		10 min.	
	seq.	int.	seq.	int.	seq.	int.	seq.	int.	seq.	int.
#V	109	109	112	112	117	117	120	120	122	122
#D	187	175	202	190	214	207	217	204	218	210
%L	16.8	15.8	7.9	6.8	3.8	3.4	2.8	2.7	1.9	1.9
DC	105,441	117,669	96,036	62,155	33,566	30,077	27,258	20,685	15,245	21,191
%DV	10.4	11.0	6.0	9.8	5.0	6.9	4.9	7.4	4.1	5.2
VC	5,980	3,819	6,201	10,909	1,109	3,901	3,048	11,116	3,344	10,220

Table 5.24: Average results of introducing buffer times

buffer	2 min.	5 min.	7 min.	10 min.
#V	123	131	129	144
#D	213	231	223	264
moves	54.6%	59.1%	62.1%	58.1%

Table 5.25: Results integrated approach with original division

From the table we can see that a different division of the trips has a dramatic effect on the final results. Not taking the buffer times into account during the splitting of the trips leads to 11 vehicles and 23 drivers more, in case of 2 minutes buffer times and even more in the other cases. This can be explained by the fact that more than half of the trips moves from one subproblem to another one, which means that the division is very sensitive to small changes in the data. Moreover, another division leads to a completely different solution. Therefore, we can reconfirm our conclusion that it is very important which division is chosen. For this particular problem, the best way is to choose the division such that the number of vehicles is minimal per definition. Indirectly, this leads to a close to optimal number of drivers too.

5.5 Summary

In this chapter we considered a dynamic approach for vehicle and crew scheduling. First, we only looked at the vehicle scheduling problem and we developed a very fast method to solve it dynamically. The results showed that such a dynamic approach outperformed the traditional, static approach with buffer times significantly. Therefore, we extended it to the crew scheduling problem and the integrated vehicle and crew scheduling problem. Due to the complexity of the integrated approach, only small and medium-sized instances were solved with such a dynamic approach. We showed that the results are good in the first case. However, we got into trouble when we solved larger instances. In those cases the static approach with buffer times performed better. This was especially caused by the splitting of the problem into several subproblems, which was necessary because of the size of the problem. Finally, we considered a large real-world instance and we discussed some effects on the outcome by introducing buffer times and several, different divisions of the problem into smaller subproblems.

Appendix: Results Dynamic Vehicle Scheduling

In the tables in this appendix the following abbreviations are used:

- BT: buffer time in minutes (0 denotes no buffer time);
- #V: the number of vehicles used;
- %L: the percentage of trips starting late;
- DC: the total delay costs.

	Day 1						Day 2					
	I			II			I			II		
l	#V	%L	DC	#V	%L	DC	#V	%L	DC	#V	%L	DC
120	116	0.3	660	116	1.0	1,120	112	0.6	1,200	113	1.6	67,890
60	118	0.8	2,760	117	1.5	3,380	113	0.4	1,720	114	1.2	28,390
30	117	1.7	12,790	116	2.5	8,560	112	1.0	2,980	115	1.6	29,120
15	116	2.3	16,070	116	3.3	10,610	112	1.4	26,930	115	3.4	94,920
10	116	3.0	13,770	116	4.2	10,020	113	0.9	6,790	115	3.0	39,330
5	116	3.5	17,590	116	5.1	15,440	114	1.2	8,770	116	3.6	46,830
1	117	3.5	17,870	116	5.6	17,660	114	1.9	8,420	116	4.1	36,750
	Day 3						Day 4					
	I			II			I			II		
l	#V	%L	DC	#V	%L	DC	#V	%L	DC	#V	%L	DC
120	112	0.9	8,250	112	1.3	8,320	114	0.8	4,210	114	1.6	5,680
60	114	1.4	1,890	116	1.9	2,710	115	1.0	4,160	114	2.4	6,650
30	113	0.8	1,120	116	2.3	3,310	115	1.9	6,010	116	3.0	6,820
15	113	1.5	6,440	116	4.4	14,090	114	2.1	8,300	116	4.0	7,180
10	112	2.4	7,700	116	5.5	20,970	115	2.5	5,660	116	4.1	8,050
5	114	2.5	14,740	116	5.9	23,950	114	2.5	7,950	117	4.3	7,170
1	113	2.6	19,780	116	6.3	26,140	115	3.0	9,250	116	5.4	17,700

Table 5.26: Detailed results D-VSP - day 1-4

	Day 5						Day 6					
	I			II			I			II		
l	#V	%L	DC	#V	%L	DC	#V	%L	DC	#V	%L	DC
120	113	0.5	1,260	114	0.6	1,310	113	1.0	260	114	0.5	690
60	114	0.4	850	114	0.7	1,580	113	1.0	260	115	1.0	5,200
30	114	0.7	2,520	114	1.4	5,570	113	0.9	740	115	1.6	5,480
15	114	1.4	7,900	115	2.5	11,100	113	1.5	10,190	116	3.0	16,990
10	114	1.6	8,210	116	3.1	16,460	113	1.7	11,280	116	3.4	19,440
5	114	1.5	6,990	116	3.4	18,400	114	1.2	4,190	116	3.9	15,720
1	113	2.0	8,400	116	3.5	21,580	114	1.4	7,280	116	4.6	18,550
	Day 7						Day 8					
	I			II			I			II		
l	#V	%L	DC	#V	%L	DC	#V	%L	DC	#V	%L	DC
120	112	0.5	610	113	0.5	550	114	0.5	140	114	0.8	630
60	114	0.5	520	114	1.5	2,580	114	0.8	2,610	115	2.2	4,970
30	114	0.6	3,900	115	2.1	13,670	114	1.1	3,310	116	3.3	6,960
15	113	1.2	5,060	115	3.4	12,950	113	3.2	6,070	116	6.0	20,220
10	114	0.9	3,770	116	3.3	11,700	114	3.1	6,780	116	6.3	27,890
5	113	1.4	5,350	115	4.0	11,590	114	3.4	7,550	116	7.5	30,950
1	114	1.0	5,730	114	4.3	18,740	114	3.5	7,960	116	8.2	27,460
	Day 9						Day 10					
	I			II			I			II		
l	#V	%L	DC	#V	%L	DC	#V	%L	DC	#V	%L	DC
120	114	0.0	0	115	0.1	10	112	0.2	130	113	0.4	100
60	114	0.2	170	115	0.7	830	113	1.2	3,330	115	0.9	3,520
30	113	0.8	1,250	115	2.1	25,820	113	0.7	2,520	115	1.7	4,400
15	113	1.4	2,020	116	3.3	58,860	114	1.3	3,260	116	2.6	6,730
10	114	1.1	1,600	116	3.4	65,380	113	1.0	2,790	116	2.4	6,530
5	113	1.4	2,330	117	4.0	62,650	114	1.2	2,420	116	2.6	6,690
1	114	1.6	2,680	117	4.9	65,770	114	0.9	2,230	116	2.8	6,810

Table 5.27: Detailed results D-VSP - day 5-10

BT	day	1	2	3	4	5	6	7	8	9	10	av.
0	%L	20.5	19.2	21.9	18.1	12.3	16.9	14.2	20.3	17.0	11.7	17.2
	DC	134,940	146,350	116,290	89,490	51,080	107,550	88,520	112,870	171,660	59,550	107,830
2	%L	8.9	9.6	8.3	8.8	5.0	7.9	6.2	7.8	8.3	5.2	7.6
	DC	69,950	85,420	47,710	56,220	25,540	138,680	34,520	27,280	55,730	14,690	55,174
5	%L	3.7	4.4	4.2	4.1	1.5	4.2	2.5	2.9	3.6	1.7	3.3
	DC	18,800	58,860	25,030	23,540	3,280	93,650	11,420	8,220	36,200	5,330	28,424
7	%L	2.3	3.3	2.0	2.7	0.6	2.7	1.6	2.1	2.4	1.2	2.1
	DC	7,340	78,180	14,950	17,750	2,170	14,790	7,870	6,400	32,300	5,260	18,683
10	%L	1.7	2.4	1.2	1.3	0.2	1.5	1.2	1.1	1.3	0.6	1.2
	DC	8,180	86,500	9,090	7,420	260	6,460	7,220	3,260	12,940	1,210	14,254

Table 5.28: Detailed results of introducing fixed buffer times

Chapter 6

Summary and Concluding Remarks

In this thesis we considered integrated and dynamic variants of the vehicle and crew scheduling problem. During the last couple of years there has been an increasing attention to integration of vehicle and crew scheduling, since it has been shown that the solution of the sequential approach, i.e. solving first the vehicle scheduling problem and, given that solution, the crew scheduling problem, can be improved upon significantly.

In fact, in Chapters 3 and 4 of this thesis, we deepened this analysis and we showed that the integrated approach can also be applied to solve instances with more than one depot and in some real-world applications. Therefore, we developed several models and algorithms, which are extensions and/or improvements of the models and algorithms suggested by Freling (1997) for the single-depot integrated vehicle and crew scheduling problem. Those algorithms combine Lagrangian relaxation and column generation. Furthermore, we used Lagrangian heuristics to generate feasible solutions. We showed that the gaps between those solutions and the lower bounds on the optimal solutions differ under different characteristics of the instances. For example, those gaps vary between 0 and 4% for the single-depot instances from the RET with changeovers allowed, while they vary between 5 and 10% for the real-world instances from Connexxion with multiple-depots. Moreover, we showed that the size of the improvements of an integrated approach compared with a sequential one is also dependent of the structure of the instance. For example, we showed that the relative savings of an integrated approach are much larger in the case of multiple depots than in the case of only one single depot. Moreover, we showed that in the case of a single depot the saving is larger when changeovers are not allowed than in the case where they are allowed. Finally, we showed that such an integrated approach can be used to solve real-world problem instances with exotic labor constraints, which have not been considered in the Operations Research literature before.

In Chapter 5, we looked at the executions of the planned schedules at the day of operation. If predetermined schedules are executed on a certain day, a lot of things can go wrong. Due to a delay at a certain moment, other trips assigned to the same vehicle can start late too. We showed that there are different ways to prevent such delays. The

first one, which is the one used in practice nowadays, is to introduce buffer times such that there are a few minutes of buffer time between every pair of trips. If the delay is smaller than the buffer time, the next trip can be executed on time, for sure. However, with larger delays this cannot be guaranteed anymore. Another disadvantage of buffer times is that the costs (i.e. required number of vehicles and crews) increase a lot. To overcome these difficulties, we suggested another approach, namely dynamic scheduling. Firstly, we applied this approach to the vehicle scheduling problem, i.e. we did not take the drivers into account. We showed that the dynamic approach performs very well in this case. The results were much better than in the static approach with buffer times. Furthermore, the computation times were quite low, which means that such an approach can be applied in practice.

Based on the success of those ideas we extended the approach to vehicle and crew scheduling. That is, we developed several approaches to solve the dynamic vehicle and crew scheduling problem, sequentially as well as integrated. As could be expected from the earlier chapters, the integrated approach significantly outperformed the sequential one. Furthermore, we showed that taking multiple scenarios for the future travel times into account instead of only looking at the average travel time, made sense. However, such a dynamic approach only performed well to solve small instances. Due to the computational complexity large or medium-sized instances could only be solved by splitting the instances into several smaller ones and by solving them separately with a dynamic approach.

In Subsection 1.1.2 we discussed some differences in the planning problems between bus and rail transport, and between bus and airline transport. The main ideas for the integrated vehicle and crew scheduling problem as proposed in this thesis, can also be applied to the other modes of transport. However, there are some differences in the details. Especially, in the railway world, the counterpart of the vehicle scheduling problem is much more complicated. Since we solve the underlying vehicle scheduling problem many times in the suggested algorithms for the integrated approach, those algorithms cannot be applied directly to railway problems. Some modifications are necessary. Since the differences between bus and airline scheduling are much smaller, the suggest algorithms can be applied with only minor modifications in the airline world. In fact, slightly different algorithms have been proposed in the literature to solve the integrated aircraft routing and crew scheduling problem (see e.g. Klabjan *et al.* (2002)).

Although, the integrated approach could be applied to other modes of transport, this is much less obvious for the proposed dynamic approaches. Of course, the main idea to schedule dynamically can also be used for railways and airlines, but if it results in the same effect is questionable. For example, in the railway world, one train can delay another one by blocking the track, which can never happen with buses. Another important aspect is that the vehicles and crews should be flexibly usable, which is not likely to be the case in the railway as well as the airline context. The reader can think of e.g. different types

of trains/planes and different properties of the drivers/pilots. Therefore, our conclusions on the dynamic approach cannot be generalized directly to other modes of transport. Introducing our ideas in these environments would be an interesting subject for future research.

If we go back to the three purposes of the thesis (see Subsection 1.5), we can conclude that all of them are fulfilled. We showed (1) in which circumstances an integrated approach gives the largest improvements over the sequential one, (2) that the integrated approach can be used to solve practical problem instances and (3) that the service to the passengers can be improved at equal costs. However, several assumptions have been made. Therefore, there can be quite some practical situations, which require more research.

Finally, we would like to provide some suggestions for future research in this area. Since there is some gap between the lower bounds we found and our best solutions, the quality of the algorithms to solve the integrated vehicle and crew scheduling problem can potentially be improved. This gap can be closed by using an exact (e.g. branch-and-price) algorithm instead of our Lagrangian heuristics. Furthermore, we suggest to continue research on faster algorithms to solve the dynamic integrated vehicle and crew scheduling problem, since the dynamic approach was only able to solve small problem instances due to the large computation time. One of the ways to do this is, is to speed up the algorithms for the (static) integrated vehicle and crew scheduling problem. This has of course also advantages if this problem is considered by itself. Moreover, we suggest to test such a dynamic approach in a real-world environment such that our assumptions can be checked and to see if such an approach also works in practice. Especially, the reactions of drivers can make it or break it!

Definitions

In this appendix we provide an overview of some important definitions, which have been used throughout this thesis.

depot	parking for vehicles that are not in use for some time
trip	movement with passengers between specified start and end location at a specified departure and arrival time
deadhead	movement in time and/or space between two trips, from a depot to the first trip and from the last trip to a depot
compatible trips	trips that can be executed by the same vehicle after each other
long arc	arc between two consecutive trips, which is long enough to let the vehicle return to the depot
short arc	arc between two consecutive trips, which is not long enough to let the vehicle return to the depot
(vehicle) block	sequence of trips and deadheads starting and ending in the depot, which can be executed by one vehicle
relief points	location and time where and when a change of crew may occur
task	part of a vehicle block between two consecutive relief points
piece (of work)	sequence of tasks on one vehicle block without a break that can be performed by a single crew member without interruption
(crew) duty	sequence of tasks assigned to the same crew member

Bibliography

Ahuja, R.K., Magnanti, T.L., & Orlin, J.B. 1993. *Network Flows, Theory, Algorithms, and Applications*. Prentice Hall, New Jersey.

Ball, M., Bodin, L., & Dial, R. 1983. A Matching Based Heuristic for Scheduling Mass Transit Crews and Vehicles. *Transportation Science*, **17**, 4–31.

Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., & Vance, P.H. 1998. Branch-and-Price: Column Generation for Solving Huge Integer Programs. *Operations Research*, **46**, 316–329.

Beasley, J.E. 1995. Lagrangean Relaxation. *Pages 243–303 of:* Reeves, C.R. (ed), *Modern Heuristic Techniques for Combinatorial Problems*. McGraw-Hill, London.

Bertossi, A.A., Carraresi, P., & Gallo, G. 1987. On Some Matching Problems Arising in Vehicle Scheduling Models. *Networks*, **17**, 271–281.

Bertsekas, D.P., & Castañon, D.A. 1992. A Forward/Reverse Auction Algorithm for Assymmetric Assignment Problems. *Computational Optimization and Applications*, **1**, 277–297.

Bodin, L., Golden, B., Assad, A., & Ball, M. 1983. Routing and Scheduling of Vehicles and Crews: The State of the Art. *Computers and Operations Research*, **10**, 63–211.

Borndörfer, R., Löbel, A., & Weider, S. 2002. *Integrierte Umlauf- und Dienstplanung im Nahverkehr*. Tech. rept. 02-10. ZIB, Berlin.

Caprara, A., Fischetti, M., & Toth, P. 1999a. A Heuristic Algorithm for the Set Covering Problem. *Operations Research*, **47**, 730–743.

Caprara, A., Fischetti, M., Guida, P.L., Toth, P., & Vigo, D. 1999b. Solution of Large-Scale Railway Crew Planning Problems: the Italian Experience. *Pages 1–18 of:* Wilson, N.H.M. (ed), *Computer-Aided Transit Scheduling*. Springer Verlag, Berlin.

Carpenato, G., Dell'Amico, M., Fischetti, M., & Toth, P. 1989. A Branch and Bound Algorithm for the Multiple Depot Vehicle Scheduling Problem. *Networks*, **19**, 531–548.

Carraresi, P., Girardi, L., & Nonato, M. 1995. Network Models, Lagrangean Relaxation and Subgradients Bundle Approach in Crew Scheduling Problems. *Pages 188–212 of:* Daduna, J.R., Branco, I., & Paixão, J.M. Pinto (eds), *Computer-Aided Transit Scheduling, Proceedings of the Sixth International Workshop*. Springer Verlag, Berlin.

Chvátal, V. 1983. *Linear Programming*. W.H. Freeman and Company, New York.

Cordeau, J-F., Stojković, G., Soumis, F., & Desrosiers, J. 2001. Benders Decomposition for Simultaneous Aircraft Routing and Crew Scheduling. *Transportation Science*, **35**, 375–388.

Daduna, J.R., Mojsilovic, M., & Schütze, P. 1993. Practical Experiences Using an Interactive Optimization Procedure for Vehicle Scheduling. *Pages 37–52 of:* Du, D.-Z., & Pardalos, P.M. (eds), *Network Optimization Problems: Algorithms, Applications and Complexity*. World Scientific, Singapore.

Dantzig, G.B., & Wolfe, P. 1960. Decomposition principles for linear programming. *Operations Research*, **8**, 101–111.

Darby-Dowman, K., Jachnik, J.K., Lewis, R.L., & Mitra, G. 1988. Integrated Decision Support Systems for Urban Transport Scheduling: Discussion of Implementation and Experience. *Pages 226–239 of:* Daduna, J.R., & Wren, A. (eds), *Computer-Aided Transit Scheduling: Proceedings of the Fourth International Workshop*. Springer Verlag, Berlin.

De Groot, S. 2003. *Een Geïntegreerde Aanpak van Voertuig- en Personeelsplanning Toegepast op Grote Probleeminstanties (in Dutch)*. Master thesis, School of Economics, Erasmus University Rotterdam.

Dell'Amico, M., Fischetti, M., & Toth, P. 1993. Heuristic Algorithms for the Multiple Depot Vehicle Scheduling Problem. *Management Science*, **39**, 115–125.

Desaulniers, G., Desrosiers, J., Dumas, Y., Marc, S., Rioux, B., Solomon, M.M., & Soumis, F. 1997. Crew Pairing at Air France. *European Journal of Operational Research*, **97**, 245–259.

Desaulniers, G., Desrosiers, J., Lasry, A., & Solomon, M.M. 1999. Crew Pairing for a Regional Carrier. *Pages 19–41 of:* Wilson, N.H.M. (ed), *Computer-Aided Transit Scheduling*. Springer Verlag, Berlin.

Desaulniers, G., Cordeau, J-F., & Desrosiers, J. 2001. *Simultaneous Multi-Depot Bus and Driver Scheduling*. TRISTAN IV preprints.

Desrochers, M., & Soumis, F. 1989. A Column Generation Approach to the Urban Transit Crew Scheduling Problem. *Transportation Science*, **23**, 1–13.

Desrochers, M., Gilbert, J., Sauve, M., & Soumis, F. 1992a. Crew-Opt: Subproblem Modeling in a Column Generation Approach to the Urban Crew Scheduling Problem. *Pages 395–405 of:* Desrochers, M., & Rousseau, J.M. (eds), *Computer-Aided Transit Scheduling: Proceedings of the Fifth International Workshop.* Springer Verlag, Berlin.

Desrochers, M., Desrosiers, J., & Solomon, M. 1992b. A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. *Operations Research*, **40**, 342–354.

Desrosiers, J., Soumis, F., & Desrochers, M. 1984. Routing with Time Windows by Column Generation. *Networks*, **14**, 545–565.

Desrosiers, J., Dumas, Y., Solomon, M.M., & Soumis, F. 1995. Time Constrained Routing and Scheduling. *Pages 35–139 of:* Ball, M.O., Magnanti, T.L., Monma, C.L., & Nemhauser, G.L. (eds), *Network Routing.* Handbooks in Operations Research and Management Science, vol. 8. North-Holland.

Dias, T. Galvão, Ferreira, J. Vasconcelos, & Cunha, J.F. 2001. Evaluating a DSS for Operational Planning in Public Transport Systems: Ten Years of Experience with the GIST System. *Pages 167–179 of:* Voß, S., & Daduna, J.R. (eds), *Computer-Aided Scheduling of Public Transport.* Springer, Berlin.

Falkner, J.C., & Ryan, D.M. 1992. EXPRESS: Set Partitioning for Bus Crew Scheduling in Christchurch. *Pages 359–378 of:* Desrochers, M., & Rousseau, J.M. (eds), *Computer-Aided Scheduling: Proceedings of the Fifth International Workshop.* Springer Verlag, Berlin.

Fischetti, M., Martello, S., & Toth, P. 1987. The Fixed Job Schedule Problem with Spread-Time Constraints. *Operations Research*, **35**, 849–858.

Fischetti, M., Martello, S., & Toth, P. 1989. The Fixed Job Schedule Problem with Working-Time Constraints. *Operations Research*, **37**, 395–403.

Fischetti, M., Lodi, A., Martello, S., & Toth, P. 2001. A Polyhedral Approach to Simplified Crew and Vehicle Scheduling Problems. *Management Science*, **47**, 833–850.

Fisher, M.L. 1981. The Lagrangian Relaxation Method for Solving Integer Programming Problems. *Management Science*, **27**, 1–18.

Forbes, M.A., Holt, J.N., & Watts, A.M. 1994. An Exact Algorithm for Multiple Depot Bus Scheduling. *European Journal of Operational Research*, **72**, 115–124.

Fores, S., Proll, L., & Wren, A. 2001. Experiences with a Flexible Driver Scheduler. *Pages 137–152 of:* Voß, S., & Daduna, J.R. (eds), *Computer-Aided Scheduling of Public Transport.* Springer, Berlin.

Freling, R. 1997. *Models and Techniques for Integrating Vehicle and Crew Scheduling.* Ph.D. thesis, Tinbergen Institute, Erasmus University Rotterdam.

Freling, R., Boender, C.G.E, & Paixão, J.M. Pinto. 1995. *An Integrated Approach to Vehicle and Crew Scheduling.* Tech. rept. 9503/A. Econometric Institute, Erasmus University Rotterdam, Rotterdam.

Freling, R., Wagelmans, A.P.M., & Paixão, J.M. Pinto. 1999. An Overview of Models and Techniques for Integrating Vehicle and Crew Scheduling. *Pages 441–460 of:* Wilson, N.H.M. (ed), *Computer-Aided Transit Scheduling.* Springer Verlag, Berlin.

Freling, R., Huisman, D., & Wagelmans, A.P.M. 2001a. Applying an Integrated Approach to Vehicle and Crew Scheduling in Practice. *Pages 73–90 of:* Voß, S., & Daduna, J.R. (eds), *Computer-Aided Scheduling of Public Transport.* Springer, Berlin.

Freling, R., Paixão, J.M. Pinto, & Wagelmans, A.P.M. 2001b. Models and Algorithms for Single-Depot Vehicle Scheduling. *Transportation Science*, **35**, 165–180.

Freling, R., Lentink, R.M., & Odijk, M.A. 2001c. Scheduling Train Crews: A Case Study for the Dutch Railways. *Pages 153–165 of:* Voß, S., & Daduna, J.R. (eds), *Computer-Aided Scheduling of Public Transport.* Springer, Berlin.

Freling, R., Lentink, R.M., Kroon, L.G., & Huisman, D. 2002. *Shunting of Passenger Train Units in a Railway Station.* Tech. rept. EI2002-26. Econometric Institute, Erasmus University Rotterdam, Rotterdam. To appear in *Transportation Science.*

Freling, R., Huisman, D., & Wagelmans, A.P.M. 2003. Models and Algorithms for Integration of Vehicle and Crew Scheduling. *Journal of Scheduling*, **6**, 63–85.

Gaffi, A., & Nonato, M. 1999. An Integrated Approach to Extra-Urban Crew and Vehicle Scheduling. *Pages 103–128 of:* Wilson, N.H.M. (ed), *Computer-Aided Transit Scheduling.* Springer Verlag, Berlin.

Garey, M.R., & Johnson, D.S. 1979. *Computers and Intractability: a Guide to the Theory of NP-Completeness.* Freeman, San Francisco.

Geoffrion, A.M. 1974. Lagrangean Relaxation for Integer Programming. *Mathematical Programming Study*, **2**, 82–114.

Haase, K., & Friberg, C. 1999. An Exact Branch and Cut Algorithm for the Vehicle and Crew Scheduling Problem. *Pages 63–80 of:* Wilson, N.H.M. (ed), *Computer-Aided Transit Scheduling.* Springer Verlag, Berlin.

Haase, K., Desaulniers, G., & Desrosiers, J. 2001. Simultaneous Vehicle and Crew Scheduling in Urban Mass Transit Systems. *Transportation Science*, **35**, 286–303.

Hadjar, A., Marcotte, O., & Soumis, F. 2001. *A Branch-and-Cut Algorithm for the Multiple Depot Vehicle Scheduling Problem.* Tech. rept. G-2001-25. Les Cahiers du Gerad, Montréal.

Haghani, A., Banihashemi, M., & Chiang, K-H. 2003. A Comparative Analysis of Bus Transit Vehicle Scheduling Models. *Transportation Research Part B*, **37**, 301–322.

Held, M., & Karp, R.M. 1971. The Traveling Salesman Problem and Minimum Spanning Trees: part II. *Mathematical Programming*, **1**, 6–25.

Hoffman, K.L., & Padberg, M. 1993. Solving Airline Crew Scheduling Problems by Branch-and-Cut. *Management Science*, **39**, 657–682.

Horner, P. 2002. How Continental Landed on its Fleet. *OR/MS Today*, **29**, 30–31.

Huisman, D. 1999. *Sequentiële en Geïntegreerde Benaderingen voor Voertuig- en Personeelsplanning bij de RET (in Dutch).* Master thesis, School of Economics, Erasmus University Rotterdam.

Huisman, D., Freling, R., & Wagelmans, A.P.M. 2001. *A Dynamic Approach to Vehicle Scheduling.* Tech. rept. EI2001-17. Econometric Institute, Erasmus University Rotterdam, Rotterdam. To appear in *Transportation Science* under the title "A Robust Solution Approach to the Dynamic Vehicle Scheduling Problem".

Huisman, D., Freling, R., & Wagelmans, A.P.M. 2003. *Multiple-Depot Integrated Vehicle and Crew Scheduling.* Tech. rept. EI2003-02. Econometric Institute, Erasmus University Rotterdam, Rotterdam. To appear in *Transportation Science.*

Ichoua, S., Gendreau, M., & Potvin, J.Y. 2000. Diversion Issues in Real-Time Vehicle Dispatching. *Transportation Science*, **34**, 426–438.

Klabjan, D., Johnson, E.L., Nemhauser, G.L., Gelman, E., & Ramaswamy, S. 2002. Airline Crew Scheduling with Time Windows and Plane-Count Constraints. *Transportation Science*, **36**, 337–348.

Kroon, L., & Fischetti, M. 2001. Crew Scheduling for Netherlands Railways "Destination: Customer". *Pages 181–201 of:* Voß, S., & Daduna, J.R. (eds), *Computer-Aided Scheduling of Public Transport.* Springer, Berlin.

Laporte, G., & Louveaux, F.V. 1998. Solving Stochastic Routing Problems with the Integer L-Shaped Method. *Pages 159–167 of:* Crainic, T.G., & Laporte, G. (eds), *Fleet Management and Logistics.* Kluwer.

Lenstra, J.K., & Rinnooy Kan, A.H.G. 1979. Computational Complexity of Discrete Optimization Problems. *Annals of Discrete Mathematics*, **4**, 121–140.

Löbel, A. 1997. *Optimal Vehicle Scheduling in Public Transit.* Ph.D. thesis, Technische Universität Berlin.

Löbel, A. 1998. Vehicle Scheduling in Public Transit and Lagrangean Pricing. *Management Science*, **44**, 1637–1649.

Lübbecke, M.E., & Desrosiers, J. 2002. *Selected Topics in Column Generation.* Tech. rept. G-2002-64. Les Cahiers du Gerad, Montréal.

Madsen, O.B.G., Ravn, H.F., & Rygaard, J.M. 1995. A Heuristic Algorithm for a Dial-a-Ride Problem with Time Windows, Multiple Capacities and Multiple Objectives. *Annals of Operations Research*, **60**, 193–208.

Mesquita, M., & Paixão, J. 1999. Exact Algorithms for the Multiple-Depot Vehicle Scheduling Problem Based on Multicommodity Network Flow Type Formulations. *Pages 223–246 of:* Wilson, N.H.M. (ed), *Computer-Aided Transit Scheduling.* Springer Verlag, Berlin.

Mingozzi, A., Boschetti, M.A., Ricciardelli, S., & Bianco, L. 1999. A Set Partitioning Approach to the Crew Scheduling Problem. *Operations Research*, **47**, 873–888.

Patrikalakis, I., & Xerocostas, D. 1992. A New Decomposition Scheme of the Urban Public Transport Scheduling Problem. *Pages 407–425 of:* Desrochers, M., & Rousseau, J.M. (eds), *Computer-Aided Transit Scheduling: Proceedings of the Fifth International Workshop.* Springer Verlag, Berlin.

Powell, W.B., Jaillet, P., & Odoni, A. 1995. Stochastic and Dynamic Networks and Routing. *Pages 141–295 of:* M. O. Ball, T. L. Magnanti, C. L. Monma, & Nemhauser, G. L. (eds), *Network Routing.* Handbooks in Operations Research and Management Science, vol. 8. North-Holland.

Powell, W.B., Towns, M.T., & Marar, A. 2000. On the Value of Optimal Myopic Solutions for Dynamic Routing and Scheduling Problems in the Presence of Noncompliance. *Transportation Science*, **34**, 67–85.

Ribeiro, C.C., & Soumis, F. 1994. A Column Generation Approach to the Multiple-Depot Vehicle Scheduling Problem. *Operations Research*, **42**, 41–52.

Rousseau, J.M., & Blais, J.Y. 1985. HASTUS: An Interactive System for Buses and Crew Scheduling. *Pages 45–60 of:* Rousseau, J.M. (ed), *Computer Scheduling of Public Transport 2.* North Holland, Amsterdam.

Schaefer, A.J., Johnson, E.L., Kleywegt, A.J., & Nemhauser, G.L. 2001. *Airline Crew Scheduling under Uncertainty.* Tech. rept. Georgia Institute of Technology, Atlanta, Georgia.

Scott, D. 1985. A Large Linear Programming Approach to the Public Transport Scheduling and Cost Model. *Pages 473–491 of:* Rousseau, J.M. (ed), *Computer Scheduling of Public Transport 2*. North Holland, Amsterdam.

Sleator, D.D., & Tarjan, R.E. 1985. Amortized Efficiency of List Update and Paging Rules. *Com. ACM*, **28**, 202–208.

Stojković, M., & Soumis, F. 2001. An Optimization Model for the Simultaneous Operational Flight and Pilot Scheduling Problem. *Management Science*, **47**, 1290–1305.

Tosini, E., & Vercellis, C. 1988. An Interactive System for Extra-Urban Vehicle and Crew Scheduling Problems. *Pages 41–53 of:* Daduna, J.R., & Wren, A. (eds), *Computer-Aided Transit Scheduling: Proceedings of the Fourth International Workshop*. Springer Verlag, Berlin.

Wolsey, L.A. 1998. *Integer Programming*. Wiley, New York.

Yen, J.W., & Birge, J.R. 2000. *A Stochastic Programming Approach to the Airline Crew Scheduling Problem*. World Wide Web, http://faculty.washington.edu/joyceyen.

Nederlandse Samenvatting

Het planningsproces van een openbaar vervoerbedrijf bestaat in het algemeen uit vier stappen: het maken van de dienstregeling, het maken van de voertuigdiensten, het maken van de diensten voor het personeel en het maken van de roosters voor het personeel. In dit proefschrift zijn twee van deze processen bekeken: voertuig- en personeelsplanning. Merk daarbij op dat we onder deze laatste term alleen het maken van diensten voor het personeel verstaan. Voertuigplanning kan kort worden samengevat als het toewijzen van ritten aan voertuigen, zodanig dat de totale voertuigkosten geminimaliseerd worden. Het resultaat is een verzameling voertuigdiensten. De personeelsplanning kan op een soortgelijke manier worden beschreven met dien verstande dat nu taken worden toegewezen aan personeelsdiensten. Met een taak wordt hier een rit of een combinatie van enkele ritten bedoeld. Daarnaast begint en eindigt een taak altijd op een zogenaamd aflospunt, waar een pauze kan plaatsvinden of het betreffende personeelslid afgelost kan worden. Het grote verschil tussen deze twee planningsprocessen bestaat uit het feit dat er voor een personeelsdienst allerlei extra eisen gelden die er niet zijn voor een voertuigdienst. Hierbij kan gedacht worden aan diverse wetten ten aanzien van rust- en rijtijden, CAO afspraken en bedrijfsregels. Zo is er bijna altijd sprake van een maximale werktijd, een maximale dienstlengte en een minimale hoeveelheid aan pauze(s). In dit proefschrift heeft de focus gelegen op het plannen van bussen en buschauffeurs. De meeste modellen en algoritmen kunnen echter ook - met enkele aanpassingen - gebruikt worden bij het plannen van andere vervoersmodaliteiten zoals treinen en vliegtuigen. Overeenkomsten en verschillen in de planningsproblematiek van deze modaliteiten zijn besproken in hoofdstuk 1.

In hoofdstuk 2 van dit proefschrift hebben we het traditionele planningsproces bekeken, waarbij eerst het voertuigplanningsprobleem wordt opgelost en vervolgens, gegeven die oplossing, de personeelsplanning wordt bepaald. In dit hoofdstuk staat ook een toepassing beschreven van het door ons ontwikkelde systeem voor personeelsplanning, DOPT. Dit systeem is momenteel in gebruik bij Stadsbus Maastricht. Zo'n sequentiële aanpak kan echter tot suboptimale oplossingen leiden. Dit houdt in dat ondanks het feit dat beide problemen ieder voor zich optimaal opgelost zijn, de uiteindelijke totale oplossing toch nog verbeterd kan worden. Daarom hebben we in de hoofdstukken 3 en 4 van dit proefschrift gekeken naar geïntegreerde voertuig- en personeelsplanning. Hierbij hebben we onderscheid gemaakt tussen problemen met één depot (garage/remise) en met meerdere.

De reden hiervoor is dat de wiskundige complexiteit van het onderliggende voertuigplanningsprobleem verandert indien we van één naar meerdere depots gaan. Het voertuigplanningsprobleem is namelijk in polynomiale tijd oplosbaar indien er sprake is van één depot, terwijl het in geval van twee of meerdere depots behoort tot de klasse van NP-moeilijk problemen. Kortweg kunnen we ook zeggen dat er in het eerste geval sprake is van een "makkelijk" probleem, terwijl er in het tweede geval sprake is van een "moeilijk" probleem. Het personeelsplanningsprobleem en daarmee ook het geïntegreerde probleem zijn echter altijd "moeilijk". Daardoor kan de rekentijd om het probleem optimaal op te lossen op een gegeven moment exploderen als de grootte van de probleeminstantie ook maar iets toeneemt. Dit is onafhankelijk van de snelheid van de computer en geldt dus ook bij het gebruik van de snelste en modernste computers.

Voor het geval met één depot hebben we in hoofdstuk 3 een uitgebreid literatuur overzicht gegeven. Vervolgens hebben we voor verschillende situaties een wiskundig model en een algoritme besproken. Met verschillende situaties doelen we hierbij op de mogelijkheid dat een chauffeur van voertuig mag wisselen tijdens zijn of haar dienst of niet. De gepresenteerde algoritmen maken allemaal gebruik van wiskundige technieken uit de mathematische besliskunde, te weten kolomgeneratie en Lagrange relaxatie. Dit zijn geavanceerde technieken die veel worden gebruikt om dit type problemen op te lossen. Aan de hand van uitgebreide rekenresultaten hebben we vervolgens aangetoond dat er vaak een besparing van één of meerdere diensten mogelijk is ten opzichte van de sequentiële aanpak. Hierbij dient echter opgemerkt te worden dat deze besparing veel groter is, als overstappen van chauffeurs tijdens een dienst niet is toegestaan. Tenslotte, hebben we in dit hoofdstuk ook nog een toepassing bij het Rotterdamse openbaarvervoerbedrijf RET bekeken. Diverse probleeminstanties van de RET hebben we met onze geïntegreerde methode opgelost. Daarvoor hebben we echter wel enkele aanpassingen moeten doen, aangezien sommige bedrijfsregels zeer gecompliceerd kunnen zijn. Deze aanpassingen zijn uitgebreid besproken in dit hoofdstuk.

In het geval van meerdere depots (hoofdstuk 4) hebben we modellen en algoritmen ontwikkeld om ook het probleem met meerdere depots op te lossen. Deze modellen en algoritmen zijn tot op zekere hoogte generalisaties van de modellen en algoritmen uit hoofdstuk 3. Hierbij dient echter wel rekening gehouden te worden met het feit dat het onderliggende voertuigplanningsprobleem nu "moeilijk" is geworden. Aan de hand van data van Connexxion, 's lands grootste streekvervoerder, hebben we aangetoond dat we met deze aanpak middelgrote probleeminstanties kunnen oplossen en dat daarmee een significante besparing verkregen kan worden ten opzichte van de sequentiële methode. Deze besparing is overigens veel groter dan in het geval van één depot. Deze besparing liep op tot zo'n 13%, terwijl er bij instanties met slechts één depot hoogstens sprake was van een besparing van 7% voor een vergelijkbare situatie met overstappen toegestaan. Tevens hebben we in dit hoofdstuk een nieuwe methode besproken om aselect probleeminstanties

te genereren. Deze methode verschilt met de eerder in de literatuur gebruikte methoden in het feit dat de structuur van "echte" instanties uit het streekvervoer terugkomen in de zelf genereerde data. Er bestaan echter ook instanties die zodanig groot zijn dat ze niet met de door ons ontwikkelde (of door andere in de literatuur beschreven) algoritmen kunnen worden opgelost. Daarom hebben we aan het eind van het hoofdstuk ook nog gekeken naar enkele methoden om grote instanties in meerdere kleine te splitsen. Door dit op een "slimme" manier te doen, blijkt de kwaliteit van de geïntegreerde oplossing nauwelijks achteruit te gaan. Daardoor blijft ook de besparing ten opzichte van de sequentiële aanpak nog steeds groot. Overigens kunnen we over zulke splitsingsmethoden geen absolute kwaliteitsgaranties geven. Ze kunnen alleen met elkaar worden vergeleken. In de praktijk neemt men hier overigens meestal wel genoegen mee.

In hoofdstuk 5 hebben we een totaal nieuwe manier van openbaar vervoerplanning bekeken, namelijk een dynamische methode voor voertuig- en personeelsplanning. In tegenstelling tot het traditionele planningsproces ligt de planning dan niet meer voor een half jaar of jaar vast, maar wordt die gedurende de dag aangepast om vertragingen zoveel mogelijk te voorkomen. Dit idee hebben we in eerste instantie toegepast op voertuigplanning. Aan de hand van data van wederom Connexxion hebben we aangetoond dat het aantal ritten dat te laat vertrekt dan behoorlijk afneemt in vergelijking tot de statische methode met vaste buffertijden. Bij deze in de praktijk vaak toegepaste methode moet er tussen ieder tweetal ritten een bepaalde, vaste tijd zitten. Als deze tijd te klein is, kan één enkele vertraging doorwerken op de volgende ritten die door dezelfde bus worden uitgevoerd. Daarentegen, als deze tijd te groot is, zijn er vele extra voertuigen nodig. Door dus volgens een dynamische aanpak te plannen kan ook het aantal gebruikte voertuigen dalen. Vervolgens hebben we deze methode uitgebreid met personeelsplanning. Dat wil zeggen dat we het voertuig- en personeelsplanning probleem geïntegreerd en dynamisch oplossen. Door de grote rekentijd van de algoritmen voor de geïntegreerde aanpak kunnen echter alleen kleine probleeminstanties op deze manier worden opgelost. Grote instanties hebben we opgelost door ze op te splitsen in meerdere kleine instanties. Dit heeft echter als nadeel dat de verkregen winst van het dynamisch plannen weer teniet wordt gedaan. Daarom hebben we helaas moeten concluderen dat deze methode in potentie aantrekkelijk is, aangezien het voor kleine instanties zeer goede resultaten geeft, maar nu nog niet toepasbaar is voor grote instanties uit de praktijk.

Het proefschrift is in hoofdstuk 6 afgesloten met enkele conclusies en suggesties voor toekomstig onderzoek. De belangrijkste conclusies zijn dat we hebben aangetoond onder welke omstandigheden een geïntegreerde aanpak loont en dat zo'n aanpak ook kan worden gebruikt om praktische problemen op te lossen. Tenslotte, hebben we laten zien dat een dynamische aanpak het aantal vertragingen (tegen gelijke kosten) kan reduceren.

Curriculum Vitae

Dennis Huisman was born in Dordrecht, the Netherlands, on July 28, 1978. In 1995, he started his study in Econometrics at *Erasmus University Rotterdam* after completing his secondary school education at *De Ring van Putten* in Spijkenisse. Moreover, he was affiliated with the *RET*, the public transport company in Rotterdam, from 1996 until 1999. In 1999, he finished his Econometrics study by writing a master thesis on applying an integrated approach to vehicle and crew scheduling at the RET.

In the same year, Dennis started his PhD study, of which this thesis is the result. Several parts of this thesis have been published (or accepted to be published) as articles in international scientific journals such as *Transportation Science* and *Journal of Scheduling*. Furthermore, as a spin-off of this research, a crew scheduling system, DOPT (Duty Optimization for Public Transport), has been developed for *Stadsbus Maastricht*.

Dennis founded, together with Richard Freling and Albert Wagelmans, ECOPT (Erasmus Center for Optimization in Public Transport) in 2001. ECOPT organized the well-visited *Conference on Optimization in Public Transport* in May 2002.

Since September 2001, Dennis also worked as a part-time assistant professor at the Econometric Institute. During this period, he taught several courses (Quantitative Methods for Location, Distribution and Transportation; Case Studies in Logistics) and he joined the Rintel-project. This is a commercial project for *NS Reizigers*, the main Dutch Railway operator, and deals with shunting passenger train units. As a result of this project, Ramon Lentink and Dennis won the first prize in the 2002 Management Science in Railroad Applications Student Competition, organized by the Rail Applications Special Interest Group of INFORMS and Railway Age. Their winning paper *Shunting Passenger Train Units in a Railway Station* will appear in *Transportation Science*.

The Tinbergen Institute is the Institute for Economic Research, which was founded in 1987 by the Faculties of Economics and Econometrics of the Erasmus Universiteit Rotterdam, Universiteit van Amsterdam and Vrije Universiteit Amsterdam. The Institute is named after the late Professor Jan Tinbergen, Dutch Nobel Prize laureate in economics in 1969. The Tinbergen Institute is located in Amsterdam and Rotterdam. The following books recently appeared in the Tinbergen Institute Research Series:

270. N.K. BOOTS, *Rare event simulation in models with heavy-tailed random variables.*
271. P.J.M. MEERSMANS, *Optimization of container handling systems.*
272. J.G. VAN ROOIJEN, *Flexibility in financial accounting; income strategies and earnings management in the Netherlands.*
273. D. ARNOLDUS, *Family, family firm, and strategy. Six Dutch family firms in the food industry 1880-1970.*
274. J.-P.P.E.F. BOSELIE, *Human resource management, work systems and performance: A theoretical-empirical approach.*
275. V.A. KARAMYCHEV, *Essays on adverse selection: A dynamic perspective.*
276. A.J. MENKVELD, *Fragmented markets: Trading and price discovery.*
277. D. ZEROM GODEFAY, *Nonparametric prediction: Some selected topics.*
278. T. DE GRAAFF, *Migration, ethnic minorities and network externalities.*
279. A. ZORLU, *Absorption of immigrants in European labour markets. The Netherlands, United Kingdom and Norway.*
280. B. JACOBS, *Public finance and human capital.*
281. PH. CUMPERAYOT, *International financial markets: Risk and extremes.*
282. E.M. BAZSA-OLDENKAMP, *Decision support for inventory systems with complete backlogging.*
283. M.A.J. THEEBE, *Housing market risks.*
284. V. SADIRAJ, *Essays on political and experimental economics.*
285. J. LOEF, *Incongruity between ads and consumer expectations of advertising.*
286. J.J.J. JONKER, *Target selection and optimal mail strategy in direct marketing.*

287. S. CASERTA, *Extreme values in auctions and risk analysis.*

288. W.H. DAAL, *A term structure model of interest rates and forward premia: An alternative approach.*

289. H.K. CHAO, *Representation and structure: The methodology of econometric models of consumption.*

290. J. DALHUISEN, *The economics of sustainable water use. Comparisons and lessons from urban areas.*

291. P. DE BRUIN, *Essays on modeling nonlinear time series.*

292. J. ARDTS, *All is well that begins well: A longitudinal study of organisational socialisation.*

293. J.E.M. VAN NIEROP, *Advanced choice models.*

294. D.J. VAN VUUREN, *The market for passenger transport by train. An empirical analysis.*

295. A. FERRER CARBONELL, *Quantitative analysis of well-being with economic applications.*

296. L.M. VINHAS DE SOUZA, *Beyond transition: Essays on the monetary integration of the accession countries in Eastern Europe.*

297. J. LEVIN, *Essays in the economics of education.*

298. E. WIERSMA, *Non-financial performance measures: An empirical analysis of a change in a firm's performance measurement system.*

299. M. MEKONNEN AKALU, *Projects for shareholder value: A capital budgeting perspective.*

300. S. ROSSETTO, *Optimal timing of strategic financial decisions.*

301. P.W. VAN FOREEST, *Essays in financial economics.*

302. A. SIEGMANN, *Optimal financial decision making under loss averse preferences.*

303. A. VAN DER HORST, *Government interference in a dynamic economy.*

304. A.P. RUSSO, *The sustainable development of heritage cities and their regions: Analysis, policy, governance.*

305. G.A.W. GRIFFIOEN, *Technical analysis in financial markets.*

306. I.S. LAMMERS, *In conflict, een geschiedenis van kennismanagement.*

307. O.L. LISTES, *Stochastic programming approaches for strategic logistics problems.*

308. A.T. DE BLAEIJ, *The value of a statistical life in road safety.*

309. S.H.K. WUYTS, *Partner selection in business markets. A structural embeddedness perspective.*

310. H.C. DEKKER, *Control of inter-organizational relationships: The effects of appropriation concerns, coordination requirements and social embeddedness.*

311. I.V. OSSOKINA, *Environmental policy and environment-saving technologies. Economic aspects of policy making under uncertainty.*

312. D. BROUNEN, *Real estate securitization and corporate strategy: From bricks to bucks.*

313. J.M.P. DE KOK, *Human resource management within small and medium-sized enterprises.*

314. T. VERHAGEN, *Towards understanding online purchase behavior.*

315. R. HOEKSTRA, *Structural change of the physical economy. Decomposition analysis of physical and hybrid-units input-output tables.*

316. R.K. AIDIS, *By law and by custom: Factors affecting small and medium sized enterprises during the transition in Lithuania.*

317. S. MANZAN, *Essays in nonlinear economic dynamics.*

318. K. OLTMER, *Agricultural policy, land use and environmental effects: Studies in quantitative research synthesis.*

319. H.J. HORBEEK, *The elastic workfloor. About the implementation of internal flexibility arrangements.*

320. P. HOUWELING, *Empirical studies on credit markets.*

321. E. MENDYS, *Essays on network economics.*

322. J. NOAILLY, *Coevolutionary modeling for sustainable economic development.*

323. Y.V. VELD-MERKOULOVA, *Essays on futures markets and corporate spin-offs.*

324. F.J. WILLE, *Auditing using Bayesian decision analysis.*