Martin Burtscher, BSc

Combined Interactive Vehicle and Duty Scheduling in the Area of Local Public Transport

MASTER'S THESIS

to achieve the university degree of

Master of Science

Master's degree programme: Software Development and Business Management

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Ing. Dr.techn. Wolfgang Slany

Institute of Software Technology

Graz University of Technology

Graz, April 2016

AFFIDAVIT

Abstract

Over the last few years, bus companies have been under increasing pressure, created by competition due to rising privatization in the public transport sector and reduced subsidies of public authorities. Moreover, human planners are dissatisfied with market-established scheduling solutions. For small bus companies market-established systems are too expensive and additionally cannot cover all the special constraints and rules which are present in rural areas.

This thesis describes a new interactive approach to creating efficient combined vehicle and duty schedules for rural areas that are usable in practice. The introduced system integrates a human scheduler and all their domain specific knowledge in the scheduling process. Special attention is paid to the concept of combining the skills of humans and computers. The human planner is not replaced by a system, rather the planner is supported by a quick and comprehensive presentation of information and statistics including schedule suggestions.

The second part of the thesis deals with implementation details of the interactive planning system. Basic elements of the system structure are discussed and three different scheduling algorithms are introduced. The heuristic scheduling algorithms are able to suggest initial feasible schedules. A human planner then has various opportunities to intervene in the scheduling process. The user interface allows a user to modify the generated solution and fix parts of a suggested or existing schedule which the user is satisfied with. Based on these modifications and selections, the system can provide further suggestions. Planners are additionally supported in their decision-making process by clear and meaningful statistics during the overall scheduling process. To enable human schedulers to include known regional requirements, various tangible configuration parameters are described. Besides built-in rules which handle legal regulations and scheduling requirements, the system can be extended with customized rules for the special demands of planners.

Evaluations and benchmarks have shown, that the system is usable in practice and is able to generate acceptable schedules within a short time. The resulting schedules are comparable with human planned solutions with regards to costs, which have been gradually optimized over years. The system and the concept of interactive scheduling has been well received by planners of various bus companies.

Zusammenfassung

In den letzten Jahren sind Busunternehmen, aufgrund von Wettbewerb durch steigende Privatisierung im öffentlichen Verkehr, als auch durch gekürzte Subventionen der öffentlichen Hand, immer stärker unter Druck geraten. Außerdem sind Planer mit den verfügbaren, etablierten Planungslösungen unzufrieden. Sie sind zu teuer und können darüber hinaus nicht alle besonderen Anforderungen erfüllen, die ländliche Regionen mit sich bringen.

Diese Arbeit beschreibt eine neue interaktive Herangehensweise um praxistaugliche und effiziente kombinierte Umlauf- und Dienstpläne für ländliche Regionen zu erstellen. Das vorgestellte System integriert einen menschlichen Planer und sein gesamtes Domänenwissen in den Planungsprozess. Besondere Aufmerksamkeit wird dabei auf die Kombination der Fähigkeiten von Menschen und Computern gelegt. Der menschliche Planer wird nicht von einem System ersetzt, vielmehr wird der Planer von einer schnellen aber umfassenden Darstellung von Informationen und Statistiken einschließlich Planungsvorschlägen unterstützt.

Der zweite Teil der Arbeit beschäftigt sich mit Implementierungsdetails des interaktiven Planungssystems. Grundlegende Elemente der Systemstruktur werden erörtert, und drei unterschiedliche Planungsalgorithmen werden vorgestellt. Die heuristischen Planungsalgorithmen sind in der Lage initiale, aber direkt nutzbare Lösungen vorzuschlagen. Der Planer hat dann unterschiedliche Möglichkeiten um in den Planungsprozess einzugreifen. Die Benutzeroberfläche erlaubt es, die generierte Lösung zu verändern und geeignete Teile der vorgeschlagenen oder existierenden Umläufe zu fixieren. Das System kann auf Basis der durchgeführten Anpassungen und der Auswahl guter Umlaufteile weitere Vorschläge liefern. Planer werden zusätzlich mit aussagekräftigen Statistiken in ihrem Entscheidungsprozess während der gesamten Planung unterstützt. Um regionale Anforderungen berücksichtigen zu können, werden verschiedene, greifbare Konfigurationsparameter beschrieben. Neben den integrierten Regeln für die Planungsanforderungen und gesetzlichen Vorschriften kann das System durch individuelle Regeln für spezielle Anforderungen erweitert werden.

Auswertungen und Benchmarks haben gezeigt, dass das System in der Praxis einsetzbar und in der Lage ist akzeptable Umläufe in kurzer Zeit zu generieren. Die Kosten der resultierenden Umläufe sind mit den manuel optimierten Lösungen vergleichbar, die über Jahre sukzessive optimiert wurden. Das System und das Konzept der interaktiven Planung wurde von Planern verschiedener Busunternehmen gut angenommen.

Acknowledgements

Thanks to all who supported me while I wrote this thesis.

I would particularly like to thank Prof. Wolfgang Slany for his help and advice during my master's thesis.

I would also like to thank Claudio Ganahl from the TeleMatrik PTS GmbH for the highly interesting topic and the opportunity to realize it as a part of my thesis.

Furthermore, my thanks go to Teresa Feuerstein, Daniel Burtscher and Michael Burtscher for reviewing my thesis.

Last but not least I sincerely thank my parents Rosmarie and Alfred Burtscher for their great support throughout my entire education.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the last few years, bus companies have come under an increasing amount of pressure. This is created by competition due to rising privatization in the public transport sector and reduced subsidies of public authorities. Moreover, subsidies for public transport have to be put out for public tender instead of being given directly to local companies. Therefore, bus companies have to reduce their costs to remain competitive. Besides lowering wages or discontinuing unprofitable lines, costs can be reduced by increasing the efficiency of current schedules.

This thesis introduces a new interactive approach to support human planners in creating vehicle and duty schedules that are efficient and usable in practicality. The first part of the thesis gives an overview of the planning involved in public transport and various solution approaches described in literature. Chapter 2 describes the planning process in public transport and particularly addresses vehicle and duty scheduling. Chapter 3 introduces a model for vehicle scheduling and the combined vehicle and duty scheduling problem. Additionally, various exact and heuristic solution approaches are introduced and the problem of optimization algorithm testing is discussed. Interactive knowledge-based solution approaches are covered in Chapter 4, where the important role of a human scheduler is addressed and different degrees of interactions are described.

The second part of the thesis covers the technical implementation of this work in detail. Chapter 5 introduces the practical problem addressed in this thesis, the partner company and the characteristics of regional test areas. Chapter 6 contains detail of the implementation, including the basic concept, the architecture and algorithms and its configuration and rules. Chapter 7 presents the results of benchmarks and feedback from human schedulers.

Additionally, existing problems regarding the implementation and concepts are presented. Chapter 8 gives an overview of the current state and the practical usage of the introduced system. Finally, further developments are discussed and an outlook is given.

Chapter 2

Planning in Public Transport

In this chapter the planning process in public transport is introduced. The areas of planning are described, whereas vehicle and duty scheduling receive particular attention and explained in detail. Important definitions of the terminology used in public transport are given and the advantages of a combined scheduling approach are discussed.

2.1 Planning Process

The planning process in public transport can be divided into strategic planning and operational planning (Weider, 2007). Strategic planning is concerned with long term processes, such as infrastructure and timetabling. Operational planning deals with the resources of a company, like vehicles and duties. An overview of the planning process is illustrated in Figure 2.1. In the following, each phase of the process is described in more detail.

1. Strategic Planning

   Network and Line Design is the initial phase of the planning process. In this phase a topology is defined which includes bus stops and their locations. The topology is then connected to a network by creating fixed lines. Additionally, a frequency has to be determined for each line which can handle the expected passenger volumes.

   The goal of the network and line design is to reduce the dwell times and bus changes required to get from an origin to a specific destination to enhance the convenience for passengers. At the

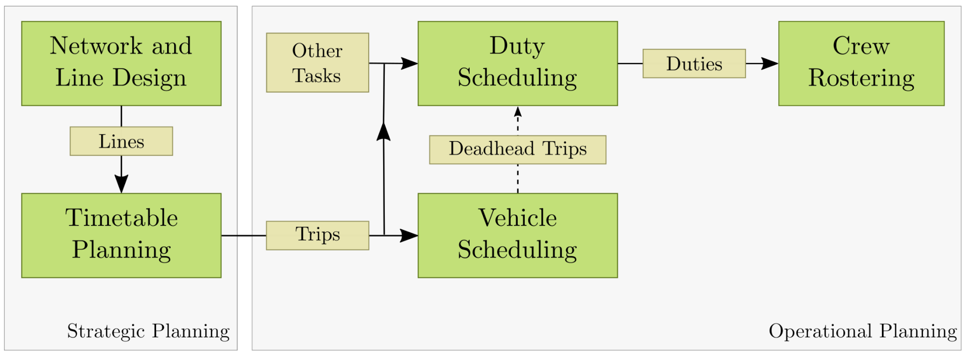


Figure 2.1: The planning process in public transportation.

same time the costs should be minimized. The output of the network and line design phase is a network of bus stops and lines with information about the locations of bus stops, required times and distances of lines as well as their required frequencies.

The process of network and line design is often historically grown and will seldom change completely. The reason for this is that for instance infrastructure, like parking areas or other service connections, are tailored to existing bus stops and lines. For this reason a network and line design is often only modified or extended slightly.

Timetable Planning Based on the defined lines and their frequencies, the arrival and departure times of vehicles at bus stations have to be defined. This process is called timetabling. An instance of a bus line with a defined arrival and departure time is called a trip. The output of the timetabling phase is trips, which cover all lines and frequencies. Additionally, if there is more than one vehicle type available, it is also assigned to the specific trips in this phase, based on demands such as passenger volumes or road conditions. The goal of the timetable planning phase is to minimize the waiting times of passengers at bus stops. This includes the coordination of different lines to make buses meet at major hubs at the same time or within a short time span.

2. Operational Planning

Vehicle Scheduling deals with the problem of assigning trips, stemmed from the timetable, to vehicles at minimum costs. Therefore the number of vehicles and deadhead trips needed to serve all trips should be as low as possible. A detailed description of vehicle scheduling including definitions and properties is given in Section 2.2.

Duty Scheduling is responsible for creating duties from various tasks. In public transport, tasks are mainly trips, stemmed from a timetable and are already planned in a vehicle schedule. Additionally, other tasks, like sign-on or sign-off tasks, which are needed to prepare a vehicle, have to be planned if needed. Duty scheduling is a hard task due to legal regulations, certain domain specific rules or restrictions from local authorities. Duty scheduling is discussed in detail in Section 2.3.

Crew Rostering is concerned with the scheduling of duties, resulting from the duty scheduling process. Duties have to be assigned to individual drivers for a fixed planning period. The rostering process is restricted by various rules, like legal regulations or preferences of employees.

A huge amount of research has been performed on solving each part of the schedule process individually, either by exact or heuristic methods. Additionally, there have been further attempts to integrate different parts of the scheduling process to achieve better optimization results.

In this thesis the vehicle scheduling problem (Section 2.2) and the duty scheduling problem (Section 2.3) are discussed in detail. The practical work, following in the second part of the thesis, is based on a combined approach of vehicle and duty scheduling which is described in Section 2.4.

2.2 Vehicle Scheduling

The vehicle scheduling problem is concerned with assigning trips, stemmed from a timetable, to vehicles at minimum cost (Weider, 2007; Ceder, 2015, Chapter 7, p. 164ff). Therefore the amount of vehicles and deadhead trips needed to serve all trips should be as low as possible.

Trips, stemmed from a timetable, are called *service trips* and have the task of transporting passengers. *Deadhead trips*, do not transport any passengers. There are different types of deadhead trips:

1. *Transfer* trips connect service trips which do not end and start at the same location.
2. *Pull-out* trips connect a depot with the first trip of a tour.
3. *Pull-in* trips connect the last trip of a tour to a depot.

4. Deadhead trips are also used to connect two trips which meet at the same location including waiting times.

A *tour* is a sequence of trips and is assigned to an individual vehicle. The result of the vehicle scheduling is a list of tours to serve all trips stemming from the timetable.

All service trips have a defined start-time and an end-time, a start-location and an end-location. Additionally, the driving time and driving distance must be known to be able to schedule service trips and calculate the trip and tour costs. The start and end time of trips are already defined in the timetable. The start and end locations and the locations of intermediate stations of service trips are defined manually in the network and line design phase. For an exact cost calculation the driving time and the distances of service trips are needed, which are also specified in the network and line design phase.

Due to the fact that there is a large number of deadhead trips possible and whether the driving time or the distance of each deadhead trip is known by human planners, deadhead trips are usually not defined manually in advance. They can be computed either by a routing planner, which provides exact time and length values as used in the practical implementation (Section 6.1.4). Or the properties result from an interpolated approach by calculating the linear distance between two coordinates. The needed time is then calculated by multiplying the interpolated distance by an average speed.

Each vehicle has a *vehicle type*, such as small bus, standard bus, double decker or articulated bus. Each vehicle type has different characteristics, such as a number of seats, an average speed, the length or height of the bus. This can be relevant for the planning process because trips can have different requirements on the vehicle type, which must be fulfilled. For example, there are trips that cannot be driven by a bus which is too long because otherwise it cannot get a round a curve, or too high to pass a low bridge. On the other hand there are trips where a high passenger rate is expected and therefore a bigger bus with more seats is needed.

The final costs result from the fixed costs of the vehicle type, the fuel consumption per km, the driving time and the costs for a driver, see Section 2.3.2. Therefore, reducing deadhead trips and waiting times will lead to lower costs.

2.3 Duty Scheduling

Duty scheduling is concerned with covering a set of tasks efficiently (Weider, 2007). Tasks in public transport stem from a timetable, or vehicle scheduling respectively. Additionally, there are other tasks like administrative work or sign-on and sign-off tasks which are needed to prepare the vehicle before and/or after a vehicle is put into operation. A duty in the area of public transport is a set of tasks served by one driver, taking into account legal regulations. Furthermore, there can be other domain specific rules and restrictions which must be observed. In the following, basic legal regulations and the duty cost calculation is described.

2.3.1 Legal Regulations

Austria and Germany are the countries that are of most interest to the partner company, which will be introduced in Section 5.2. Therefore, only Austrian and German regulations are addressed in this thesis. There are different legal regulations concerning duty scheduling. First, the EU regulation (EC) No 561/2006[1] introduces basic definitions and says:

> “ The Member States should lay down rules for vehicles used for the carriage of passengers on regular services where the route covered does not exceed 50 km. Those rules should provide adequate protection in terms of permitted driving times and mandatory breaks and rest periods.”

This means that all routes that do no exceed the 50 kilometre limit are regulated by national states, which applies to nearly all routes in local public transport.

National legal regulations concerning duty scheduling are the working time law (Arbeitszeitgesetz) and the driving time directive (Lenkzeitverordnung), for German regulations, see e.g., Bagdahn (2015). However, the Austrian and German Rules are very similar, with the difference that the quotient rule, explained in the following, is no longer valid in Austria and the driving period is 4 1/2 hours in Germany and 4 hours in Austria.

In the following, definitions and the most common legal regulations are introduced. The implementation and a detailed description of the legal reg-

[1]`http://eur-lex.europa.eu/resource.html?uri=cellar:5cf5ebde-d494-40eb-86a7-2131294ccbd9.0005.02/DOC_1&format=PDF`

ulation as rules is given in Section 6.3.1.

- The *daily working time* of a driver is at most 13 hours.
- The *daily driving time* of a driver is at most 9 hours.
- The *non-interrupted driving time*, which means the accumulated driving time after or before a break or rest, is at most 4 hours in Austria and 4 1/2 hours in Germany. These non-interrupted driving blocks are called *driving periods.*
- *Breaks*: To start a new driving period, the driving period must be interrupted by one break of at least 30 minutes or by two breaks of at least 20 minutes each or by three breaks of at least 15 minutes each. These breaks must be taken before the driving period reaches 4 hours, or 4 1/2 hours respectively.
- *Quotient-rules*, namely the one-sixth rule, are only valid in Germany. The one-sixth rule additionally allows the required breaks to be divided into blocks of at least 10 minutes each. The duty is valid if the total break time is at least one sixth of the total driving time. The one-sixth rule must also be valid within a driving period.

2.3.2 Duty Costs

Costs of duties mainly result from the working time of a driver. However, in accordance to the time law, a driver must have appropriate breaks which could be paid or unpaid. The legal regulations, concerning paid and unpaid breaks, vary from country to country.

In Austria, a maximum 1 1/2 hour of break time within a duty can be unpaid. Additionally, an unpaid break is only possible if the duration of a duty is more than six hours. Finally, breaks in the first two hours and in the last 2 hours of a duty are always paid. For a detailed explanation and special regulations of the Austrian time law relating to paid and unpaid breaks see, e.g., Schmeidl (2012).

Although, human and automated schedulers try to create or generate balanced duties with adequate break length, there are often 'better' and 'worse' duties, which means if a driver likes a tour or not. This also depends on the country. In some countries it is usual to have a long lunch break, in other countries it is more common to have a short lunch break, but end work earlier (Weider, 2007). Therefore, bus companies often have special agreements concerning paid and unpaid breaks with bus drivers, besides the legal framework, which must be fulfilled anyway. For example, breaks in 'worse' duties are paid more often than required by law, to compensate the resulting inconveniences. Different penalization strategies can be introduced to prevent algorithms from generating unbalanced or 'worse' duties. One example is the penalization of too short or too long breaks.

Furthermore, duty costs can change abruptly even if only one additional trip is added or removed from a duty. For example, if on the one hand the duty duration drops below six hours, it will lead to completely paid breaks, based on the Austrian time law described above. This may result in significantly higher costs. On the other hand, if additional trips are added and the tour duration then exceeds six hours, the costs may even decrease because unpaid breaks are possible.

It is very challenging to calculate correct duty costs, while respecting all legal regulations and it is practically impossible to calculate the exact cost including special agreements between bus companies and drivers. Therefore, the costs calculated by a scheduling system are always only a reference value.

2.4 Combined Vehicle and Duty Scheduling

For each phase in the planning process, described in Section 2.1, numerous solution approaches are known, which solve each problem sequentially and individually. In the following, combined approaches are presented, which try to optimize different areas of the planning process in one step. This enables more possibilities and is in some cases even needed to get an optimal solution.

A lot of research has been done in the area of solving each step of the public planning process individually. However, it often leads to significantly better results if different steps of the planning process are combined and solved in one step. Van den Heuvel, Akker, and Niekerk (2008) presented an integrated approach to timetabling and vehicle scheduling. The results showed that a significant reduction in the operational cost can be achieved by slightly changing the departure and arrival times in the timetable. Of course, it is not always possible to change the timetable, this depends on local authorities. Sometimes the timetables are given and offered by that region. A consumer then has to take the timetables and generate efficient vehicle schedules from it. To take advantage of combined timetabling and vehicle scheduling, changing the timetable must be allowed to a certain extent.

By combining vehicle and duty scheduling, on which this thesis is focused on, better results can be achieved and problems can even be solved where a sequential approach would fail. In regional public transport, trips are longer and fewer relief points exist. It is often the case that a driver starts at a depot and finishes his duty in the same depot again. Weider (2007) presented an example of a rural area, where a sequential approach cannot produce any feasible solutions without violating the working time law or the driving time directive. It is illustrated as a network flow diagram in Figure 2.2. In these kind of diagrams, deadheads are also used to connect trips which meet at the same location, including waiting times between the trips if present.

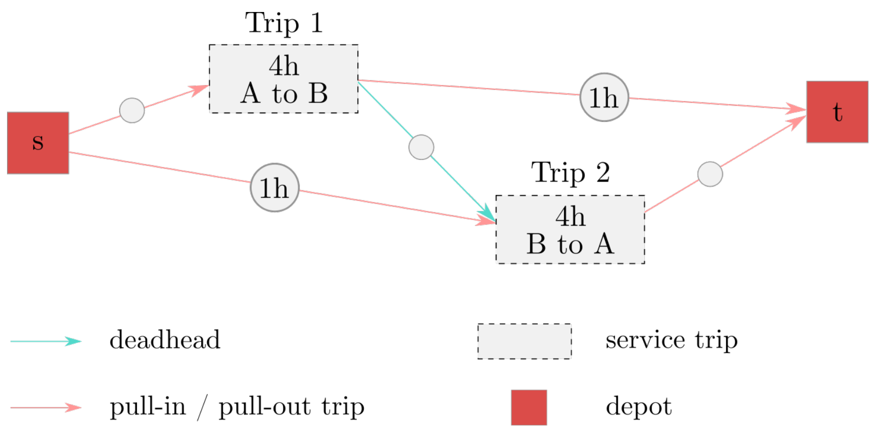


Figure 2.2: In this example, a sequential approach will not lead to a feasible solution, only a combined approach of vehicle and duty scheduling can produce valid results. Image is based on Weider (2007).

In the example there are two service trips to schedule. Trip 1 takes four hours to get from A to B. It is directly followed by Trip 2 which also takes four hours to get back from B to A. Trip A starts at depot s. From depot s to trip B, one hour is needed, which means that the trip A and B do not take the direct route. A vehicle scheduling algorithm will schedule Trip 1 and Trip 2 in a row, for which exactly one vehicle is needed, which is the optimal solution. For a duty scheduler, it will be impossible to generate a feasible duty from the vehicle schedule, where Trip 1 and 2 are bound to one vehicle. The driving time direction of 4 1/2 (based on German rules) will be violated. The problem can be solved by using a second vehicle, which serves Trip 2. The solution is shown in Figure 2.3.

To achieve the driving time direction, a driver on bus 1, which serves Trip 1 has to take a break of 30 minutes, before a depot ride can be taken. Furthermore, before Trip 2 can be served, a break of 30 minutes is needed as well.

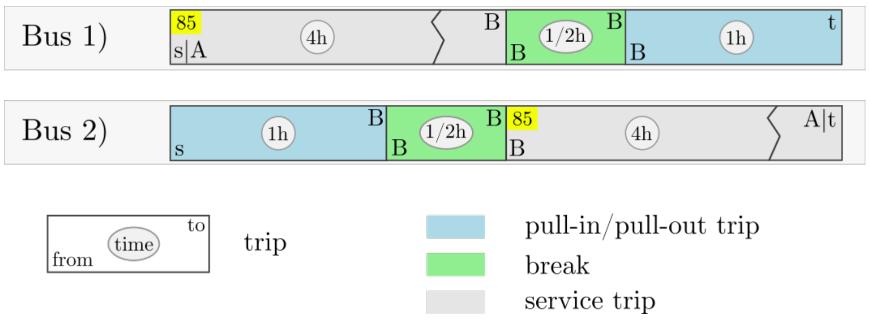


Figure 2.3: The resulting duty schedule if a combined vehicle and duty schedule approach is used.

Chapter 3

Solution Approaches

Over the last few decades extensive research has taken place in all areas of the public transport planning process, which is discussed in Section 2.1. Vehicle scheduling and the combination of vehicle and duty scheduling are particularly focused on. The main problem of vehicle scheduling is the number of feasible solutions due to the large number of possible deadheads, especially when taking multiple depots into account (Ceder, 2011). Due to its size, it is not possible to solve the problem using an exhaustive approach. Therefore, most of the literature is concerned with the computational issues of the vehicle scheduling problem. The published solution approaches in this area can basically be divided into *exact approaches* and *heuristics* (X. Zuo et al., 2015). In the following, the scheduling problem and basic algorithms which are used in exact and heuristic solution approaches are discussed.

3.1 The Problem

As already mentioned, the main problem with vehicle and duty scheduling is the huge number of feasible solutions. Different models of the vehicle scheduling problem are presented in the literature. However, Valouxis and Housos (2002) modelled a combined vehicle and duty scheduling problem as a zero-one integer programming problem (IP). In theory all existing valid day shifts could be generated using a search tree. While creating this search tree, all legal regulations and special requirements can be indirectly checked during the creation process. By excluding these regulations and requirements from the model, the following zero-one program can be defined:

$$\text{Minimize} \quad \sum_{s=1}^{S} c_s x_s \tag{3.1}$$

$$\text{Subject to} \quad \sum_{s=1}^{S} \beta_{bs} x_s \leq 1, \qquad b \in B, \tag{3.2}$$

$$\text{and} \quad \sum_{s=1}^{S} \tau_{ts} x_s = 1, \qquad t \in T, \tag{3.3}$$

$$x_s = \{1, 0\}, \qquad j = 1, \ldots, n,$$

where c_s are the costs of a shift s and S is the number of feasible generated shifts. B is the set of available buses and T the set of all trips. The coefficient $\beta_{bs} = 1$ if a bus b is covered by a shift s and $\beta_{bs} = 0$ otherwise. The coefficient $\tau_{ts} = 1$ if a trip t is covered by a shift s and $\tau_{ts} = 0$ otherwise. Finally, x_s indicates whether a generated shift is taken or not.

The objective function in (3.1) forces the selection of shifts with minimal total costs. The constraint given in (3.2) ensures that each bus is assigned to a maximum of one shift. Finally, the constraint in (3.3) guarantees that each trip is covered by exactly one shift. To create reasonable solutions with this model, it is necessary to generate a significant number of shifts assigned to the available buses. This leads to a huge IP problem, which is very difficult or even impossible to solve in reasonable time.

In the following, basic exact and heuristic solution approaches are discussed which are used in various solution approaches for the vehicle scheduling problem in the literature.

3.2 Branch and Bound

In the literature, exact solution approaches of the vehicle scheduling problem and the combination of vehicle and duty schedule are often modelled as an Integer Linear Programming Problem (ILP) or a zero-one integer program. The main problem of an ILP are constraints that limit some objectives to integer values. Various solutions have been proposed which solve the vehicle scheduling problem modelled as an ILP. One basic component used by various solution approaches to solve an ILP is the Branch and Bound algorithm, proposed by Land and Doig (1960) or variations of it. Haase, Desaulniers, and Desrosiers (2001), for instance, presented a solution approach for the simultaneous vehicle and duty scheduling problem using Branch and Bound. Also Carpaneto et al. (1989) uses Branch and Bound to solve the multiple depot vehicle scheduling problem. In the following the Branch and Bound algorithm is explained (Gurobi, 2016) with an example illustrated in Figure 3.1.

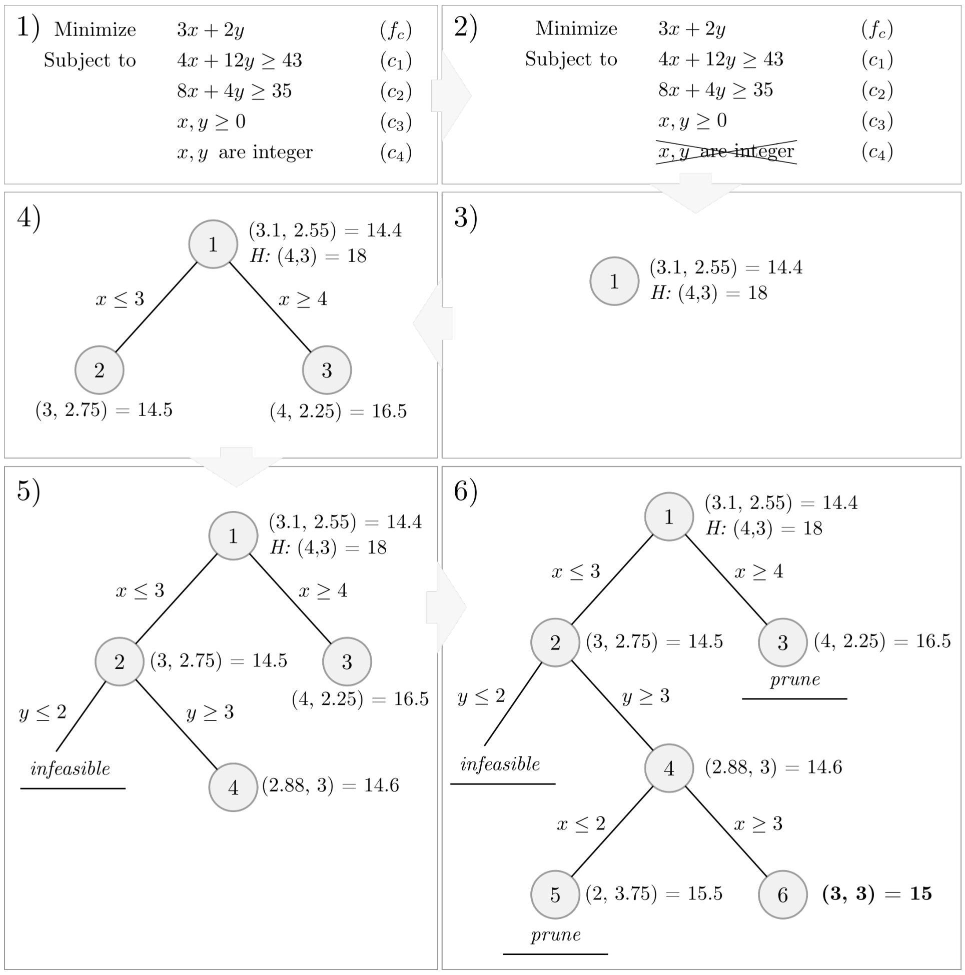


Figure 3.1: In the image a step-by-step example of solving an integer linear programming problem is shown.

Step 1) The Branch and Bound algorithm starts with the initial ILP problem.

Step 2) Next, the problem is relaxed by removing all integer constraints though allowing fractional solutions. The resulting problem is called linear-programming (LP) relaxation of the original ILP and can be solved by Mixed-Integer Programming (MIP) solvers based on the Simplex algorithm.

Step 3) In the example, the result 3.1 for x and 2.55 for y is found, which leads to the cost r_1 of 14.4. This solution is the best value that can theoretically be achieved and is called *lower bound*, in a minimization problem. The closer a feasible solution is to the lower bound, the better it is. If the very unlikely case happens that the result satisfies all integer constraints, although they were not imposed, the optimal solution of the initial ILP has been found and the algorithm can stop.

This is usually not the case and therefore the next step of the algorithm is to calculate an initial solution called *upper bound*, in a minimization problem. The initial upper bound is a first feasible solution calculated by a heuristic. In the example, a very simple heuristic H is used, which rounds up the fractional values of x and y to $x = 4$ and $y = 3$. Inserted in the cost function, it leads to the upper bound of 18. All solutions that are worse than the upper bound can be discarded.

To measure the quality of the solution, an optimality, gap G_{opt}, between the lower and upper bound is calculated. The optimality gap is calculated as follows:

$$G_{opt} = \frac{U - L}{L} \times 100$$

where U is the upper bound and L is the lower bound. This leads to an initial optimality gap of 25%.

Step 4) The next step is called branching. Therefore, a variable is selected, which is an integer restricted variable in the initial ILP, but has a fractional value in the result of the MIP solver.

In the example, variable x is selected first. To create branches, additional constraints are added, which divides the original problem, in this case, into two sub problems. For one branch, the constraint $c_{11} : x \leq 3$ is added, and another branch is created by adding the constraint $c_{12} : x \geq 4$. These constraints exclude fractional solutions for x between 3 and 4. For both branches, the new problem is solved by MIP solver, with the additional constraints c_{11} and c_{12} respectively, but still without the integer restrictions. This leads to the solution of $x = 3$ and $y = 2.75$ with the costs $r_2 = 14.5$ in node 2 and $x = 4$ and $y = 2.25$ with the costs $r_3 = 16.5$ in node 3.

If node 2 and 3 are evaluated, a new upper bound can be set, by taking $min(r_2, r_3)$, which is 14.5, because no better solution can be achieved in this state. Based on the new upper bound, the optimality gap can be recalculated and is now 24.1%. The best solution at this point is still 18, because there are no sub solutions, which meets all conditions of the initial ILP.

Step 5) As y is still fractional in both leaf nodes, branching has to be continued and a node has to be chosen to go on with. In this example a best first strategy is used and therefore node 2 is selected, because its cost r_2 is better than the costs r_3 in node 3.

Therefore, like with x in step 4), additional constraints for y are added. These are $c_{21} : y \leq 2$ for the left branch and $c_{22} : y \geq 3$ for the right branch, to exclude fractional solutions between 2 and 3 for y. The left branch of node 2 is, due to its additional constraint, infeasible and can be discarded. The right branch leads to a solution of $x = 2.88$ and $x = 3$ with a result of 14.6 in node 4, which is additional the new lower bound and leads to the new optimality gap of 23.3%.

Step 6) As variable x has a fractional result again, branching has to be continued. Node 4 is selected due to its best lower bound. Therefore, the constraints $c_{31} : x \leq 2$ for the left branch and $c_{32} : x \geq 3$ for the right branch are added. In the left branch of node 4, $x = 2$ and $y = 3.75$ with a result of 15.5 in node 5. In the right branch of node 4, $x = 3$ and $y = 3$ with a result of 15 in node 6. Additionally, all initial constraints of the ILP are fulfilled, including the integer constraints for x and y. This leads to a new upper bound of 15 and an optimality gap of 2.7%. Moreover, the lower bound in node 3 and the lower bound in node 5 are greater than the new upper bound of 15 in node 6. Therefore, both branches can be pruned, because no better solution will be possible there. Due to the fact that all branches are either infeasible or pruned, the result in node 6 is the optimal solution for the initial ILP problem.

The Branch and Bound algorithm can find exact solutions for a ILP. However, if the size of the problem is very large, the Branch and Bound algorithm can take a very long time. Therefore, it is possible to stop the evaluation of new nodes by accepting a solution where the optimality gap reaches a defined threshold. In that case, the initial exact approach of the Branch and Bound algorithm turns into an heuristic approach.

3.3 Basic Heuristic Solution Approaches

Although, exact solution approaches are able to find optimal solutions, the computational time is often unacceptable (X. Zuo et al., 2015). Heuristic approaches can lead to acceptable results in a shorter time. In the following, a selection of heuristic algorithms, which are able to solve combinatorial problems and therefore are used in the area of vehicle scheduling and combined vehicle and duty scheduling, are described. These are *Simulated Annealing*, *Tabu Search* and *Genetic Algorithms*.

3.3.1 Simulated Annealing

Simulated Annealing(SA) is originally inspired by the process of annealing metal work, by heating and cooling the material to change the inner structure (Bertsimas and Tsitsiklis, 1993). When the temperature of the metal goes down, the new structure becomes fixed. In SA, a temperature variable represents the heating process. In the beginning, the temperature variable is high, which allows the algorithm to also accept solutions which are worse than the current solution. This gives the algorithm the ability to leave local optima and scan a wider range of the search space. As the temperature goes down, the acceptance of worse solutions will be reduced to hopefully find solutions which are close to the global optimum.

The acceptance of a new neighbour solution depends on two characteristics. First, if the neighbour solution is better than the current solution, it will be always accepted and replaces the current solution. If the new solution is worse than the current solution, the acceptance function decides if the solution is accepted or not. If the goal of the optimization is minimization, the acceptance function is defined as:

$$f_{accept} = e^{\dfrac{c_{cur} - c_{new}}{T}}$$

where f_{accept} is the acceptance function, c_{cur} are the current costs, c_{new} are the new costs and T is the current temperature of the system. If the temperature is high, as it is in the beginning, a solution is more likely to be accepted, also if the new cost value is significantly worse than the current cost. This ensures that the algorithm can better explore the entire search space, before the temperature cools and therefore the algorithm will focus on a smaller region. Summarized, the algorithm works as follows:

1. Create an initial solution and set an initial temperature.
2. Select a neighbour by making small changes to the current solution.
3. Decide whether the new solution is accepted or not (acceptance function).
4. Decrease the temperature.
5. Stop if the system has sufficiently cooled or an acceptable solution has been found. If this is not the case, go to step 2.

The big advantage over simple hill climbing algorithms is that SA does not have the tendency to get stuck in local optima by also accepting worse solutions, especially in the beginning. As shown in Figure 3.2, the hill climbing algorithm highly depends on the initial starting point and therefore often only finds a local optima, because it stops when no better neighbour can be found. However, the SA algorithm will explore the entire search space and will more likely find an optimal solution.

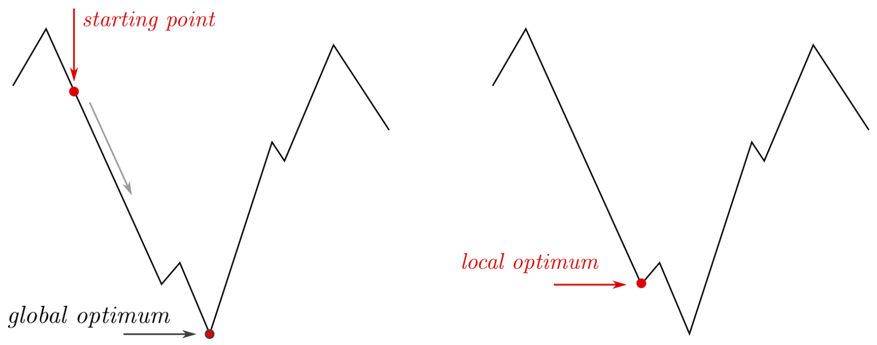


Figure 3.2: The example[1]shows that the performance of a hill climber highly depends on the initial starting point. It stops when no better neighbours can be found.

SA was successfully used by Laurent and Hao (2007). They presented an approach for simultaneous vehicle and driver scheduling in two steps. First, an initial solution was created. Afterwards, the initial solution was progressively improved by SA.

3.3.2 Tabu Search

As Simulated Annealing, Tabu Search (TS) is a local search method, which tries to avoid getting stuck in a local optimum (Glover, 1990). With TS hard combinatorial optimization problems were solved successfully. However, TS can be used to guide any solution process which employs moves that can transform a solution into another and that provides a measuring for the attractiveness of these moves. Examples of these moves are changing a variable, adding or deleting elements from a solution, interchanging the positions of elements, etc.

TS is mainly based on two characteristics, namely *worsening moves* and *prohibitions*. TS allows worse solutions, if no better solution can be found in the next move. Additionally, TS saves previously visited states to avoid the search coming back to already checked solutions (therefore the name

[1]The example image is based on `http://www.theprojectspot.com/images/post-assets/hc_1.jpg` [2016/04]

tabu). Glover (1990) describes three memory structures for saving visited states. First, he describes short-term memory structures, which saves only the last moves which prevents future moves from undoing changes. Next, intermediate-term memory structures are described which has the intention to bias moves toward promising areas of the search space. And finally, longer-term memory structures can be used to promote a general diversity in the search process. The TS algorithm, using a short term memory structure works as follows:

1. Create an initial solution.
2. Create a candidate list of moves from the current solution, where each move leads to a new solution.
3. Choose the best admissible solution candidate. Therefore, all solution candidates are evaluated and compared. Then the best candidate is taken which is not in a list of forbidden solutions (tabu list). The chosen solution is the next current solution. If it also improves the previous best solution, record it as the new best solution.
4. If the stopping criteria is reached (number of iterations or elapsed time since the last best solution is found), the best recorded solution is returned. If the stopping criteria is not reached, go to step 2.

TS was successfully used for an interactive scheduling approach by Kopfer and Schönberger (2002) and is part of various other solution approaches for vehicle scheduling problems.

3.3.3 Genetic Algorithms

Chen and Xingquan Zuo (2014) use a Genetic Algorithm (GA) to solve the vehicle scheduling problem of an urban bus line. GAs are often used for optimization problems, because of their ability to explore a huge search space and find a combination of parameters which will lead to the best solutions. GAs are inspired by the evolution of natural living beings by natural selection and work as follows (Mitchell, 2011):

1. Generate an initial set of solutions, a population.
2. Calculate the fitness for each solution in the population using a cost function which needs to be optimized.

3. Select those solutions with the best fitness from the population, which will be the parents of the next generation. In the literature different selection operators are described. But the basic idea is to select more likely fitter solutions.

4. Crossover the selected parents. Each pair produces new offspring for the new population, by combining parts of the parents with a slight random mutation. Do this as long as the new population has the same number of solutions as the initial population. Afterwards, the new population becomes the current population.

5. Stop, if an acceptable solution is found or a limit is reached. If this is not the case, go to step 2.

GAs are very general algorithms, which can be used in any search space and often are described as the second best solution for every problem. Nevertheless, Chen and Xingquan Zuo (2014) could improve the experience-based solution by 5 percent. Furthermore, the authors stated that the problem of existing approaches is that they may only produce one solution, whereas GAs can produce multiple pareto solutions, which is highly desirable for decision makers, as human schedulers, in practice.

3.4 Testing of Automated Optimizations

Testing is an important part of the software development and maintenance process. An optimization algorithm has to fulfil different requirements like generating good and feasible solutions. Testing optimization results on their feasibility, can be achieved by unit tests which check if all given constraints are fulfilled. The attribute of 'good' results, for instance if the results are also practically relevant, is much harder to check automatically.

One attempt for testing the performance of optimization algorithms is to introduce benchmark test suites. With a significant number of use cases, the performance of different configurations of optimization algorithms, or different variants of algorithms, can be tested and compared. This enables a human expert to tune various weights of the algorithm and check if the modification does not lead to worse results in other use cases. The disadvantage of this approach is the missing measure of the quality, regarding the practical relevance of a problem. Where a new configuration of the algorithm can lead to better results with regards to the cost function of the problem, it might lead to worse robustness as well, which cannot be measured for all test cases.

To solve this problem, Slany (1996) introduced a consistency check, to ensure that configuration changes are consistent with decisions made earlier by human experts. The idea is to create a test suite which contains a set of reference pairs. Each pair consists of two samples which are initially ranked by human experts. Additionally, each pair is created with the explicit aim that a specific configuration change, entails a new order of the samples. Backed with this system, a human expert can change the configuration if he is dissatisfied with a solution generated by the system. Afterwards, the consistency check for all reference pairs is tested with the new configuration. This is done by applying the new configuration to all reference pairs and calculating an evaluation score to be able to compare the results. If the order of each reference pair remains unchanged, the new configuration does not violate earlier decisions made by human experts. If the order changes, the new configuration is either wrong and has to be revoked or the reference pair ranking is obsolete and can be removed. However, an inconsistency is detected by the system and has to be resolved by a human expert to ensure that all decisions, made by the system, remain rational, predictable and understandable.

Slany (1996) suggests that several human experts should agree on a common and undisputed subset of reference pairs, indicating their general tendency for decision making. Or they can agree on a set of reference ranking pair pools distinguished by specific characteristics which corresponds to different configurations. These pools are named by their characteristics, such as 'highest-quality', 'robust' or 'standard-mix'.

Chapter 4

Interactive Knowledge-Based Systems

In this section an interactive approach to vehicle and duty scheduling is presented. An introduction and motivation is given and the role of human schedulers is discussed. Finally, the requirements of interactive systems and a general implementation structure is described.

4.1 Introduction and Motivation

Interactive scheduling describes the idea of integrating the human scheduler into the planning process, equipped by the process power of computers. The idea of interactive scheduling is quite old and has had little attention in recent research.

In fully automated systems, legal regulations and also domain specific knowledge are encoded in various rules. Encoding the complete domain knowledge is hard to achieve completely or is at least a very hard task, as it is with the cost calculation of duties, discussed in Section 2.3.2. For this reason, interactive systems are introduced to integrate a human scheduler and its domain specific knowledge in the scheduling process to be able to generate better and practically useful solutions.

4.2 Degrees of Interaction

In the literature different degrees of interactions are discussed. Sims (1991) uses an interaction approach which only indirectly affects the solution generation. The solution approach is based on biological inspired algorithms, like the genetic algorithm discussed in the previous section. In the paper, the user replaces or supports the automated selection operators, respectively. Other systems, like those presented by Colgan, Spence, and Rankin (1995), allow more interactivity, by controlling optimization and search parameters or change constraints while the optimization process evolves. An even more interactive system is presented by Waters (1984) to solve the vehicle routing problem. It allows a user to modify computer generated solutions. Based on a current, maybe modified solution and new optimization parameters, the system generates new optimized solutions.

4.3 Role of a Human Scheduler

In interactive systems a human scheduler is a fixed part of the optimization process. In contrast to interactive approaches, fully computerized approaches try to replace human schedulers completely (Waters, 1984). This follows on from the fact that computers can do a huge amount of calculations that cannot be undertaken manually by a human scheduler. But these approaches ignore the contribution which experienced schedulers can make. Furthermore, Waters (1984) stated that generalized approaches do not have the flexibility needed for real problems and attributes human schedulers' important characteristics:

> "Although human schedulers cannot do the calculations, they are better at recognizing patterns, finding an acceptable balance between conflicting objectives, using past experiences of similar problems, applying imagination to find unusual solutions, overriding rigid constraints and generally applying subjectivity to a problem."

This statement in strengthened by various case studies, for instance by G. G. Brown and Graves (1981). They stated that human dispatchers cannot be replaced, rather the dispatchers should be supported by a quick and comprehensive presentation of information and statistic concerning the task or problem to be solved. Similarly, Waters (1984) suggest combining the skills of humans and computers. Computers should and can provide suggestions, generate statistics, store results, compare new results with previous results, and so on. Then, based on the provided suggestions and information, a human can make decisions.

4.4 Interactive Approaches Today

The discussed papers of Waters (1984) and G. G. Brown and Graves (1981) are not state of the art any more. As previous sections have shown, current approaches and algorithms provide efficient results for a wide range of requirements. This is especially true for urban areas, where a human scheduler cannot embrace the entire region structure, vehicles, duties and relief points due to its size and complexity. In such areas, an interactive approach will overtax a user by the complexity, and may not lead to better results.

However, in contrast to urban regions, regional, rural areas can enormously benefit from the domain knowledge of a human scheduler, even nowadays. There are different reasons for that. For example, there is the fact that a specific domain knowledge is sometimes very hard to encode entirely in rules. Additionally, like Waters (1984) stated, the ability of overriding constraints in the case of vehicle and duty scheduling depending duty costs and tough find unusual solutions which may be better in the end.

One practical example is timetables which are often grown historically. A lot of small improvements are made to them over years, to be able to create better and cheaper vehicle and duty schedules. The knowledge of such improvements can lead to better results. Another example is a scenario where a very efficient vehicle and duty schedule can be produced, if one or two trips would not be present. An experienced scheduler knows, or recognizes during planning, that these few trips can be ignored for the vehicle and duty scheduling because, for instance, a secretary has also a bus license and drives these trips once a day or week. For this reason, no additional driver is needed and all other trips can be served efficiently and to the satisfaction of the drivers, because they do not have to take long rests or 'worse' duties.

4.5 General Structure and Requirements

To work successfully with an interactive planning system, various requirements have to be fulfilled. An interactive scheduling system basically contains three main parts illustrated in Figure 4.1 and described in the following (Bechara and Galvão, 1984):

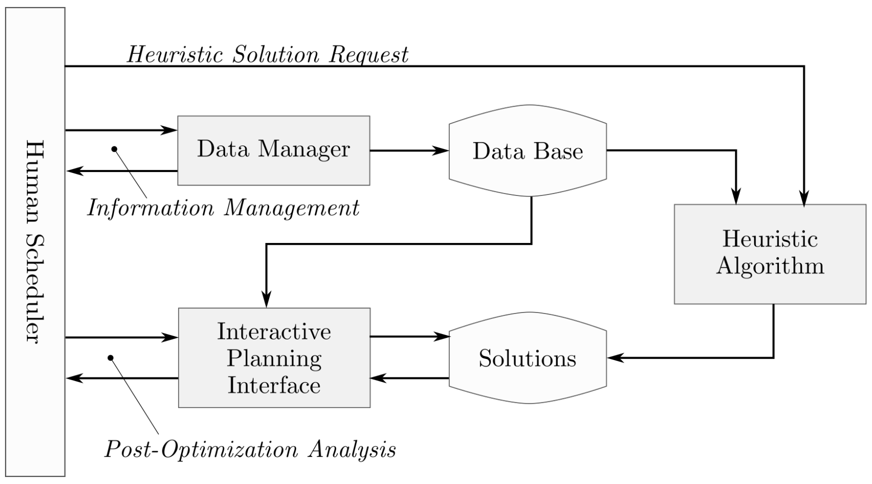


Figure 4.1: An Interactive Scheduling System. Adapted from Bechara and Galvão (1984)

1. A **Data Manager** is responsible for all data included in planning. It provides all information about rules, trips, stop points, solutions, results and their statistics. Additionally, the data manager allows a user to modify properties of trips, stop points or change planning data such as distances between stop points. Furthermore, generated or saved solutions should be available at any time of the scheduling process for comparison of current and existing solutions.

2. **Heuristic Algorithms** are responsible for generating solutions based on trips stemmed from the timetable. The scheduling algorithms should be able to generate initial solutions but also suggest solutions based on existing solutions constructed or improved by a human scheduler.

3. The **Interactive Planning Interface** allows a user to interact with the system to access and modify data that is saved by the data manager, work with the scheduling algorithms and construct or improve solutions suggested by the scheduling algorithms. Furthermore, the interactive planning interface allows a human scheduler to:

   (a) Enter or optimize its own solution manually.

   (b) Compare an initial solution with fully generated solutions by the planning system or a solution that is constructed with the help of suggestions of the scheduling algorithms.

   (c) Iteratively improve a solution based on suggestions provided by the scheduling algorithms.

The interactive planning interface is one of the most important parts of the system. No matter how good the underlying algorithms and the suggestions are, the system will fail if the planning interface cannot present all information and statistics clearly. The system has to enable an easy and convenient way to achieve modifications. Moreover, the scheduling algorithms of interactive planning systems do not even have the goal nor aspire to guarantee optimal solutions. Instead, like D. E. Brown, Marin, and Scherer (1995) stated, " [...] they attempt to employ reasoning that is easily understood and accepted by human users to produce good schedules".

Therefore, an interactive vehicle and duty scheduling system is focused on human schedulers, with their domain knowledge and human solution capabilities, supported by the processing power as well as the evaluation and analysing possibilities of computers.

Chapter 5

Interactive Vehicle and Duty Planning in Practice

5.1 Introduction

In the previous chapters, different solution approaches have been described. Especially, in Chapter 3 different solution approaches and algorithms are discussed, which solve individual and combined problems of the planning process. During the discussion with local bus companies, it turns out that planners of rural areas are not satisfied with the solutions of companies which offer optimization software for vehicle and duty scheduling. The main problem they mentioned was that marked-established optimization software is very expensive and produces solutions, which might be cheap, but cannot be used in practice. One explanation as to why these optimization tools do not work properly in rural areas is that many hidden constraints and requirements exist that are hard or even impossible to entirely implement as rules.

Due to regional planners being dissatisfied with market established solutions, bus companies in regional areas create their schedules manually without the support of scheduling tools. The purpose of this thesis is to introduce an interactive solution, based on the concept of Bechara and Galvão (1984) and Waters (1984). A system which does not have the intention of replacing human schedulers, but to integrate the planners in the scheduling process, to use their domain specific knowledge and support them as much as possible.

5.2 Partner Company

This thesis is written in collaboration with the company *TeleMatrik PTS GmbH*[1]. TeleMatrik develops solutions in the area of public transport and offers therefore different software tools, described in the following:

- **PlanMATRIK** maps the planning process of public transport, introduced in Section 2.1. It handles the basic data management, topology, network and line design, timetabling and it provides modules for vehicle and duty scheduling as well as crew rostering.
- **BusMATRIK** is the on-board computer technology in buses. It is responsible for real time data, ticket machine, ticketing and connection protection.
- **ActionMATRIK** is a software tool to monitor the bus fleet and provides a clear overview of the current operating status and historical data.
- **InfoMATRIK** is a passenger information system. It provides departure information on bus stops, course information on buses and other important information that is needed to inform passengers.

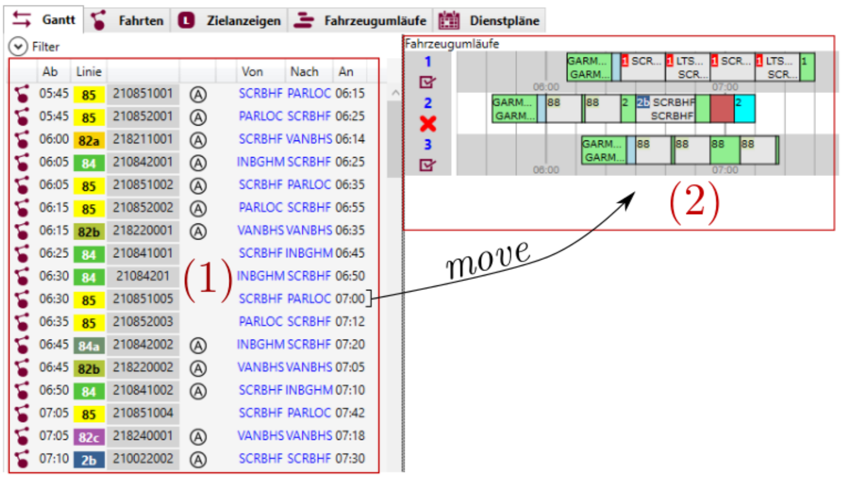
In the current version, the vehicle and duty scheduling tool allows the manual creation of vehicle schedules. In Figure 5.1, the vehicle and duty scheduling module is shown. Timetabled trips have to be selected from a list (1) and must be planned in the vehicle schedule (2) individually. This needs a lot of time and effort to produce efficient schedules. The purpose of this thesis is to improve the vehicle and duty scheduling module of PlanMATRIK.

5.3 Test Regions

To test the combined and interactive vehicle and duty scheduling approach three different regions in rural areas are used. The test regions differ in size and custom requirements. An overview is given in Table 5.1.

Region A is a small and relatively simple region, with almost circular routes and no special requirements. Region B is a medium sized region with only one relief point and one depot. It has various special constraints, because it connects a number of ski resorts and depends on the weather

[1]TeleMatrik PTS GmbH | Public Transport Solutions: http://www.telematrik.com



Ab	Linie			Von	Nach	An
05:45	85	210851001	Ⓐ	SCRBHF	PARLOC	06:15
05:45	85	210852001	Ⓐ	PARLOC	SCRBHF	06:25
06:00	82a	218211001	Ⓐ	SCRBHF	VANBHS	06:14
06:05	84	210842001	Ⓐ	INBGHM	SCRBHF	06:25
06:05	85	210851002	Ⓐ	SCRBHF	PARLOC	06:35
06:15	85	210852002	Ⓐ	PARLOC	SCRBHF	06:55
06:15	82b	218220001	Ⓐ	VANBHS	VANBHS	06:35
06:25	84	210841001		SCRBHF	INBGHM	06:45
06:30	84	21084201		INBGHM	SCRBHF	06:50
06:30	85	210851005		SCRBHF	PARLOC	07:00
06:35	85	210852003		PARLOC	SCRBHF	07:12
06:45	84a	210842002	Ⓐ	INBGHM	SCRBHF	07:20
06:45	82b	218220002	Ⓐ	VANBHS	VANBHS	07:05
06:50	84	210841002	Ⓐ	SCRBHF	INBGHM	07:10
07:05	85	210851004		SCRBHF	PARLOC	07:42
07:05	82c	218240001	Ⓐ	VANBHS	VANBHS	07:18
07:10	2b	210022002	Ⓐ	SCRBHF	SCRBHF	07:30

Figure 5.1: Current version of PlanMATRIK's manual scheduling approach.

conditions and the resulting road conditions in winter. Region C is a proportional big region with longer trips and several relief points and depots, but fewer special constraints.

	#Service Trips	Combined Approach
Region A	276	No
Region B	313	Yes
Region C	514	Partly

Table 5.1: Test Regions Overview

All regions are manually planned by human schedulers. For region A, only a vehicle schedule is available for comparison. Therefore, only vehicle schedules are generated for this region in the practical part to be able to compare the existing solution with a generate solution by the system. Region B uses a fully combined approach. Each tour is planned in a way that one driver is able to drive this tour in one day, regarding all legal regulations and custom requirements. Finally, region C is more difficult for an exact comparison. This region is almost planned as region B, where one driver is able to drive one tour at one day. But there are some exceptions, where a driver changes occur. Therefore, results from the system cannot be fully compared.

5.4 Requirements

In initial discussions with regional bus companies and after internal observations, different basic requirements have emerged which are important for a combined, interactive vehicle and duty scheduling system in the area of local public transport. In the following the requirements are listed and described. Some of the presented ideas have already been discussed in Chapter 5 and overlap with the ideas of Bechara and Galvão (1984) and Waters (1984).

R1 – Suggest Reasonable and Feasible Solutions The system should be able to suggest reasonable solutions from timetabled trips. In this case, reasonable means that all legal regulations and custom rules of a region are considered in a resulting solution. Additionally, it is important that not only one, theoretically very good, solution is provided. This is often the case in market established products. Conversely, multiple reasonable solutions should be generated. This enables a human scheduler to evaluate the suggestions and afterwards select the best suitable solutions or parts of the solution.

R2 – Handle Existing Solutions Another component of the system should be able to handle existing solutions, or parts of existing solutions. An existing solution is either a fully manual schedule, given by a human scheduler, or a solution or solution part generated by the system or fixed by the human scheduler. It must be possible to compare an existing solution to suggested solutions generated by the system. Additionally, it must be allowed to modify existing solutions or let the system suggest further trips based on an initial tour part.

R3 – Fixing Trips After the system has generated solutions, a human scheduler should be able to fix individual trips or parts of the solution, of which the scheduler is satisfied with. Fixed trips are not modified any more by the system, but the system can generate new solutions based on fixed trips and can also connect further trips to fixed trips.

R4 – Fixing Tours Furthermore, a human scheduler can fix tours if he is satisfied with a complete tour. A fixed tour will not be part of the further optimization process, no modifications will be done and no further trips will be added by the system to it.

R5 – Statistics At any time in the scheduling process, the system should provide clear and meaningful statistics of suggested solutions to support human schedulers in their decision-making process. As an example, statistics should include driving and break times, transfer times, paid and unpaid breaks, theoretical costs or the minimum amount of needed buses at any time of day.

R6 – Customizable Rules Each region can have very different requirements on their vehicle and duty schedules. To support the human schedulers as much as possible, it is particularly important to provide a simple, flexible and fast way to implement specially, domain specific requirements as rules and embed them into the system.

The presented requirements have been implemented in the practical part of the thesis. The purpose of this thesis is to verify if an interactive system is still effective today and can lead to nearly as good results as fully computerized approaches do but give a higher level of satisfaction to the human planners in regional areas. If it turns out that the concept works and it will be accepted by bus companies and their schedulers, the system can be refined and extended.

5.5 Incremental Scheduling Process

The introduced requirements in the last section emerged mainly from an interactive scheduling process in mind. The process is illustrated in Figure 5.2 and shows how a solution is incrementally created and improved.

The process starts with an initial solution. This solution is either generated by the system or it it is an existing human planned solution that needs an improvement. After the system has *generated solution suggestions*, a human scheduler can *evaluate the suggestions* based on the graphical results and statistics. Additionally, a user can *compare the current solution* with previous solutions or initial human scheduled solutions. Next, a user can *fix parts* of the solution, with which he is satisfied with or *fix entire tours* to mark them as finished and exclude them from the scheduling process. If the user is satisfied with the modifications, he can *adjust scheduling parameters or custom rules* and let the system make further suggestions. If a user is satisfied with an entire solution after a number of iterations, the solution can be accepted and the scheduling process is stopped.

Although, the scheduling process is an incremental approach, a user should always be able to work with a complete solution, where all trips are scheduled. This has the advantage that a user always has a complete overview and can move any trip at any time to immediately see the effect of any modification.

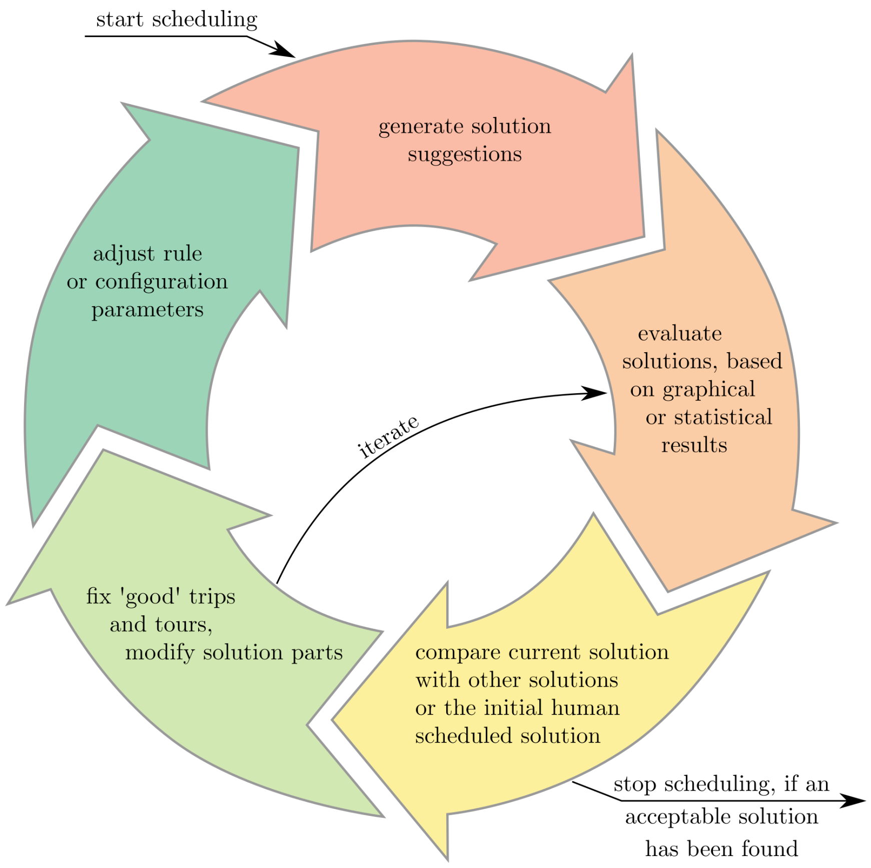

start scheduling
generate solution suggestions
evaluate solutions, based on graphical or statistical results
compare current solution with other solutions or the initial human scheduled solution
stop scheduling, if an acceptable solution has been found
fix 'good' trips and tours, modify solution parts
adjust rule or configuration parameters
iterate

Figure 5.2: Incremental Interactive Scheduling Process.

5.6 Initial Situation in Regional Areas

To enable a smooth interactive scheduling, the computing time of new suggestions must be low. In fact, it is not possible to do a real optimization in a fraction of a second. For this reason, the approach, described in Chapter 6, tries to take advantage of special situations of regional areas, to provide good and feasible solutions in less computation time.

Regional areas often have less relief points than urban areas. Additionally, there are special geographical circumstances and historically grown timetables, which were optimized over years to get better vehicle and duty schedules. Based on these factors, several trip connections are clear in advance and can be scheduled by only considering local costs. An example for an obvious trip connection is for instance a bus line that starts at a main station and ends in a valley. The only reasonable next trip will be the return trip, from the valley back to a main station. With a very high level of probability, there will be no schedule, where the return trip is driven by another bus.

The complexity of reasonable trip connections will be decreased by making assumptions on local situations. Of course, it does not guarantee any optimality, but it will lead to reasonable solutions, which are comprehensible for a human planner, in a very short time.

Chapter 6

Implementation

In this chapter the architecture and details about the implementation of the interactive vehicle and duty scheduling system are described. In the beginning the architecture and the basic concepts are introduced. Furthermore, the used model and the data structures used for optimization are explained in detail. Afterwards, rules representing the legal regulations but also rules needed for additional configuration are discussed. Finally, the implemented algorithms are explained and various algorithm configuration parameters are introduced.

6.1 Basic Concepts

The basic idea of all algorithms shown later in Section 6.4 and the reason why they work and provide reasonable solutions without a comprehensive optimization, is that timetables, which are the input of vehicle scheduling, are not randomly generated data. The timetables are either grown and manually optimized over years. Or they are always created with the problem of vehicle scheduling in mind. Additionally, in local public transport the geographical situation often limits the reasonable possibilities of scheduling trips, described in Section 5.6.

However, good and reasonable results can be achieved by considering local situations and therefore only local costs. Already Valouxis and Housos (2002) take advantage of local situations to build good and reasonable initial solutions. Furthermore, they do not generate their solutions based on real costs, only the time differences between potential connections of trips, plus additional penalties were taken as measure. For the purpose of an interactive vehicle scheduling system, this approach leads to feasible and good enough schedules in a very short time. Therefore, the algorithms introduced in the following are based on the initial solution generation process of Valouxis and Housos (2002).

6.1.1 Internal Scheduling Model

The implemented model used for the optimization is kept as simple as possible. It contains only four classes that include all information needed for scheduling.

Stop Point is a geographical location defined by a coordinate with latitude and longitude. Stop points are needed for transfer time and distance calculations.

Trip is a time block with a departure time, an arrival time, a line number, a trip type, a starting stop point and an ending stop point.

Tour is a sequence of trips. It additionally defines a tour number. In combined vehicle and duty scheduling, a tour is equal to a shift of a bus driver.

Trip Type is crucial for scheduling, rule checking and cost calculation. There are seven different trip types described in the following.

- **Service Trip** is a trip resulting from the timetable and needs to be scheduled.
- **Transfer Trip** is a deadhead trip. It is needed if two trips are connected which do not meet at the same stop point. For instance if trip A ends at stop point s1 and trip B begins at stop point s2, a transfer trip is needed from s1 to s2, if trip A and B gets connected.
- **Standby** is a time where the bus driver is not on a trip. It is a paid break where the bus driver can, for instance, clean the bus.
- **Break** is also a time where the bus driver is not on a trip. A break is an uninterrupted period of time, where a driver is not allowed to do any work. The driver can freely dispose on his time and can be unpaid.
- **Depot Ride** is a deadhead trip. It was described in Section 2.2 as pull-out and pull-in trip. Before starting a tour, a trip is needed from the depot to the first trip of the tour, the pull-out trip. At the end of a tour a trip is needed from the last trip of the tour

back to the depot, the pull-in trip. It is also possible to have depot rides in between the tour, this happens often if a long break is present.

Custom Trip is not resulting from the timetable. It is an additional trip created by the scheduler, for instance for school buses. Trips for school buses often do not stem from timetables.

Other Work is neither a driving trip nor a break. Other work is often used for a set-up time at the beginning of a tour and a cleaning time at the end of a tour.

6.1.2 Connection Matrix

The connection matrix (M) provides a very fast way to check if two trips are connectable, including the local costs of the potential connection. M is a $n \times n$ adjacency matrix, where n is the number of trips to schedule (T). In the example, shown in Figure 6.1, M contains six trips to schedule, $T = \{A, B, C, D, E, F\}$.

	A	B	C	D	E	F
A	-	10	10	10	50	65
B	-	-	-	-	10	15
C	-	-	-	-	-	5
D	-	-	-	-	10	25
E	-	-	-	-	-	-
F	-	-	-	-	-	-

Figure 6.1: An example of a connection matrix.

Each cell $M[r, c]$, where r is the row and c is the column in M, represents a connection between two trips. As an example, trip A can be connected to trip B at the costs of 10 time units. If this connection is taken, trip B will follow trip A in the resulting schedule. Two trips are not connectable if:

1. the arrival time of trip r is after the departure time of trip c,
2. the arrival time of trip r plus the transfer time to trip c is after the departure time of trip c,
3. $r = c$ because a trip cannot be connected with itself.

If one of the above criteria matches, a connection between the trips r and c is not allowed and a negative value is saved at $M[r, c]$. If none of the criteria matches, a connection is feasible and the local costs for the connection are saved at $M[r, c]$. The local costs are calculated as follows:

$$costs = (gap - TT) + TT \times TW$$

where TT is the transfer time, TW is the transfer weight and gap is the time between the arrival time of trip r and the departure time of trip c. Additionally, all trips in M are ordered by their departure time, therefore, the lower left half of M will never contain any feasible connections.

There are several reasons why an independent value for the local costs is taken instead of real costs. The real cost calculation depends on different factors and requires a lot of information in order to be calculated. As an example, breaks can be paid or unpaid depending on the tour progress, how many breaks are already unpaid, or how long the tour duration is at the moment. Additionally, it turns out that if real costs are used for weighing a transfer trip in the generation process, it is far too cheap to minimize the overall transfer trips.

An example of a connection matrix which is generated based on the trips stemmed from a timetable is shown in Figure 6.2.

Timetable

A	6^{00} – 6^{50}
B	7^{00} – 7^{30}
C	7^{00} – 7^{50}
D	7^{00} – 7^{30}
E	7^{40} – 8^{00}
F	7^{55} – 8^{20}

Connection Matrix

	A	B	C	D	E	F
A	-	10	10	10	50	65
B	-	-	-	-	10	15
C	-	-	-	-	-	5
D	-	-	-	-	10	25
E	-	-	-	-	-	-
F	-	-	-	-	-	-

Figure 6.2: A resulting connection matrix from a given timetable.

Each matrix can be visualized as a directed graph or a tree which shows all connections and possible tours. Figure 6.3 visualizes the timetable introduced in Figure 6.2 as a tree structure. Each branch of the tree, represents a potential tour. During the creation of the tree, the legal regulations and other rules have to be checked, to ensure that only feasible tours are generated.

Although the connection matrix is a simple construct, it enables many user interactions and configurations. One example for such a user interaction is 'beaming'. 'Beaming' means that trips are allowed to overlap or that needed transfer times are ignored. For example, if a trip A finishes at 14:02

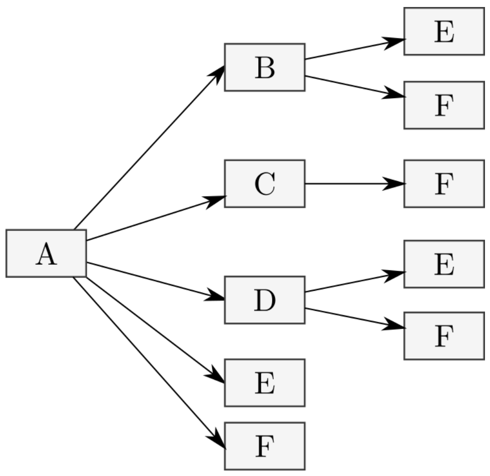


Figure 6.3: This graph represents all possible tours generated from the connection matrix given in Figure 6.2.

and another trip B starts at 14:00 at the same stop point where trip A finishes, no connection will be generated initially. But if a human scheduler knows that this is a very plausible connection, the connection matrix can be changed or overwritten to allow the connection anyway.

Therefore, a bus which serves a tour with trip A and trip B will always begin trip B two minutes late. But it may be no problem because the bus can regain the delay. It would be also possible to change the timetable to eliminate the problem. However, this is may not be possible or not wanted because of, for instance, a 'nice' timetable which will be broken by such adaptions. Small interactions, like the described connection matrix change, are hard or even impossible to automate, but can help an algorithm to generate better results.

The connection matrix is only generated once for each scheduling request. It is a pre-computation procedure and in relation to the overall scheduling process very cheap. Moreover, the more connections that can be marked as not connectable, the faster the scheduling algorithm will work, due to the reduced number of possibilities. Therefore, different rules and configurations are introduced to generate a more sparse connection matrix, described in Section 6.5.

At the beginning, the connection matrix was additionally used to represent scheduled solutions. Every time a suitable trip connection was found, the corresponding $M[r, c]$ cell was marked with a connected flag. All other cells in row r and column c were additionally set to an invalid state, because trips r and c are no longer available for a connection. Therefore, no additional data structure was needed to save a partial or final solution. It was enough to parse the connection matrix and build the final scheduled solution.

In terms of scheduling speed it does not have a significant influence in the beginning. However, when starting with thousands of different permutations, the connection matrix had to be cloned for each individual permutation. This required a tremendous amount of memory($O(p \times n^2)$, where n is the number of trips and p the number of permutations). In order to resolve this problem, a tree structure was introduced to enable fast cloning of tours, with a lower memory consumption.

6.1.3 Solution Data Structure

As already described, the connection matrix was initially used for saving partial or final solutions, beyond their other responsibilities. Because of the tremendous amount of memory needed for thousands of those connection matrices, a more lightweight design was required.

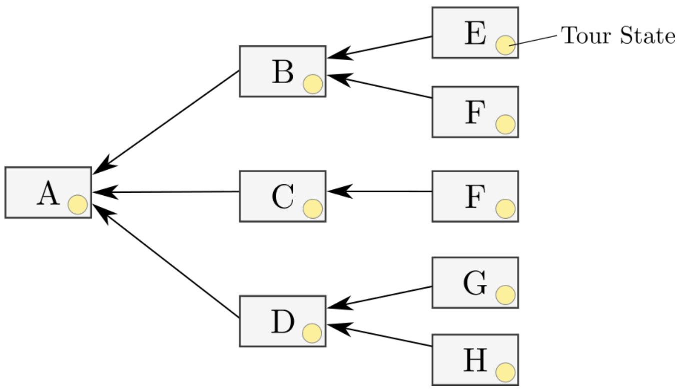


Figure 6.4: This graph represents the solution data structure including tour states.

Therefore a tree structure was implemented to save different tours in a very compact way. In those tour trees, equal tour parts are saved only once. If a tour differs at a node, a new branch is created, which has a reference back to its parent node, but does not need a clone of the existing tour. The only information that must be cloned is the rule state. Each time a connection between two trips is checked against legal regulations and user defined rules, a new rule state is created which includes all information to validate the current tour with an additional trip. The rule state is saved to gain better performance. If a new trip is added, the feasibility of the tour with the additional trip can be checked without revalidating the entire tour. An example of a solution tree is shown in Figure 6.4. However, the validation based on existing rule states only works for tours where a trip is added at the end, shown in Figure 6.5.

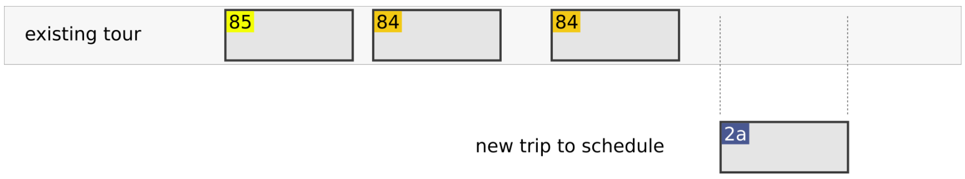


Figure 6.5: Adding a trip at the end of a tour.

If a trip is added in between two other existing trips, as shown in Figure 6.6, a full recalculation of the tour is needed to check if it is valid with the new trip. This is the case if various tour parts are fixed, or manual modifications are done by a human scheduler.

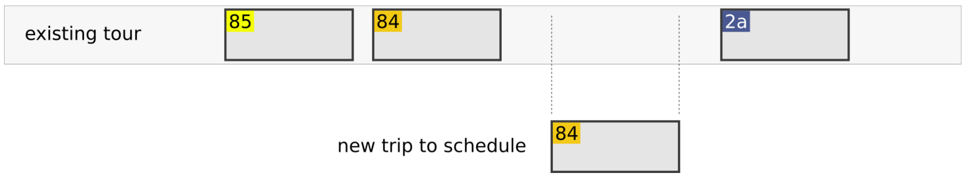


Figure 6.6: Adding a trip in between other trips.

6.1.4 Transfer Matrix

The transfer matrix (T), contains information about the needed times and distances for deadhead trips. It is needed to generate the connection matrix, calculate costs and adding depot trips. T a is fully connected $n \times n$ matrix, where n is the number of stop points, existing in the scheduling region. An example is shown in Figure 6.7.

	S1	S2	S3
S1	0 | 0	500 | 5000	450 | 4000
S2	500 | 5000	0 | 0	440 | 4400
S3	300 | 3000	440 | 4400	0 | 0

Figure 6.7: An example of a transfer matrix, where each cell contains the information of time in seconds and distance in meters (in the form of $time|distance$), needed from one stop point to another.

Each row or column in the matrix represents one stop point. Each trip has two stop points that are relevant for the transfer matrix. This is the starting stop point of a trip and the ending stop point of a trip. All other intermediate stop points of a trip can be ignored. Additionally, the transfer matrix includes all depot stop points, to be able to calculate the connection costs of each trip to each depot and vice versa. The cell $T[r, c]$ where r is the row and c is the column in T hold the information of the transfer time and the transfer distance between stop point r and stop point c.

T is mainly a symmetric matrix and the time and distance needed from one stop point to another is the same as the way back, as it is for S1 and S2. But it is absolutely possible that the needed times and distances are significantly different, as it is for S1 and S3. This results from, for instance, traffic restrictions such as one-way streets which can be used for one direction, but for the way back, another way has to be taken.

6.1.5 Transfer Matrix Generation

The generation of the transfer matrix is based on the Open Source Routing Machine[1] (OSRM) project. OSRM is a high performance routing engine written in C++ and is based on OpenStreetMap[2] data. The OSRM project provides a test server, with which the functionality can be tested easily via a web API. However, it is also possible to install and set-up OSRM on a self hosted server. While developing this project, the OSRM test server was used to get the routing information. Although, the OSRM project has a great functionality, only one API function was required for the purpose of the project. The *table* service of OSRM computes a distance table for given coordinates within one request. Listing 1 shows the table request command described also in the API documentation[3] of the OSRM project. Listing 2 shows an example request to the OSRM table API function.

```
http://{server}/table
                ?loc={lat,lon}
                &loc={lat,lon}
               <&loc={lat,lon} ...>
```

Listing 1: The example shows the structure of the OSRM *table* request, where *loc* defines the position as a coordinate with *lat* the latitude and *lon* the longitude of a stop point.

The response of the OSRM *table* request is shown in Listing 3. It contains a distance table, where the measure of distance is given as travel time in

[1]http://project-osrm.org [2016-02]
[2]https://www.openstreetmap.org [2016-02]
[3]https://github.com/Project-OSRM/osrm-backend/wiki/Server-api [2016-02]

```
http://router.project-osrm.org/table?
            loc=52.554070,13.160621&
            loc=52.431272,13.720654&
            loc=52.554070,13.720654
```

Listing 2: The example shows an OSRM *table* request for three coordinates.

tenths of a second. The first array of the *distance_ table* represents the travel times from the first stop point to all other stop points including itself, where the travel time is zero. The second array represents the travel times from the second stop point to all other stop points, and so on.

```
{
    "distance_table": [
        [
            0,
            31089,
            31224
        ],
        [
            31248,
            0,
            13138
        ],
        [
            31167,
            13188,
            0
        ]
    ],
}
```

Listing 3: The response from the OSRM *table* request

In the current version of the OSRM *table* service, only the travel time between two stop points is calculated but not the distance of the transfer route. The distance is only important for the cost calculation and not for the scheduling of the trips itself. Therefore the distance d is calculated with a default velocity v_d and the travel time t_t:

$$d = \frac{v_d}{t_t}$$

The OSRM service does not support a vehicle type bus as mode of transportation. Therefore, all responded travel times are multiplied with a constant factor to simulate adequate travel times for buses. In the current version there is only one constant for each bus. However, this multiplication factor can be tied to a vehicle type, because it makes a big difference if a small bus or a big bus serves a particular transfer trip. The multiplication factor was calculated by requesting the distance table for different known transfer trips and then taking the mean value of the differences between the OSRM result and the known needed times, which results in a multiplication factor of 1.31.

There are also other services which support the same functionality as OSRM, including distances, like, for instance, Google Maps. But those service often have restrictions, like a maximum number of stop points allowed in one request, or a maximum daily limit of requests. Additionally those services are not open source and cannot be installed on a self hosted server.

In the first version of the transfer matrix, the distance API of Google Maps was used. This works great for small examples. However, when changing to bigger examples, the restrictions were hit quickly. Fortunately, the transfer matrix calculation can be exchanged very easily and the OSRM project replaces the Google service. Additionally, it is possible to change the transfer matrix calculation to some other service, or change it to a self calculated matrix. This data can be taken from known trips which are already in an existing system. Additionally, it is possible to change the needed time and distance between two stop points, for whatever reason.

These possibilities make the transfer matrix transparent, easy to handle and comprehensible.

6.2 Architecture and Scheduling Process

The interactive scheduler system provides an interface which can handle different use cases illustrated in Figure 6.8. First, the scheduler service has to be initialized with all trips of a region that needs to be scheduled, all depots which are defined in this region and all relevant stop points needed by the trips.

After the initializing phase a user can start with the scheduling process of the region. There are different scheduling approaches available. A user can schedule the entire region at once to get an initial drivable schedule or consider only a selected part of a region. Additionally, a user can schedule in forward or backward mode. Then the planner has the option to fix individual

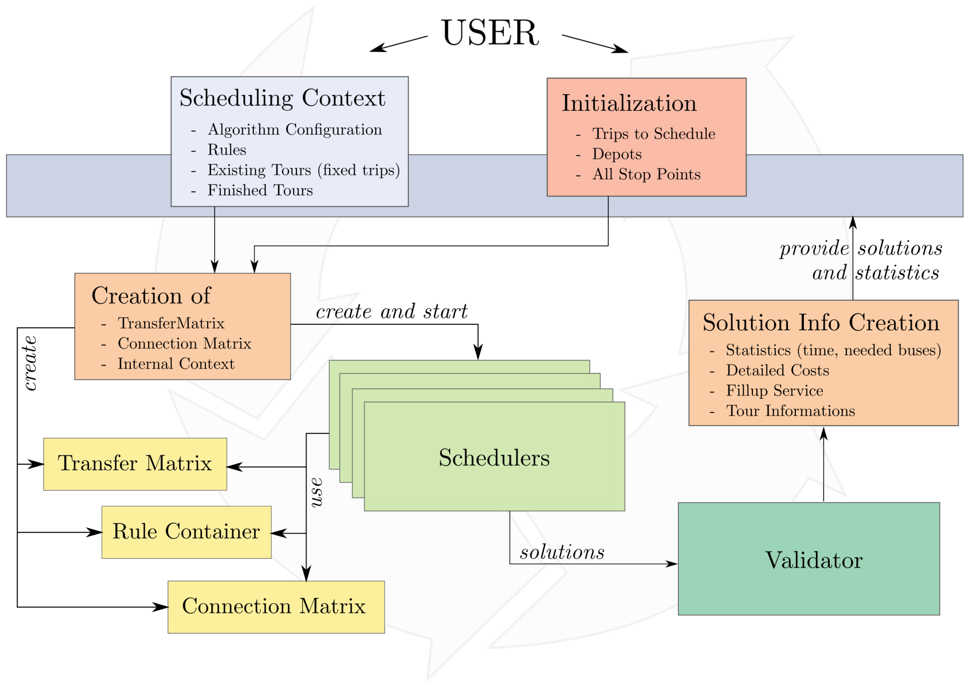


Figure 6.8: Scheduler Service Architecture and Scheduling Process

trips, which means that a human scheduler can predefine parts that are not changed by the algorithm. These fixed parts are not touched by the scheduling algorithm but it will add additional trips at the beginning, at the end or even in the middle of fixed parts if it is possible. However, it is also possible to fix complete tours. Fixed tours will not be touched by the scheduler algorithm, also no additional trips will be added, but costs and statistics are calculated anyway. To change the algorithms' behaviour and adapt them to regional requirements, it is possible to configure the schedulers, which is described in detail in Section 6.5. Finally rules can be defined which check if a tour is valid or not. Different rules like legal regulations or custom rules are discussed in Section 6.3.

When all the needed information has been provided, a transfer matrix, a connection matrix and a rule container are created. These data structures are already described in Section 6.1. Afterwards, the chosen schedule algorithm is created and the scheduling process, based on the configuration, is started. The scheduler heavily uses the connection matrix and the transfer matrix to check if a trip connection is possible and uses the provided rules to test if a newly generated tour is valid and feasible.

After the scheduling process has finished, the generated solutions are validated again by a tour validator. The code in the validator is written as simply as possible and is not optimized for a fast execution, like the rules code. The tour validator is executed once only for each scheduled solution. The purpose of the tour validator is to double check the legal regulations and to ensure that only feasible and legal schedules are produced. If the implemented rules do not contain any failures, the tour validator must not find an error either.

Finally, a solution info is created from the scheduled solutions. This solution info contains statistics of the individual tours and solutions like driving time, break time, needed buses and detailed costs. Additionally, different versions of the scheduled tours, with and without filled up transfer trips and breaks, are provided.

6.3 Scheduling Rules

The scheduling process is influenced by rules defined or selected by a user. In the following, tour validation rules, which mainly implement the legal regulations, introduced in Section 2.3.1, are described. Moreover, other rules like trip insertion rules or cost rules are discussed.

6.3.1 Tour Validation Rules

Tour validation rules can mark a generated tour as valid or invalid. Most of the tour validation rules are implementations of legal regulations, introduced in Section 2.3.1. Two different kinds of tour validation rules exist in the implementation. First, *Stateless Tour Validation Rules* do not have any state and do not store any information. As an input, they get a generated tour and check if it is valid. The second kind of rules belong to *Stateful Tour Validation Rules*. These rules are able to save information about a tour, to avoid an entire revalidation, and therefore be faster, when a new trip is added.

A tour is accepted if all selected rules are valid. If only one rule is violated, a tour is marked as invalid and will not be generated. However, there are rules where only one of two or more rules must be valid. For such cases an OR operator was implemented to fulfil those requirements. An example of the OR operator is shown in Listing 4, where either the 1/6 six rule or the driving period validation rule must be valid, the rest time validation rule must always be valid.

In the following all rules which are used for the current implementation

```
ruleContainer.TourValidationRules.Add(
    new OneSixthValidationRule() |
    new DrivingPeriodValidationRule()
);

ruleContainer.TourValidationRules.Add(
    new RestTimeValidationRule()
);
```

Listing 4: In this example the rest time rule must be valid. Additionally either the 1/6 rule or the driving period rule must be valid but not both.

are described and linked to the legal regulation, if one exists:

Overlapping Rule checks if two trips overlap. If the schedule direction is always forward or backward and no trips are fixed, this rule will always be true because the connection matrix already makes sure that trips cannot overlap. But if a tour contains fixed trips, an overlap between two trips can occur. As shown in Figure 6.9 the trips with the line numbers *85*, *84* and *2a* are fixed trips. The algorithm tries now to add the trip with the number *82*. A connection based on the connection matrix is possible to the trip *84* but it overlaps with the fixed trip *2a*.

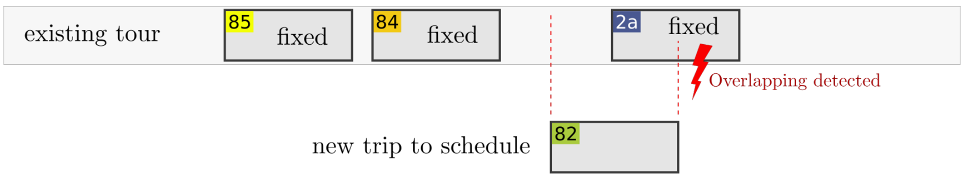


Figure 6.9: Detection of overlapping trips.

The overlapping rule is a stateless validation rule because it is executed only if a tour gets revalidated. As already discussed, a revalidation happens if the saved tour state cannot be used. This is the case if a trip is not placed at the end of a tour.

Total Daily Driving Time Rule checks if the total daily driving time is exceeded. Regulations in the EU, in Austria and in Germany define that the total daily driving time must not exceed nine hours. The total daily driving time rule sums up all driving trip times and checks if the nine hour limit is exceeded. The rule is stateful and saves the driving times for each newly added trip to the rule state. If a new trip is added the saved driving time is only updated with the new driving times but no recalculation of the full tour is required.

Total Tour Duration Rule checks if the total tour duration time is exceeded. The rule does not save any data because it is recalculated every time a new trip is added. The tour duration d is only a subtraction of the arrival time of the last trip, b_{last}, of the tour and the departure time of the first trip, e_{first}, of the tour and therefore very cheap to calculate.

$$d = e_{first} - b_{last}$$

In accordance with the EU regulation (EC) No 561/2006[1], as well as Austrian and German regulations, the total daily rest period has to be at least 11 hours. This leads to a maximal total tour duration or daily operation time of 13 hours.

Driving Period Rule checks the driving time in a driving period. In Austria the maximum driving period duration is 4 hours. After four hours of driving, a driver has to take an uninterrupted break of at least 30 minutes. This break can be replaced by two breaks of at least 20 minutes each or three breaks of at least 15 minutes each. Once a valid break was held, a new driving period is started, regardless of whether the 4 hours driving time is reached or not. An example is shown in Figure 6.10.

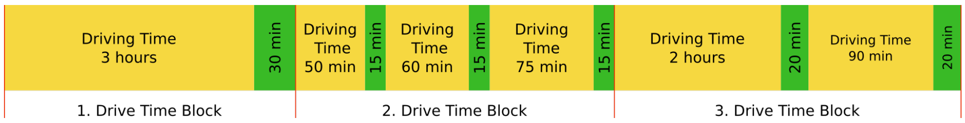


Figure 6.10: In this example valid driving periods are shown. Figure is based on an example in Bagdahn (2015).

The only difference for Germany is the driving period duration. In Germany a driving period of 4 1/2 hours is allowed. The break regulations are the same as in Austria. For this reason this rule is parametrizable with the duration of the driving period. Additionally, a buffer time can be defined to prevent the algorithm from making full use of the driving period. This is very important, especially in Austria, because of the digital speedometer. Each violation of the driving period rule, even only by one minute, leads to significant penalties. Therefore, a buffer time can be defined to make the tours more robust to avoid penalties. A practical example of the usefulness of the buffer time is rough road conditions after snowfall in winter.

[1]`http://eur-lex.europa.eu/resource.html?uri=cellar:5cf5ebde-d494-40eb-86a7-2131294ccbd9.0005.02/DOC_1&format=PDF`

One-Sixth Rule The one-sixth rule is a quotient-rule and is no longer valid for Austria and therefore only relevant for Germany. The one-sixth rule allows interruptions of 10 minutes to count as breaks, if the total break time is at least one sixth of the total driving time. This must also be valid for each driving period. This means that at any time in a route, the one sixth rule has to also be valid within the current driving period, as shown in Figure 6.11.

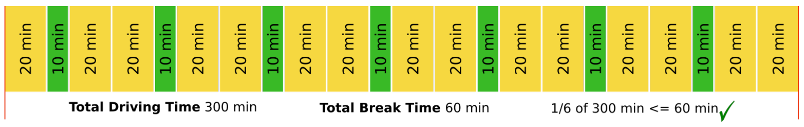


Figure 6.11: This is a valid tour based on the one-sixth rule. The total break time is greater than one sixth of the total driving time and additionally for each time a valid driving period can be found.

It is not allowed to have a long break in the morning, then an uninterrupted or insufficient uninterrupted driving period of more than 4 1/2 hours and then another long break in the evening, like shown in Figure 6.12. Although the total break time is greater than one sixth of the total driving time, at least one driving period is broken.

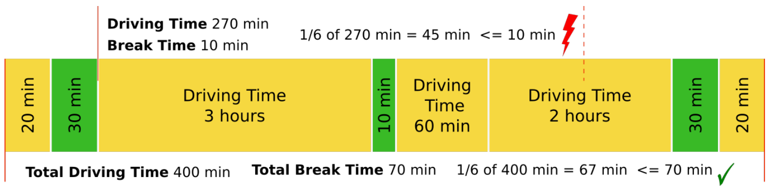


Figure 6.12: In this tour the total break time is greater than one sixth of the total driving time but there exists a driving period where the driving time of the period is not greater or equal than the break time of the period.

Rest Time Rule implements the working hours law (Arbeitszeitgesetz). According to the working hours law, work has to be interrupted by at least 30 minutes for a working time of 6 to 9 hours and at least 45 minutes for a working time greater than 9 hours. However, it is also possible to split the break time into two or three 15 minutes blocks respectively. In this case the first 15 minute block has to be taken in the first 6 hours of the working time block.

This rule only needs to be checked if the one-sixth rule is applied. In the case of the driving period rule, the working hours law is always fulfilled because after 4 or 4 1/2 hours a break of at least 30 minutes has to be taken.

6.3.2 Other Rules

In addition to the tour validation rules there are other rules which extend a generated tour by some trips or replace some trips by others. There are also rules which handle the cost calculation or rules which produce various statistics about generated tours.

Break and Transfer Trip Insertion Rule is a build-in rule which cannot be modified by an external user. If a new trip is added to a tour, the resulting gap between the connected trips is filled with a break. If the connection stop point is not the same, additionally a transfer trip is added, like shown in Figure 6.13.

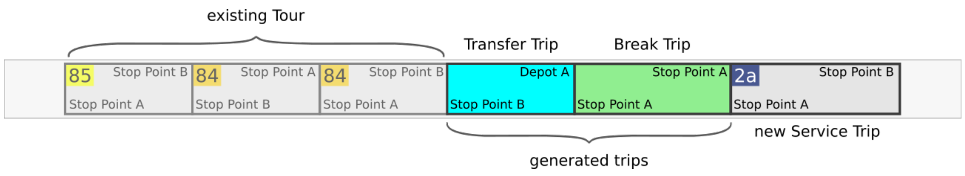


Figure 6.13: If a newly added trip does not have the same departure time as the arrival time of the trip that it gets connected to, a break trip is added to fill the gap. Additionally, if the stop points are different, a transfer trip is added and the remaining gap is filled with a break trip.

Depot Trip Insertion Rule If depots are defined for a region being scheduled, an option can be enabled to automatically connect the border trips, which are the first trip in the tour and the last trip in the tour, to an available depot. The depot insertion is only temporary and is redone every time a new trip is added. In the current implementation depot trips are not directly included in the driving period calculation. If this would be the case, every time an additional buffer time, which is the duration of a depot ride, would be part of the driving period calculation. If the newly added trip is the last trip, it would be correct, but if this is not the case and another trip is added afterwards, it can lead to bad results because of including big buffer times in each driving period calculation.

In the example shown in Figure 6.14, the driving period in tour 2) is not valid with the final depot ride. The tour would only be valid if a trip is connected that produces a break of at least 30 minutes, which will lead to a new driving period. Or a trip is added where the duration of the trip plus the resulting depot ride from the new trip is less than 15 minutes, within a valid driving period duration of 4 hours. In tour 1), the depot ride is added to a new driving period, therefore, a temporary break is inserted and the

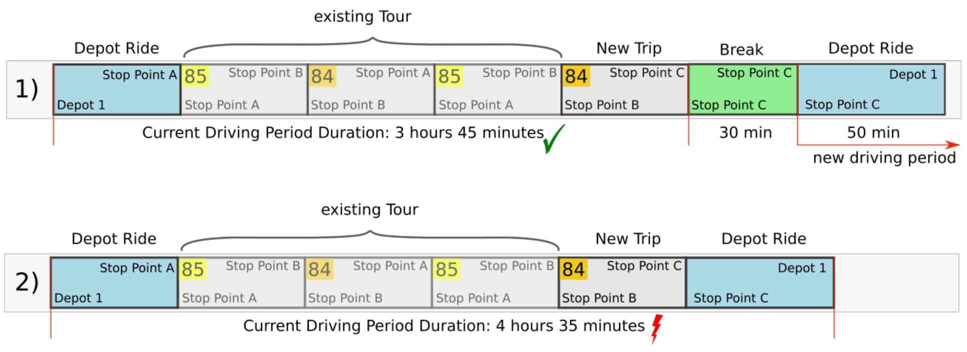


Figure 6.14: In tour 1), the ending depot ride is not added to the current driving period and the new trip is valid. In tour 2), however, the depot ride is added to the driving period and the tour will be invalid.

tour is valid.

Sign-On and Sign-Off Task Insertion Rule is similar to the Depot Trip Insertion Rule. Instead of a transfer trip at the beginning and at the end of a tour, a time block is inserted for the sign-on and sign-off task of a bus. The inserted block is of type *Other Work* and can be a different duration at the beginning and at the end of a tour. Like inserted depot trips, sign-on and sign-off blocks are not directly included in the driving period calculation.

Tour Info Rule is responsible for generating statistics of scheduled solutions and their tours. If no special requirements are needed the build in rule can be used. The build in statistics rule calculates driving times and distances, standby and break times, transfer times and distances, depot ride times and distances, buffer times, sign-on and sign-off times as well as the buses needed to serve all tours.

Cost Rule is responsible for the calculation of appropriate costs of generated solutions. As an input, the entire solution, where all gaps of a tour are filled up with transfer trips and breaks respectively, is given. Therefore each tour can be parsed and the cost can be calculated, based on all region specific requirements. Additionally, the tour information, created by the tour info rule, is provided.

6.4 Scheduling Algorithms

In the following, the implemented algorithms, used for the interactive scheduling process, are described. All algorithms extensively use the connection matrix and the transfer matrix introduced in Section 6.1. Each time one of the algorithms add a trip to the schedule, all provided rules are executed to ensure that only valid tours are generated. All available build-in rules are already discussed in Section 6.3.

6.4.1 Demands on Algorithms

All algorithms must fulfil the requirements, introduced in Section 5.4, to enable interactive scheduling. Therefore, it must be possible to schedule a given timetable from scratch. Additionally, there must be the ability to generate a final solution based on partly existing solutions. This means a user should have the ability to fix individual trips or parts of a tour. The algorithm should then generate a solution based on these fixed trips or intermediate tours. Furthermore, it must be possible to set a scheduled tour to a finished state, so that the algorithm ignores that tour and only calculates statistics of it and validates the tour respectively. Finally, it must be possible to change the schedule direction. This means that a user can start with any trip and then schedule trips that depart later, which is called a forward scheduling, but also schedule trips that depart earlier, which is called a backward scheduling.

6.4.2 Simple

The Simple algorithm does not do any optimization at all. It is a greedy algorithm which takes only pre-calculated local costs, encoded in the connection matrix, and additional configurations made by a user, into account. The basic idea of the Simple algorithm is to connect a new trip, to another trip of a tour, which has already been scheduled, at minimum costs. Listing 5 shows the Simple algorithm procedure.

The Simple algorithm starts with an initial connection matrix and an empty solution. Afterwards, all columns of the connection matrix (M), which represent the trips to schedule, sorted by their departure time in ascending order, are iterated. For each column, all rows, which are the potential trips to connect with, are checked if the connection costs are greater or equal to zero, which represents a possible connection. Additionally, the tour to which a new trip is connected with, is validated against all rules, including

```
var M = GenerateConnectionMatrix();
var solution = new Solution();

foreach column in M
   var possibleTrips = new List();
   foreach row in M.OpenRows

      if M[row, column] < 0
         continue;

      if AreRulesValid( M[row, column] ) == false
         continue;

      possibleTrips.Add(M[row, column]);


   if possibleTrips.Count() == 0
      solution.AddNewTourWithStartingTrip(column);
   else
      var bestTrip = SelectBestTrip(possibleTrips);
      solution.AddTripConnection(bestTrip);
      M.MarkRowAsUsed(bestTrip.Row);

return solution;
```

Listing 5: A simplified pseudo code of the Simple algorithm which always generates a valid and drivable solution.

legal regulations and custom rules. If the trip connection is possible and the resulting tour is valid, the trip connection is added to a possible trips list. If all trip connections (rows) of the current trip (column) are checked, two cases are possible:

1. If the possible trips count is zero, the new trip cannot be connected to an existing tour. Therefore, a new tour will be added to the solution.
2. Else, if there is at least one possible trip connection, the best trip connection, based on local costs, is selected and added to the solution. Additionally, the selected row has to be marked as blocked or used, to exclude it from further connections.

This procedure is repeated until the last column of the connection matrix is reached.

For a better understanding, an example of the Simple algorithm is shown in Figure 6.15, which generates 3 valid tours from a given timetable. In the following example, all rule checks are ignored, which are required in practice to produce feasible tours.

Step 1

	A	B	C	D	E	F
A	-	10	10	10	50	65
B	-	-	-	-	15	20
C	-	-	-	-	-	5
D	-	-	-	-	10	25
E	-	-	-	-	-	-
F	-	-	-	-	-	-

Step 2

	A	B	C	D	E	F
A	×	⑩	×	×	×	×
B	-	×	-	-	15	20
C	-	×	-	-	-	5
D	-	×	-	-	10	25
E	-	×	-	-	-	-
F	-	×	-	-	-	-

Step 3

	A	B	C	D	E	F
A	×	**10**	×	×	×	×
B	-	×	-	-	×	20
C	-	×	-	-	×	5
D	×	×	×	×	⑩	×
E	-	×	-	-	×	-
F	-	×	-	-	×	-

Step 4

	A	B	C	D	E	F
A	×	**10**	×	×	×	×
B	-	×	-	-	×	×
C	×	×	×	×	×	⑤
D	×	×	×	×	**10**	×
E	-	×	-	-	×	×
F	-	×	-	-	×	×

Figure 6.15: This image shows an example the Simple algorithm process. It starts with the initial connection matrix in Step 1 and incrementally generates the solution A $\rightarrow$ B, C $\rightarrow$ F and D $\rightarrow$ E, resulting in Step 4.

Step 1 shows the initial timetable. First, trip A is selected as a tour starting trip, as there are no possible connections available for it, because column A is empty. Next, trip B is connected to trip A in Step 2. It is the only available connection trip for trip B. If a trip connection is selected, all other possible connections, which would allow a connection to trip A, have to be marked as used or blocked (×). Trip C and D do not have any possible connection trips, therefore both trips are starting trips of a new tour. In Step 3, trip E is connected to trip D, because of the lower costs of 10, compared to the connection costs with trip B, of 15. Finally, trip F is connected to trip C in Step 4, because a connection to trip B would be more expensive. By parsing the resulting connection matrix in Step 4, the solution A $\rightarrow$ B, C $\rightarrow$ F and D $\rightarrow$ E can be created.

In the example, the selection of the connection trip was solely based on local costs. However, the selection of the connection trip can be influenced by various configuration parameters described in Section 6.5.

6.4.3 Simple Permutation

The Simple Permutation algorithm is based on the same concept as the Simple algorithm. However, it does not only select one trip of the possible trips pool, it takes all of them and generates a new solution for each possible trip. Listing 6 shows the procedure of generating different solutions.

```
1   var M = GenerateConnectionMatrix();
2   var solutions = new List<Solution>();
3   solutions.Add(new Solution(M));
4
5   foreach column in M
6      foreach solution in solutions
7         var possibleTrips = GetPossibleTrips(solution, column);
8
9         if possibleTrips.Count() == 0
10           solution.AddNewTourWithStartingTrip(column);
11
12        else if possibleTrips.Count() == 1
13           solution.AddTripConnection(possibleTrips.First());
14
15        else
16           solutions.Remove(solution);
17
18           foreach trip in possibleTrips
19              var newSolution = solution.Clone();
20              newSolution.AddTripConnection(trip);
21              solutions.Add(newSolution);
22
23     if solutions.Count() > LIMIT
24        solutions = SelectSubSet(solutions);
25
26  return solutions;
```

Listing 6: A simplified pseudo code of the Simple Permutation algorithm which is able to generate a huge number of cheap solutions which are as different as possible, valid and drivable.

First, a column, which represents a trip to schedule, is selected (line 5) and all possible connection trips are calculated (line 7). In contrast to the Simple algorithm, this step has to be executed for each solution (line 6), because each solution provides different connection trips, which may be part of different tours. When the possible connection trips for a solution and a new trip (column) are calculated, three different cases are possible:

1. If no possible trips are available (line 9), no additional solution is needed and a new tour, starting with the current trip (column) is added to the solution.
2. If exactly one possible trip is available (line 12), there is also no additional solution needed and the trip can be connected to the current tour because it is the only trip that is possible.
3. If more than one possible trip connection is available (line 15), all possible trip connections are generated. Therefore, for each possible trip (row), the current solution is cloned. The trip to schedule (column) is then connected in the cloned solution with the possible trip (row).

In this configuration the algorithm will generate all possible permutations that exist. In fact, the generation of all possible solutions in a reasonable time is not possible. For this reason a selection operator is needed to select a subset of the solutions, generated by the procedure, when a defined limit is reached. However, the selection of intermediate solutions based on local cost only, will not lead to better results than the Simple algorithm already provided. For this reason, the *SelectSubSet* function in line 24 is based on three different characteristics to select intermediate solutions:

Cheap For each newly added trip, three different evaluation criteria are calculated. First, the local costs which are the costs for a newly added connection, that is the waiting time plus the transfer time between two trips. Next, the local area costs, which are the accumulated local costs of a small number of previously added trips. And finally, the intermediate global costs resulting from the cost rule, which are unreliable, especially in the beginning, where a tour only contains a few trips.

Suitable To decide if an intermediate or final solution is suitable, an experienced planner with a comprehensive domain knowledge is needed. Although, a human would be the best choice for this task in theory, a human scheduler will not be able to select the best solutions out of millions. Therefore, this task must be undertaken by computers and is currently based on user defined rules.

Different Tests have shown that if the selection is only based on local and global cost, only a few solutions, which are very similar, dominate the solution space. This follows from the fact, that in the beginning, some intermediate solutions have already become very cheap and effective, which leads to a thorough pruning of other more expensive intermediate solutions. However, as already mentioned, costs can abruptly change and initially expensive solutions can become cheaper in the end and vice versa.

The goal of the selection operator is to choose cheap and effective solutions on the one hand, but also as different as possible solutions on the other hand, to increase the likelihood of entire cheap solutions in the end.

The calculation of local, local area and global costs are already described. The similarity or difference operator is implemented based on the Levenshtein Distance (Levenshtein, 1966). The Levenshtein Distance is a metric to measure the difference between two sequences, for instance strings. The metric is based on the number of *insertions*, *deletions* and *substitutions* required to change one sequence into the other. The higher the number of edits, the bigger the difference between two sequences are. As an example, the Levenshtein distance to change the word "sch**edu**l**e**" into "sch**oo**l" is four, whereas two substitutions ("e" for "o", "d" for "o") and two deletions ("u" and "e") are needed. To change the word "schedule" into "schedule**r**" only one insertion ("r") at the end is needed, which results in a Levenshtein distance of one.

In the case of a vehicle and duty schedule, each tour is a sequence of trips which is identified by a unique number. Based on the generated tours and the identifiers of the trips, the Levenshtein distance is calculated. The higher the Levenshtein distance is, the more diverse two tours are.

6.4.4 Tour Permutation

The Tour Permutation algorithm is not directly based on the Simple algorithm concept. It does not schedule a complete solution, it only generates permutations of tours. The algorithm tries to generate tours with a high density, which means a tour with a high rate of service trips and less breaks, rests and transfer trips. The reason why a cost independent measure is used, has already been discussed in Section 2.3.2 and 6.1.2.

However, the costs of an intermediate tour are not representative, because they can change abruptly with one additional trip. Therefore to use the global cost calculation to compare intermediate tours or solutions could lead to bad results.

Listing 7 shows the procedure of how the Tour Permutation algorithm generates dense permutations of tours.

```
var M = GenerateConnectionMatrix();
var tree = new SolutionTree(startingTrip);

foreach leaf in tree.GetNextLeaf()
   var possibleTrips = new List();
   foreach column in M.OpenColumns

      if M[leaf.Index, column] < 0
         continue;

      if AreRulesValid( M[leaf.Index, column] ) == false
         continue;

      possibleTrips.Add(M[leaf.Index, column]);

   if possibleTrips.Count() == 0
      tree.FinalizeLeaf(leaf);
   else
      tree.ReplaceLeaf(leaf, possibleTrips);

   if tree.IsLevelCompleted(leaf) && tree.LeafCount > LIMIT
      tree.Trim();

return tree.CreateBestTours();
```

Listing 7: A simplified non optimized pseudo code of the Tour Permutation algorithm, which is able to calculate a huge number of dense tours.

In case of the Tour Permutation algorithm, a new trip is not tested against an existing tour and then added to it, as the Simple based algorithms does it. In this case, for an existing trip of a tour, all possible connection trips are calculated and added to a tour solution tree afterwards.

First, a solution tree is initialized with a starting trip. The starting trip is the root node in the tree, or the first leaf in the tree. Afterwards, for each leaf in the tree, all possible connection trips are selected by checking the connection matrix and validating the resulting tours against the defined rules. When all possible trips are calculated, two cases are possible:

1. If a leaf does not have any further possible connection trips, the leaf gets finalized and removed from the next leaf list, to ensure that it is not tested again.
2. Else, if there are trip connections possible, they get connected with the current leaf. All newly connected trips become new leafs and the

current leaf is removed from the leaf list, because it is not a leaf any more.

If a number of leaves are recalculated completely and the leaf count is greater than a defined limit, the tree is trimmed to ensure results are gained in a reasonable time. The trim procedure trims all leaves and its representing tours from the solution tree whose density is low. The algorithm continues until all leaves are finalized.

6.4.5 Reverse Scheduling

One demand to the algorithms was the ability to do a reverse scheduling. To achieve this requirement it is only required to transpose the transfer matrix and start with the last trip of the connection matrix and decrement the index instead of increment it. This has the advantage that a lot of the algorithmic code can be reused. Otherwise, all row and column indices need to be changed. An example of a transposed connection matrix is shown in Figure 6.16.

Connection Matrix

	A	B	C	D	E	F
A	-	10	10	10	50	65
B	-	-	-	-	10	15
C	-	-	-	-	-	5
D	-	-	-	-	10	25
E	-	-	-	-	-	-
F	-	-	-	-	-	-

Transposed Connection Matrix

	A	B	C	D	E	F
A	-	-	-	-	-	-
B	10	-	-	-	-	-
C	10	-	-	-	-	-
D	10	-	-	-	-	-
E	50	10	-	10	-	-
F	65	15	5	25	-	-

Figure 6.16: Transposing the connection matrix for the reverse scheduling.

6.5 Algorithm Configuration

It has already been noted that the algorithms can be influenced by different configurations and weights. All configuration possibilities, discussed in the following, have the advantage that they are tangible for human schedulers and are not seemingly random weight values which can be changed. The configuration parameters are introduced to support the customers' special needs and are used to additionally improve the scheduling results for different kinds of regions with varied requirements.

6.5.1 Line Loyalty

It is often the case that a time table is designed in a way that trips with the same line number are meant to be driven in a row and therefore it might be preferable to schedule them in one tour. For that reason, the line loyalty is introduced to prefer trip connections, where the trips have the same line number. In order to get reasonable results, a time threshold needs to be defined, within which trips should be preferred to other trips. Figure 6.17 shows an example of the line loyalty.

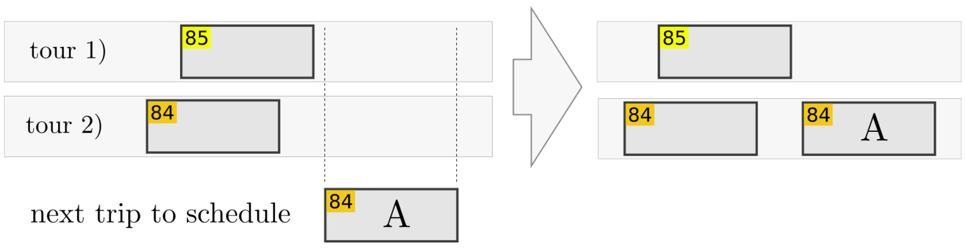


Figure 6.17: Trips with the same line number are preferred within a defined threshold, if the line loyalty option is enabled, although the costs are higher.

Trip A, with the line number 84 has two possibilities to be connected with. Although, the connection costs, resulting from the waiting time of the connection, of trip A to tour 1) are lower, it gets connected to tour 2), because the connection trip has the same line number.

6.5.2 Local Search for Line Loyalty

The local search configuration also takes trips into account that will be scheduled within a defined time threshold after the current trip's departure time. It is therefore only useful if the line loyalty option is enabled and no connection trip with the same line number for the current trip is found. If there exists a trip within the threshold that can be connected to a trip with the same line number of the existing solution, to which also the current trip could be connected with, this connection possibility is blocked for the current trip being scheduled. Figure 6.18 shows an example of the local search process.

Although, the trip connection costs of trip A to tour 2) are lower, trip A is connected to tour 1) because tour 2) is blocked by local search. It is

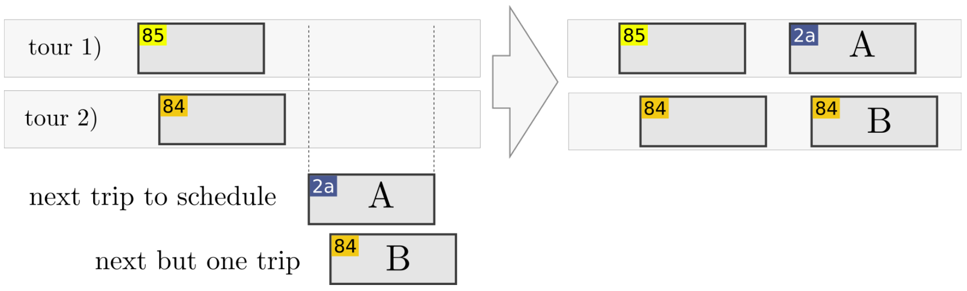


Figure 6.18: With the local search option enabled, trips with the same line number, with departures in a defined threshold, are preferred.

blocked because trip B has the same line number as the connection trip in tour 2. Local search simply forces the line loyalty to a wider range of trips.

6.5.3 Break Reduction

Always using the cheapest trips often leads to big gaps in other tours. This results on the one hand in tours with many long breaks that have to be paid, and on the other hand in tours with only the minimum needed breaks. The resulting unbalanced solutions lead to higher total costs in the end. For this reason the break reduction configuration property was introduced.

The break reduction configuration reduces long breaks and ensures more balanced tours. If the gap between an existing trip and a new trip to schedule gets bigger than a defined threshold, the costs for this connection are reduced and the connection is preferred over a initially cheaper connection. In Figure 6.19, the trip connection costs of trip A to tour 1) is by far less than the connection to tour 2). But with an enabled break reduction, the costs for the connection to tour 2) are reduced, because the gap is greater than the defined threshold. Therefore, trip A is scheduled in tour 2).

The break reduction is implemented as a generation rule for the connection matrix. Therefore, it is a pre-computation and does not influence the scheduling performance. Additionally, the break reduction can be used to generate very different tours, by varying the break reduction threshold.

6.5.4 Maximum Allowed Break Time

The break reduction is a soft constraint which tries to reduce long breaks. If there is no other trip connection possible, long breaks can still occur. The maximum allowed break time configuration does not allow gaps that

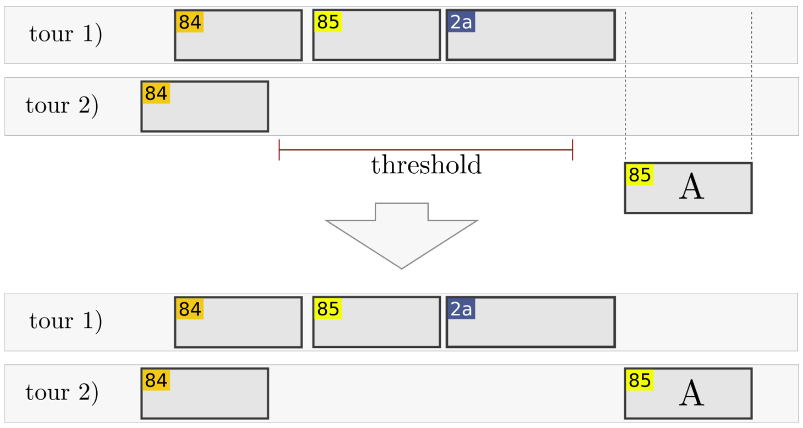


Figure 6.19: The break reduction configuration tries to prevent the generation of tours with very long breaks and ensure more balanced tours.

are greater than a defined threshold and a new tour will be started if no acceptable connection is found. As well as the break reduction, the maximum allowed break time is a pre-computation and modifies the connection matrix. If a gap is longer than a defined threshold, the connection will not be generated in the connection matrix.

The value of the maximum allowed break time can strongly influence the number of total tours. If the maximum allowed break time is set to a small value, the probability of a higher tour count increases, because there are simply less possibilities of connection trips. An advantage of less possibilities is that the time needed for the scheduling decreases.

Setting the break reduction threshold to a higher value than the maximum allowed break time makes no sense, because the connection is not possible anyway.

6.5.5 Break Penalty Time

Depending on the cost calculation, some break durations are very unfavourable. For instance, looking at the Austrian regulation of driving periods, breaks only count, if their duration is greater than 15 minutes. This means a break of 14 minutes is the worst case that can happen, because the 14 minutes are fully paid and the break does not count as a break regarding to the driving period. So it is much better, if breaks are greater than 15 minutes, rather than breaks with a duration of just below 15 minutes.

For this reason, breaks which are less than 15 minutes are penalized, shown in Figure 6.20. Additionally, the penalization does not start at zero minutes because it will lead to many breaks. It starts at five minutes and increases the closer the value is to 15. This configuration is also a pre-computation and changes the connection matrix in a way that the trip connection costs, for trips which have breaks greater than 15 minutes, are cheaper than breaks that are less than 15 minutes.

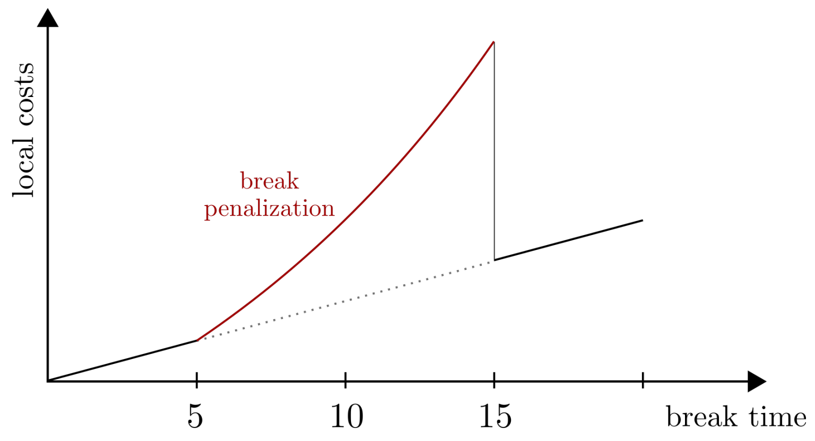


Figure 6.20: Unfavourable break durations are penalized by the Break Penalty Time configuration.

Break penalties can also be defined depending on the time of day. In parts of a day, where the trip distribution is very dense, the break penalty time can be disabled to keep the number of tours as low as possible.

6.5.6 Transfer Weight

The transfer weight is the only configuration parameter which is not as tangible as other configuration possibilities are. The transfer weight defines the multiplication factor of transfer times. This means, if the transfer weight is set to a value of two, the transfer time is twice as expensive as an equal break time. The higher transfer weights are, the rarer transfer trips will be inserted.

6.5.7 Automatic Calculation of Configuration Values

Some configuration parameters have a high impact on the resulting solutions. Therefore single parameters can be automatically calculated by generating a solution with each configuration value. As a result, it is not only the best value which is provided that leads to the lowest costs. Additionally, all generated solutions can be reviewed and ordered by either the solution costs or the needed tours and costs. This enables a human scheduler to decide on which solution the further work should be based on.

Chapter 7

Scheduling Results

In this chapter the results and benchmarks of different test regions of the introduced scheduling system are presented and discussed. Next, a proven scheduling work-flow, which results during testing, is shown. Finally, the feedback from human schedulers and bus companies is discussed.

7.1 Results and Benchmarks

The initial requirements, introduced in Section 5.4, have been fulfilled in the practical implementation. All introduced algorithms can produce reasonable and feasible solutions (R1) from which a scheduler can choose from. It is also possible to work on existing solutions (R2) and improve them with the help of the scheduling system. The system allows a user to fix trips (R3) and fix tours (R4) with a graphical interface. Furthermore, with all suggestions, various statistics (R5) are included and presented in a clear way, next to the solutions. Finally, new rules and configurations (R6) can be added to the system to improve or change the scheduling behaviour.

In the following, the computational results for the three test regions, described in Section 5.3, are presented. All provided results are generated by the Simple algorithm with a maximum wait time of four hours and a varying break reduction threshold from zero to four hours in the interval of one minute. This results in 240 solutions generated in each test run, of which the best is taken, based on the tour count and the total costs of the solution. Furthermore, each test run was executed with different algorithm configurations. Finally, the best solution of each region, generated by the Simple algorithm, was improved by an interactive scheduling process, performed by the author of this thesis, with limited domain knowledge of the regional test regions.

All times in the following tables are in hours:minutes:seconds. Additionally, various short cuts are used to be able to present the computational results in a clear way, explained in the following:

Cost Diff	The Cost difference to the existing solution in percent. A positive value means higher costs.
#Tours	The number of needed tours
#Buses	The number of needed buses
BRT	Break Reduction Threshold that leads to the best solution
Sched.Time	The time needed by the algorithm to generate the solution or the time needed by a human scheduler
DT	Total Driving Time
DDT	Total Deadhead Driving Time
IT	Total Idle Time, Break Time
Paid IT	Paid Idle Time, of the total IT
Unpaid IT	Unpaid Idle Time, of the total IT
Existing	The existing human scheduled solution
S	Simple algorithm
SR	Simple algorithm in backward scheduling mode
BP	Break Penalty option is enabled
LL	Line Loyalty option is enabled
LS	Local Search option is enabled
S LL LS BP	The simple algorithm is used, with the configuration options LL, SS and BP enabled
Interactive	The scheduling process by a human scheduler supported by all capabilities of the system

7.1.1 Region A

Region A is a vehicle schedule only. The results of region A, presented in Table 7.1, show that even the Simple algorithm, with no additional configurations, can generate a solution which has exactly the same values as the human planned schedule. A is a region, where all trip connections are obvious, if only the local area is examined. It therefore follows that region A is very simple in terms of scheduling. This is being strengthened by the fact that different combinations of the configuration parameters do not lead to different solutions. Therefore the results, shown in Table 7.1, only show the statistics of the existing solution and one generated solution, because they are all the same.

	Cost Diff	#Tour	#Buses	Schedule Time
Existing	-	7	7	-
S	0.00%	7	7	130ms

	DT	DDT	IT	Paid IT	Unpaid IT
Existing	84:02:00	00:00:00	11:56:00	11:56:00	00:00:00
S	84:02:00	00:00:00	11:56:00	11:56:00	00:00:00

Table 7.1: Computational results for region A.

7.1.2 Region B

Region B is a more interesting region, where the existing solution is a combined vehicle and duty schedule. The results in Table 7.2 and 7.3 show that the system cannot produce a solution with lower costs than the existing human planned solution.

The best solution can be generated if the options line loyalty and break penalty are enabled and the Simple algorithm has been executed in backward mode. Although, the Simple algorithm only takes care of local factors, it can produce a solution which is only 1.75% more expensive than a solution schedule from a human scheduler. The generated schedule satisfies all legal regulations and is drivable. Of course, the results do not provide any meaningful information about the quality of the solution, nor how satisfied the bus company is about it. The reason for the high scheduling times, needed to produce 240 solutions, in the backward scheduling mode, in contrast to the forward scheduling mode, is that in backward mode no tour state is saved, which means that entire tours are revalidated very often. These big time differences, regarding the scheduling time, show the great advantage of saving the tour state to avoid the revalidation of a tour each time a trip is added.

By using the interactive tools, provided by the scheduling system, the generated solution can be improved by a further 1.9% in 10 minutes by the

	Cost Diff	#Tours	#Buses	BRT	Sched.Time
Existing	-	16	15	-	-
S	3.00%	16	15	00:25:00	00:00:09
S BP	2.90%	16	15	00:25:00	00:00:09
S LL	3.83%	16	15	03:31:00	00:00:09
S LL BP	4.80%	16	15	00:36:00	00:00:09
S LL LS	2.16%	16	15	00:05:00	00:00:18
S LL LS BP	2,08%	16	15	00:26:00	00:00:18
SR	2.92%	17	15	02:31:00	00:01:12
SR BP	2.95%	17	15	00:50:00	00:01:12
SR LL	2.14%	16	15	00:50:00	00:01:08
SR LL BP	1.75%	16	15	01:00:00	00:01:09
SR LL LS	4.43%	18	15	01:14:00	00:04:54
SR LL LS BP	3.88%	18	15	01:29:00	00:04:59
Interactive	-0,15%	16	15	-	00:10:00

Table 7.2: Computational results for region B. (Part 1)

	DT	DDT	IT	Paid IT	Unpaid IT
Existing	117:54:07	04:09:07	51:26:06	31:37:48	19:48:18
S	119:07:58	05:22:58	58:33:01	37:38:08	20:54:53
S BP	119:09:57	05:24:57	58:11:02	37:16:09	20:54:53
S LL	120:20:12	06:35:12	57:28:39	37:17:13	20:11:26
S LL BP	119:05:56	05:20:56	65:20:28	42:54:10	22:26:18
S LL LS	118:30:40	04:45:40	58:49:30	37:21:27	21:28:03
S LL LS BP	117:53:47	04:08:47	59:28:23	38:24:57	21:03:26
SR	121:28:28	07:43:28	50:26:48	31:50:46	18:36:02
SR BP	120:33:11	06:48:11	52:56:59	33:09:41	19:47:18
SR LL	119:16:12	05:31:12	53:37:33	33:59:56	19:37:37
SR LL BP	118:44:31	04:59:31	54:34:41	34:51:05	19:43:36
SR LL LS	121:42:34	07:57:34	53:07:21	34:33:26	18:33:55
SR LL LS BP	121:31:48	07:46:48	51:57:00	33:24:21	18:32:39
Interactive	117:24:53	03:39:53	55:15:59	34:32:33	20:43:26

Table 7.3: Computational results for region B. (Part 2)

author, who is not an expert in planning. Therefore, the improved solution is even better than the existing solution by 0.15% using the same number of needed tours and buses. But what is even more important and interesting, a comparable schedule can be generated and improved by the author of this thesis in minutes, in contrast to the days or even weeks an experienced human planner would need with the original approach.

7.1.3 Region C

Region C is not a fully combined vehicle and duty schedule. The existing, human planned schedule contains tours, which do not satisfy all legal regulations regarding the driving time directive and working time law. Therefore, driver changes are necessary to be able to serve the generated tours. As a result, the existing solution cannot be compared directly to the generated solutions according to the number of tours.

The results of region C, shown in Table 7.4 and 7.5, show that the human planned solution can be beaten regarding costs, in backward scheduling mode with the break penalty option enabled, by 0.43%. But the needed tours, to serve all trips, are 51, which is very high in contrast to 38 tours needed in the human schedule solution. Therefore, the best result, regarding the needed tours and costs, is generated in forward scheduling mode with the options line loyalty, local search and break penalty enabled. The resulting solution is only 1.34% more expensive but still needs seven tours more, than the existing solution.

In contrast to region A and B, the simple algorithm is not able to produce as good solutions as the existing schedule regarding the needed tours. As already mentioned, the number of needed tours cannot be compared directly. Additionally, region C has more trips to schedule and even more importantly it has a different structure with more relevant major hubs where many lines meet. Conversely, Region A and B only have one central hub, where all lines meet sooner or later. All of these factors strongly affect the decision-making process, based only on local circumstances, of the Simple algorithm and lead to visibly worse results.

By using the interactive tool set of the system, the solution can be improved to be 2.47 % better than the existing solution regarding costs. It was not possible for the author to reduce the number of needed tours of 45. But, the number of buses has been reduced to 34 in contrast to 36 needed by the existing solution. However, the size of region C is already an upper bound for an interactive scheduling approach. It was hard for the author to maintain the overview of all trips and tours and perform the introduced

	Cost Diff	#Tours	#Buses	BRT	Sched.Time
Existing	-	38	36	-	-
S	2.35%	45	36	00:20:00	00:00:27
S BP	2.10%	45	36	02:30:00	00:00:30
S LL	2.99%	45	35	00:29:00	00:00:29
S LL BP	4.04%	45	36	01:35:00	00:00:30
S LL LS	2.21%	45	36	02:14:00	00:01:03
S LL LS BP	1.34%	45	35	03:00:00	00:01:05
SR	0.64%	51	36	01:30:00	00:03:37
SR BP	-0.43%	51	35	02:01:00	00:03:41
SR LL	0.30%	51	35	01:33:00	00:03:29
SR LL BP	1.28%	50	36	01:21:00	00:03:33
SR LL LS	0.29%	51	36	01:41:00	00:16:49
SR LL LS BP	0.50%	51	36	02:15:00	00:16:55
Interactive	-2.47%	45	34	-	00:10:00

Table 7.4: Computational results for region C (Part 1)

	DT	DDT	IT	Paid IT	Unpaid IT
Existing	289:33:00	21:23:00	173:25:50	121:26:39	51:59:11
S	295:12:54	27:02:54	182:04:37	130:45:06	51:19:31
S BP	298:34:18	30:24:18	171:54:02	120:35:39	51:18:23
S LL	298:08:21	29:58:21	179:33:04	127:49:13	51:43:51
S LL BP	302:57:47	34:47:47	181:23:33	127:32:33	53:51:00
S LL LS	297:58:58	29:48:58	176:36:54	124:17:33	52:19:21
S LL LS BP	299:48:40	31:38:40	162:01:16	112:01:39	49:59:37
SR	299:51:57	31:41:57	154:56:23	103:24:16	51:32:07
SR BP	300:11:16	32:01:16	143:22:22	92:49:28	50:32:54
SR LL	300:43:21	32:33:21	149:46:06	98:06:11	51:39:55
SR LL BP	300:19:25	32:09:25	159:53:06	107:38:50	52:14:16
SR LL LS	299:34:24	31:24:24	152:13:52	100:16:38	51:57:14
SR LL LS BP	299:39:10	31:29:10	152:31:26	101:44:49	50:46:37
Interactive	303:23:48	35:13:48	135:39:37	90:05:03	45:34:34

Table 7.5: Computational results for region C (Part 2)

iterative and interactive scheduling process. But nevertheless, an acceptable and feasible initial solution can be created in a very short time. An experienced human planner will need several weeks to schedule a region of this size and complexity.

7.1.4 Results Conclusion

The Simple algorithm provides, in all cases, reasonable and feasible initial solutions. It turns out that the outcome of the Simple algorithm cannot beat the existing solutions, but it can be further improved with the interaction of a human scheduler, although the author of this thesis does not have a comprehensive knowledge of the regions.

One early idea was, to generate optimal solutions by planning one tour after another with the support of the Tour Permutation algorithm. The Tour Permutation algorithm suggests best possible solutions for one tour. It turns out that if too many tours are planned as well as possible, the overall result will be worse. This follows from the fact, that other possible tours which have to be generated from the remaining trips, need long paid breaks or transfer trips which are more expensive than the savings resulting from the optimal planned tours. The Tour Permutation Algorithm is a helpful tool to show a scheduler different solution approaches for a tour and plan parts of a schedule as well as possible. However, a scheduler always has to keep the entire solution in mind to get good solutions in the end.

The Simple Permutation algorithm was introduced to generate better solutions using a wider search process and taking into account more solution possibilities than the Simple algorithm. The performance of the Simple Permutation algorithm is not shown in the region results, because it could not generate better solutions than the Simple algorithm with varying configurations. There are two main problems with the Simple Permutation algorithm. First, the selection operators do not work as expected for intermediate solutions or solution parts. Good branches are pruned too early, because they have bad properties in an intermediate state. It is, especially for Austrian regulations, very hard or even impossible to automatically decide if a partial solution is efficient or not. This follows from the fact, that if only one trip is added or removed from a partial solution, it can lead to an abrupt change of costs. Although the difficulties in selection of solutions are known, it was not possible yet to select solution branches, which lead to entire efficient schedules in the end. Second, the solution data structure memory consumption is too high. Only one million intermediate solutions can be maintained in parallel. These leads to pruning of solution branches in an early state and in the end to a similar performance as the Simple algorithm with a higher scheduling duration.

The results have shown that different configuration options and parameters lead to very different results. Although, the results for region B and C can significantly be improved by using the configuration options line loyalty, local search and break penalization, the used break reduction threshold which leads to the best result is very different. This confirms that each region has several varying local factors for which different configuration parameters are needed to be able to generate efficient schedules.

7.2 Proven Scheduling Work-flow

During the testing phase of the interactive vehicle and duty scheduling, a scheduling work-flow has emerged that has proved to be effective. First, an initial schedule is generated and reviewed. It has been proven that if there are only a few night courses, it is best to start the planning process with the last trip. From this starting point, a reverse tour permutation is started, which takes about five seconds for a set of tour suggestions from which the planner can choose from. The advantage of border trips, like night trips, is that they can often be scheduled without impacting other tours because they are detached from other trips. After selecting an efficient tour suggestion, it may be slightly modified and fixed.

After planning the border trips and tours, a new schedule is generated with the Simple algorithm by setting the region specific configuration parameters and start an automatic calculation of the break threshold parameter. Next, efficient parts of the intermediate schedule are fixed and further tour permutations are started and promising tour parts are fixed by the human expert. Finally, the entire solution is incrementally improved by moving, exchanging and fixing trips.

The described proven scheduling work-flow which emerged from a wide range of test runs, overlaps largely with the planned incremental scheduling work-flow introduced in Section 5.5.

7.3 Feedback from Human Schedulers

The presented results in previous sections show the performance of the introduced algorithms, which do not have the intention of generating optimal solutions. The resulting costs are close to the human scheduled solutions, but they do not provide any meaningful information about the quality. Therefore, feedback was collected from bus companies and planners to examine the acceptance and practical usability of the introduced interactive and combined vehicle and duty scheduling concept. The feedback is based on an early version of the presented system which is integrated in PlanMATRIK. The integration process is still in progress. The current version does not support all planning features and needs further user interface improvements.

The idea of an interactive scheduling concept with an iterative scheduling process has been well received. Fixing parts of intermediate solutions and letting the system suggest further possibilities have met with good response. Additionally, it is very helpful for planners to have an initial, fast but drivable version of a region to get an upper bound of buses and drivers required, as well as costs. Every customer has very different requirements, but with the rule system, additional requirements can be implemented very fast, without touching the scheduling algorithms themselves.

Surprisingly, the customers were particularly interested in the statistics module of the system, which was not expected to this extent. The customers liked the live statistics of a region, which immediately change when a trip is moved, added or removed. Additionally, they want to make use of the live statistics of an automatically generated vehicle and duty schedule, resulting from timetable modifications, to optimize their timetables and schedules.

It is not technically an issue to execute the Simple algorithm on the modified timetable and present the results. But the problem at this point is that customers on the one hand explained that a fully automated vehicle and duty scheduling is not possible in their regions, because no practically usable solutions can be generated automatically. On the other hand, regarding the timetable and schedule example, exactly this behaviour was expected by the customers. It has to be pointed out again, that the Simple algorithm can produce feasible and comparable solutions regarding costs in a very short time. But it cannot guarantee optimality at all, especially regarding the quality of the automatically generated schedules. Therefore it should be used carefully, if for instance timetable decision are made based on the statistics of a region, which the Simple algorithm provides, without reviewing the generated solution.

The statistic tool is also very helpful for local authorities or transport associations, which offers public tenderings for their bus regions. They can easily evaluate the received offers on their feasibility and quality. One example of an infeasible offer, of which authorities or transport associations are often concerned with, is that schedules are submitted which need less buses than are needed at a minimum for a region. This offer can therefore be cheaper than a feasible offer of a competitor. The lower bound of needed buses of a region can simply be calculated by counting the trips which overlap and must be driven in parallel. Based on these statistics the local authorities or transport associations can reject infeasible offers early.

The overall feedback is positive, but it also turns out that there is still a lot of work to do, to achieve all requirements of the customers and to release a final version.

Chapter 8

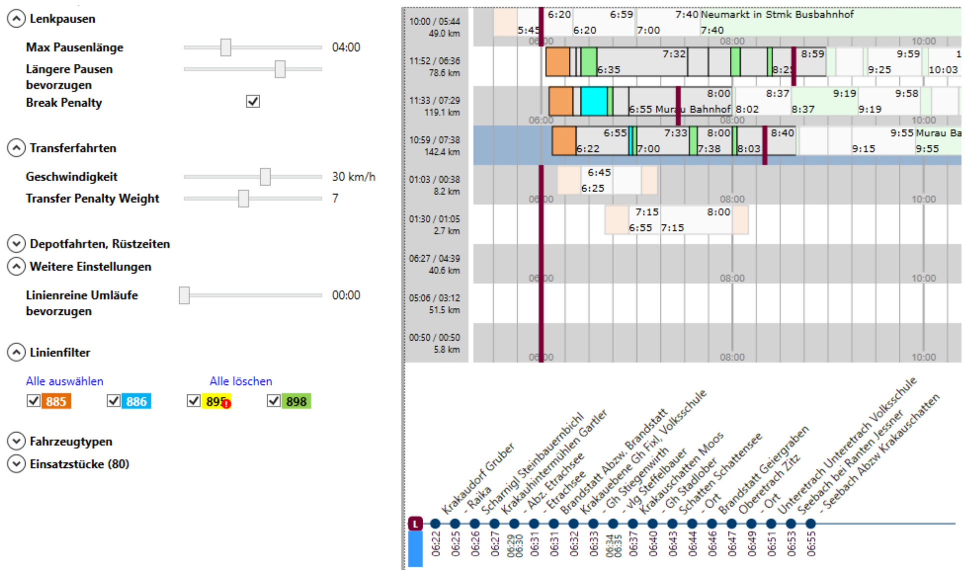
Practical Usage

In this chapter, the practical usage of the implemented system is described. Additionally, missing or not yet completed features are discussed which will be addressed in further developments. Finally, an outlook of the usefulness and practicability of the introduced interactive and combined vehicle and duty scheduling system in practice is given.

8.1 Integration in PlanMATRIK

The interactive scheduling system was developed as a separate and independent solution, outside of PlanMATRIK. It was planned to integrate the new scheduler, if it turns out that the interactive concept works.

At the time this thesis is written, the integration process into PlanMATRIK has already started. The back-end with its algorithms, rule and schedule configuration system have been integrated in PlanMATRIK without any modifications. Changes are only planned for the user interface. The complexity of the configuration parameters will be reduced. Additionally, the process of fixing trips and tours will be simplified. In the first integration step, the interactive scheduler module must be enabled manually and is called *Scheduler Assistant*. All data and solutions can simply be adopted from one module to the other. This enables a user to test the Scheduler Assistant, without losing the familiar planning module or the manually created schedules which are used. Figure 8.1 shows an extract of the Scheduler Assistant module in an early integration state.


Lenkpausen

Max Pausenlänge 04:00

Längere Pausen bevorzugen

Break Penalty ✓

Transferfahrten

Geschwindigkeit 30 km/h

Transfer Penalty Weight 7

Depotfahrten, Rüstzeiten

Weitere Einstellungen

Linienreine Umläufe bevorzugen 00:00

Linienfilter

Alle auswählen Alle löschen

✓ 885 ✓ 886 ✓ 895 ✓ 898

Fahrzeugtypen

Einsatzstücke (80)

Figure 8.1: The interactive system is integrated as a Scheduler Assistant in PlanMATRIK. The configuration parameters can be modified in the left section of the module. The interactive scheduling is done in the right section, where trips can be fixed and changes are immediately visible.

8.2 Further Development

As mentioned in Chapter 7, various parts of the implementation do not work properly yet. As an example, the memory-intensive data structure, which maps the generated solutions must be more efficient without loosing the property of fast cloning. Next, the algorithms can be simplified reducing them to their essentials, without losing functionality. Moreover, scheduling configurations should be introduced to force a wider range of local search to improve the suggestions. Furthermore, the concept of the Simple Permutation algorithm should be reconsidered. To optimize a schedule, an entire solution should be taken into account, instead of trying to generate solutions from scratch and prune early, non promising, intermediate tours.

8.3 Outlook

In addition to the short term further developments, the overall solution quality and interactive scheduling work-flow has to be improved. This improving process has to be done in close collaboration with various bus companies and their schedulers, to ensure a high acceptance of the software. Moreover, during the improving process more custom practices and knowledge of the schedulers and bus companies can be learned and embedded into the software to produce more reasonable solutions and provide an intuitive and familiar scheduling process.

Furthermore, beside the interactive solution suggestions, an over-night optimization, based on an initial solution, is planned. This optimization is not limited in time. Therefore more combinations can be tested, and better initial solutions can be expected. Finally, a combined version with the timetabling phase should be introduced. This combination has emerged from discussions with schedulers, and will allow them to immediately see the effect of modifications on timetables in the vehicle and duty schedules.

Chapter 9

Conclusion

In this thesis an interactive, combined vehicle and duty scheduling approach especially for rural areas was introduced. In the beginning, the planning process in public transport was described and exact as well as heuristic solution approaches for the vehicle and duty scheduling problem were discussed. Moreover, various interactive scheduling approaches from the literature which are based on a profound domain knowledge of human schedulers were described, including different degrees of interaction of a planner and the role of the human planner in the scheduling process.

In the practical part of the thesis, a new interactive scheduling concept was introduced and a detailed description of the implementation was given. Evaluations and benchmarks have shown that the system is usable in practice and it has been well received by planners of various bus companies. The system can generate feasible solutions in a very short time and offers a user interface which enables a human planner to interactively participate in the scheduling process. Beside the interactive possibilities of fixing tours and trips, the system allows a user to adjust various tangible configuration parameters for their custom needs. Moreover, the system provides clear and meaningful statistics to support human planners in their decision-making process. Additionally, the statistics module was helpful for local authorities or transport associations to pre-evaluate offers for local tenderings on its feasibility. But the results have also shown that the interactive concept reaches its limits if a region exceeds a certain size, because it is hard to maintain the overview for a human planner.

Further tests and discussions with bus companies and planners will show the strengths and weaknesses of the concept and the current implementation. The feedback will be used to further improve the introduced system.

Bibliography

Bagdahn, Peter (2015). *Arbeitszeiten im Linienverkehr - Alle Vorschriften für Busfahrten bis 50 km.* 1st edition. huss, 2015.

Bechara, João José B. and Roberto D. Galvão (1984). *Recent Advances in System Modelling and Optimization: Proceedings of the IFIP-WG 7/1 Working Conference Santiago, Chile.* Springer Berlin Heidelberg, 1984. Chapter The use of interactive computing for vehicle routeing, pages 22–32. DOI: 10.1007/BFb0006776.

Bertsimas, Dimitris and John Tsitsiklis (1993). „Simulated Annealing“. In: *Statistical Science* 8.1 (Feb. 1993), pages 10–15. DOI: 10.1214/ss/1177011077.

Brown, Donald E., John A. Marin, and William T. Scherer (1995). *Intelligent Scheduling Systems.* Springer US, 1995. Chapter A Survey of Intelligent Scheduling Systems, pages 1–40. DOI: 10.1007/978-1-4615-2263-8_1.

Brown, Gerald G. and Glenn W. Graves (1981). „Real-Time Dispatch of Petroleum Tank Trucks“. In: *Management Science* 27.1 (Jan. 1981), pages 19–32.

Carpaneto, G. et al. (1989). „A branch and bound algorithm for the multiple depot vehicle scheduling problem“. In: *Networks* 19.5 (1989), pages 531–548. DOI: 10.1002/net.3230190505.

Ceder, Avishai (2011). „Public-transport vehicle scheduling with multi vehicle type“. In: *Transportation Research Part C: Emerging Technologies* 19.3 (2011), pages 485–497. DOI: 10.1016/j.trc.2010.07.007.

Ceder, Avishai (2015). *Public Transit Planning and Operation: Modeling, Practice and Behavior, Second Edition.* 2nd edition. CRC Press, 2015.

Chen, Cheng and Xingquan Zuo (2014). „A multi-objective genetic algorithm based bus vehicle scheduling approach“. In: *Control and Decision Conference (2014 CCDC), The 26th Chinese.* (Changsha, China). IEEE, May 2014, pages 2675–2679. DOI: 10.1109/CCDC.2014.6852625.

Colgan, Lynne, Robert Spence, and Paul Rankin (1995). „The Cockpit Metaphor“. In: *Behaviour & Information Technology* 14.4 (1995). DOI: 10.1080/01449299508914638.

Glover, Fred (1990). „Tabu Search: A Tutorial“. In: *Interfaces* 20.4 (1990), pages 74–94. DOI: 10.1287/inte.20.4.74.

Gurobi (2016). *Mixed-Integer Programming (MIP) - A Primer on the Basics.* 2016. URL: http://www.gurobi.com/resources/getting-started/mip-basics.

Haase, Knut, Guy Desaulniers, and Jacques Desrosiers (2001). „Simultaneous Vehicle and Crew Scheduling in Urban Mass Transit Systems“. In: *Transportation Science* 35.3 (2001), pages 286–303. DOI: 10.1287/trsc.35.3.286.10153.

Kopfer, Herbert and Jörn Schönberger (2002). „Interactive Solving of Vehicle Routing and Scheduling Problems: Basic Concepts and Qualification of Tabu Search Approaches“. In: *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference.* (Hawaii, USA). IEEE, Jan. 2002, pages 1425–1434. DOI: 10.1109/HICSS.2002.994009.

Land, A. H. and A. G. Doig (1960). „An Automatic Method of Solving Discrete Programming Problems“. In: *Econometrica* 28.3 (1960), pages 497–520.

Laurent, Benoît and Jin-Kao Hao (2007). „Simultaneous vehicle and driver scheduling: A case study in a limousine rental company“. In: *Computers & Industrial Engineering* 53.3 (2007), pages 542–558. DOI: 10.1016/j.cie.2007.05.011.

Levenshtein, Vladimir (1966). „Binary codes capable of correcting deletions, insertions, and reversals“. In: *Soviet Physics Doklady* 10.8 (Feb. 1966), pages 707–710.

Mitchell, Melanie (2011). *Complexity - A Guided Tour.* 1st edition. Oxford University Press, 2011.

Schmeidl, Christian (2012). *Handbuch Lenk- und Ruhezeiten Sozialvorschriften (WKO).* 2012. URL: https://www.wko.at/Content.Node/branchen/oe/TransportVerkehr/Handbuch_Lenk-_und_Ruhezeiten.pdf.

Sims, Karl (1991). „Artificial Evolution for Computer Graphics“. In: *SIGGRAPH Comput. Graph.* 25.4 (July 1991), pages 319–328. DOI: 10.1145/127719.122752.

Slany, Wolfgang (1996). „A knowledge-base revision tool for the fuzzy constraints-based *FLIP++ scheduling library“. In: volume 1. New Orleans, LA: IEEE, Sept. 1996, pages 306–312. DOI: 10.1109/FUZZY.1996.551759.

Valouxis, Christos and Efthymios Housos (2002). „Combined bus and driver scheduling“. In: *Computers & Operations Research* 29.3 (Jan. 2002), pages 243–259. DOI: 10.1016/S0305-0548(00)00067-8.

Van den Heuvel, .P.R., J. M. Van Den Akker, and M. E. Van Kooten Niekerk (2008). *Integrating timetabling and vehicle scheduling in public bus transportation.* Technical report. Department of Information and Computing Sciences Utrecht University, Utrecht, The Netherlands, Feb. 2008.

Waters, C. D. J. (1984). „Interactive Vehicle Routeing“. In: *J Oper Res Soc* 35.9 (1984), pages 821–826. DOI: 10.1057/jors.1984.164.

Weider, Steffen (2007). *Integration of Vehicle and Duty Scheduling in Public Transport.* 1st edition. Cuvillier, 2007.

Zuo, X. et al. (2015). „Vehicle Scheduling of an Urban Bus Line via an Improved Multiobjective Genetic Algorithm“. In: *Transactions on Intelligent Transportation Systems* 16.2 (Apr. 2015), pages 1030–1041. DOI: `10.1109/TITS.2014.2352599`.