Wissenschaftliche Arbeit zur Erlangung des Grades
Master of Science
an der TUM School of Management
der Technischen Universität München

A Decomposition Approach to the Hospital-wide Therapist Scheduling Problem

Referent:	Prof. Dr. Rainer Kolisch Lehrstuhl für Operations Management Technische Universität München
Betreuer:	Dipl.-Math. oec. Markus Frey Dipl.-Inform. Med. Daniel Gartner
Studiengang:	TUM-BWL Master
Eingereicht von:	Alexander Döge, B.Sc. Schäufeleinstraße 27B D-80687 München Tel.: +49 (0) 89 / 71674088 Matrikelnummer: 03603409
Eingereicht am:	München, 29. November 2013

When receiving a series of physical treatments, patients prefer to be treated by a therapist they already know. Accordingly, the objective of a therapy scheduler is to maximize the matches of patients and their preferred therapists. At the collaborating hospital in Freising, Germany, physical therapists are scheduled on a daily basis. At the beginning of the day, therapy jobs are known. The necessary decision are when, where and by which therapist a therapy job has to be undertaken. In our case, we consider a general hospital where patients can be treated in both, the hospital's therapy centers (TC) and the hospital's wards. Hence, the scheduler has to take into account walking times between different job locations. Currently, the schedule is developed manually which is time-consuming and far from optimal. To treat this problem we propose two approaches. The first approach interprets the problem as a vehicle routing problem (VRP) with time windows. The second approach decomposes the routing problem employing a Dantzig-Wolfe (DW) decomposition. A modified label correcting algorithm is used to solve the sub-problem (SP). It is examined how efficient these approaches are in terms of solving real-world problems.

Keywords: Therapist scheduling · Column generation · Label correcting algorithm

Table of Contents

List of Figures

List of Tables

Abbreviations

CG	column generation.
DW	Dantzig-Wolfe.
IP	integer program.
L-RMP	linear relaxation of the RMP.
LLF	largest label first.
LP	linear program.
MP	master problem.
RC	reduced cost.
RMP	restricted master problem.
SLL	smallest label last.
SP	sub-problem.
SPP	shortest path problem.
TC	therapy center.
ThSP	therapist scheduling problem.
VRP	vehicle routing problem.

Symbols

Δ^{br}	maximum time interval between br_1 and br_2.
Δ^{end}	minimum time to shift end where no break can be scheduled.
δ_i^{job}	variable indicating if a job i is done within a tour.
$\delta_{r,t}^{\text{room}}$	variable indicating if a room r is occupied in time period t.
λ_0	root label.
Λ_v	set of last nodes for all labels in Λ with $\Lambda_v = \{v_\lambda \mid \lambda \in \Lambda\}$.
λ^*	label representing the best tour found so far.
λ^{BW}	label from $\Lambda^{\text{usefulBW}}$.
λ^{combi}	label that results from the combination of λ^{BW} and λ^{FW}.
λ^{FW}	label from $\Lambda^{\text{usefulFW}}$.
λ^{new}	new label that results from extension.
λ^{parent}	label that has to be extended.
$\Lambda^{\text{unprocessed}}$	set of unprocessed labels/paths.
Λ^{useful}	set of useful labels/paths.
$\Lambda^{\text{usefulBW}}$	set of useful paths from backward iteration.
$\Lambda_v^{\text{usefulBW}}$	set of last nodes for all labels in $\Lambda^{\text{usefulBW}}$.
$\Lambda^{\text{usefulFW}}$	set of useful paths from forward iteration.
$\Lambda_v^{\text{usefulFW}}$	set of last nodes for all labels in $\Lambda^{\text{usefulFW}}$.
$\Theta_{n,i}^{\text{job}}$	equal to one, if in tour n job i is treated, otherwise zero.
Θ_n^{match}	number of matches within a tour n.
$\Theta_{n,r,t}^{\text{room}}$	equal to one, if in tour n room $r \in \mathcal{R}^{\text{tc}}$ is occupied at time t, otherwise zero.
Π	a path.
Π_λ	path so far in a label λ.
π_p^{conv}	dual variables for convexity constraint.
π_i^{job}	dual variable for job assignment constraint.
$\pi_{r,t}^{\text{room}}$	dual variable for TC capacity constraint.
$\mathcal{A}$	set of arcs in $\mathcal{G}$.
$a_{r,i,t}$	equal to one if in room $r \in \mathcal{R}^{\text{tc}}$ job $i \in \mathcal{I}$ is performed at time $t \in \mathcal{W}_i$, otherwise zero.

$\mathcal{A}^{\text{backward}}$	set of arcs for backward iteration.
$\mathcal{A}^{\text{forward}}$	set of arcs for forward iteration.
$\mathcal{A}^{\text{red}}$	reduced set of arcs.
$b_\lambda^{\text{AlreadyBr1}}$	a variable indicating if the first break is already in the path.
$b_\lambda^{\text{AlreadyBr2}}$	a variable indicating if the second break is already in the path.
$b_\lambda^{\text{LongShift}}$	a variable indicating if the path so far represents a long shift.
$b_\lambda^{\text{ShortShift}}$	a variable indicating if the path so far represents a short shift.
br_1	first break.
br_2	second break.
$c_{r,l}$	time to travel from room r to room l.
d_n^{br}	duration of break $n \in \{1, 2\}$.
d_i^{job}	duration of job i.
D^{In}	dummy job in.
D^{norm}	duration of a regular shift.
D^{Out}	dummy job out.
D^{short}	duration of a short shift.
$E_p^{\text{br}_n}$	earliest start time of break $n \in \{1, 2\}$ for therapist p.
$f_{p,i}$	equal to one if therapist $p \in \mathcal{P}$ treats job $i \in \mathcal{I}_p^{\text{Fav}}$, otherwise zero.
$\mathcal{G}$	a graph.
$\mathcal{I}$	set of real-existing therapy jobs.
$\mathcal{I}_\lambda$	set of jobs that still can be done in a label λ.
$\mathcal{I}_r$	set of jobs that can be done in room r.
i_v	job i that is represented by a node $v \in \mathcal{V}$.
$\mathcal{I}^{\text{br}}$	set of jobs including break jobs (br_1, br_2).
$\mathcal{I}_p^{\text{br}}$	set of jobs that can be done by therapist p including break jobs.
$\mathcal{I}^{br,D}$	set of jobs including break jobs and dummy jobs (D^{Out}, D^{In}).
$\mathcal{I}_p^{\text{Fav}}$	set of jobs for which therapist p is favored by the patient.
$\mathcal{I}_{p,i}^{\text{Pre}}$	set of possible predecessor jobs for therapist p and job i.
$\mathcal{I}^{S^{\text{end}}}$	set of jobs representing the end of shift.

$\mathcal{I}^{S^{\text{endNorm}}}$	set of jobs representing the end of a normal shift.
$\mathcal{I}^{S^{\text{endLong}}}$	set of jobs representing the end of a long shift.
$\mathcal{I}^{S^{\text{endShort}}}$	set of jobs representing the end of a short shift.
$\mathcal{I}_i^{\text{Succ}}$	set of possible successor jobs for job i.
$\mathcal{I}_{p,i}^{\text{Succ}}$	set of possible successor jobs for therapist p and job i.
K_r^{tc}	capacity of therapist center room r.
$L_p^{\text{br}_n}$	latest start time of break $n \in \{1, 2\}$ for therapist p.
$L^{\text{unprocessed}}$	a set of unprocessed nodes.
$\mathcal{N}$	set of all possible tours.
$\mathcal{N}_p$	set of all possible tours for therapist $p \in \mathcal{P}$.
n^{new}	a new shift to be added to $\mathcal{N}$.
$\mathcal{P}$	set of therapists.
$\mathcal{P}_i$	set of therapists who can perform job i.
$\mathcal{R}$	set of rooms.
$\mathcal{R}_i$	set of rooms in which job i can take place.
r_v	room r that is represented by a node $v \in \mathcal{V}$.
$\mathcal{R}^{\text{tc}}$	set of TC rooms.
RC_λ	accumulated reduced cost in a label λ.
$\overline{RC}$	average reduced cost of $\Lambda^{\text{unprocessed}}$.
RC_n	reduced cost for a given tour $n \in \mathcal{N}$.
$\overline{RC}^{\text{new}}$	$\overline{RC}$ after a new label enters or leaves $\Lambda^{\text{unprocessed}}$.
$\overline{RC}^{\text{old}}$	$\overline{RC}$ before a new label enters or leaves $\Lambda^{\text{unprocessed}}$.
RC_v	proportional reduced cost of a node v.
$\mathcal{S}_p^{end}$	possible end times of shift for therapist p.
$\mathcal{S}_p^{St}$	start time of shift for therapist p.
$\mathcal{T}$	set of time-intervals.
t^{sep}	separation time to divide graph.
t_v	time t that is represented by a node $v \in \mathcal{V}$.
$t_{i,r}^{\text{earliest}}$	earliest possible start time for a given job-room-combination (i, r).
$t_{i,r}^{\text{latest}}$	latest possible start time for a given job-room-combination (i, r).
$\mathcal{U}$	set of direct successors of $v \in \mathcal{V}$ which have the same job-room-combination.

$\mathcal{V}$	set of vertexes/nodes in $\mathcal{G}$.
v_λ	last node of a path in label λ.
v^{IN}	dummy node representing the sink of $\mathcal{G}$.
$\mathcal{V}_v^{\text{Pred}}$	set of predecessor nodes for node v.
$\mathcal{V}^{\text{red}}$	reduced set of vertexes.
v^{OUT}	dummy node representing the source of $\mathcal{G}$.
$\mathcal{V}_v^{\text{Succ}}$	set of successor nodes for node v.
$\mathcal{W}_i$	set of time-intervals when job i can start.
$w(\Pi)$	weight of a path defined as the sum over all arc weights.
$w(\Pi)^*$	minimum path weight.
$x^p_{i,j,r,l}$	equal to one if therapist $p \in \mathcal{P}$ treats job $i \in \mathcal{I}^{\text{br}} \cup \{D^{\text{Out}}\}$ in room $r \in \mathcal{R}_i$ before job $j \in \mathcal{I}^{\text{Succ}}_{p,i}$ is treated in room $l \in \mathcal{R}_j$, otherwise zero.
$y^{\text{end}}_{p,t}$	equal to one if therapist $p \in \mathcal{P}$ ends his shift at time $t \in \mathcal{S}^{\text{end}}_p$, otherwise zero.
z_n	equal to one if tour $n \in \mathcal{N}$ is used, otherwise zero.
$z^p_{i,r,t}$	equal to one if therapist $p \in \mathcal{P}$ treats job $i \in \mathcal{I}^{\text{br,D}}$ which is started in room $r \in \mathcal{R}_i$ at time $t \in \mathcal{W}_i$, otherwise zero.

1 Introduction

In 2012, non-surgical therapies represented, besides surgeries, the majority of hospital treatments in Germany (Statistisches Bundesamt (2013)). Mainly elderly people require physical therapies and their share of the entire population is rising (Statistisches Bundesamt (2010, 2011)). Thus in future, the demand for physical therapies will further increase. This strongly emphasizes the need for scheduling physical therapies effectively and efficiently.

At the collaborating hospital in Freising, Germany, physical therapists are scheduled on a daily basis. At the beginning of the day, therapy jobs are known. The necessary decisions are when, where and by which therapist a therapy job has to be undertaken. Patients prefer to be treated by a therapist they already know. Accordingly, the objective of a therapy scheduler is to maximize the matches of patients and their preferred therapists. The reasons for this objective are twofold. First, the hospital's mission is to provide the best possible treatment for each patient (Klinikum Freising (25.11.2013)). Second, hospitals in Germany face a hard competition. More than half of all hospitals make losses (Blum et al. (2013)) and in densely populated areas such as the greater Munich area patients can easily change to other hospitals. However, the hospital expects that satisfied patients are more likely to return to the hospital when they are in need of care again. Therefore, the hospital tries to maximize the patients satisfaction by maximizing the number of matches of patients and their preferred therapist.

The planning in Freising is hospital-wide. That means that in contrast to rehabilitation hospitals the scheduler has to decide whether a job has to be done in the therapy center (TC) or at a ward. The TC consists of differently equipped rooms where patients can go for a treatment. Out-patients can only be treated at the TC while in-patients can be treated at the TC or at wards or both. An example for in-patients who can only be treated at wards are patients staying in the intensive care unit. Hence, the scheduler has to take into account walking distances between different job locations. This approach provides more flexibility but increases the problem's complexity, too.

Apart from the routing decision and patients preferences the scheduler has to consider the following things while planning:

- Each therapy job has to be done.
- Therapists have different qualifications.
- Therapists have different working patterns (different shift start and shift length).

- Part-time and full-time worker are available.
- TCs have limited capacity.
- Breaks must be assigned to shifts in accordance with the German law.

Currently, the scheduling is done manually and takes approximately one hour each day. The process is time consuming and it is unknown how close the derived solution is to optimality.

The aim of this thesis is to develop an algorithm that solves the therapist scheduling problem (ThSP) to optimality as efficient as possible. Further, the aim is to examine if such an algorithms is able to solve real-world instances at most within one hour.

The ThSP can be classified as offline operational resource capacity planning according to the health care planning matrix of Hans et al. (2012). That means that the ThSP uses planning decisions that were derived at a higher planning level. For example, the number of jobs is planned at the strategic level and the number of therapists and their shift schedules at a tactical level. For the ThSP those information are considered deterministic.

There is a rich literature on planning decisions in health care. A recent review can be found in Hulshof et al. (2012) who classified more than 450 papers. It appears that the hospital-wide view of the ThSP was not treated yet. Either physical therapies are scheduled where patients have to visit a TC or medical personnel is scheduled where patients are visited. The first aspect of scheduling physical therapies considers rehabilitation treatment planning where a patient has to undergo a series of treatments. In general, the main aim is to provide appointment schedules that are more efficient than the current solution in terms of patients waiting time and/or resource utilization (see e.g. Chien et al. (2008), Braaksma et al. (2012), Griffiths et al. (2012) and Schimmelpfeng et al. (2012)). The second aspect of routing medical personnel occurs mostly in home care applications. Home care can be defined as the process of delivering services such as medical, paramedical and social services to patients at their home (Lanzarone et al. (2010)). Recent contributions to this topic have been made by Bard et al. (2013a), Bard et al. (2013b), Mankowska et al. (2013), Liu et al. (2013) and Shao et al. (2012). Especially the articles by Bard et al. (2013a), Bard et al. (2013b) and Shao et al. (2012) are highly relevant to the ThSP since they focus on the routing of therapists.

However, to the best of our knowledge the possibility that therapists can visit patients and vice versa has not been treated yet. Only in the very recent work of Ceselli et al. (2013) a similar setting occurs. They study the problem of distributing drugs or vaccines in a given region. There are two possibilities how citizen can receive the drugs: either they are delivered by a vehicle or the citizens pick up the drugs at a distribution center. This is similar to the ThSP where patients can be visited by a therapist or visit the TC. However, the main difference is that the citizens are grouped in advance. Then for each group it is decided whether the drugs are de-

livered by vehicles or provided by a distribution center. In the ThSP no groups are created in advance. For each patient who can be treated at the TC and ward the decision where the patient is treated is made separately . For out-patients who have to be treated in the TC and in-patients who have to be treated at wards no choice exist. However, those groups interact with each other since they are using the same resources such as TC rooms and therapists. In contrast to that the groups in the setting described in Ceselli et al. (2013) are separated. Thus, it seems that the ThSP treats a problem that has no yet been investigated.

The remainder of this thesis is structured as follows. In section 2, the problem setting is explained in detail. Moreover, it is described how the real-world problem can be abstracted and an integer program (IP) formulation for the ThSP is provided. In section 3, a DW decomposition is developed in order to provide an efficient algorithm to the ThSP. Section 3.1 describes the master problem (MP) and the course of the column generation (CG) algorithm. In section 3.2, the SP is detailed. It is described why the SP can be seen as a shortest path problem (SPP) and how the underlying graph can be reduced. Moreover, a modified bidirectional label correcting algorithm is presented to solve the SP. In section 4, a numeric study is given. The performance of the different approaches is tested. We examine how efficient the means to accelerate the algorithm are. In addition, we discuss how the algorithm could possibly be speed up further. The work closes with a conclusion and outlook in 5.

2 Formal Problem Statement

The following section describes the ThSP in a formal manner. In 2.1 it is explained what information is needed to solve the problem and how it is derived. In 2.2 an IP is developed. This IP can be seen as an exact mathematical description of the ThSP.

2.1 Problem Definition

The ThSP contains three main elements: (1) the therapists, (2) the therapy jobs and (3) the hospital layout. Those elements must be clearly defined to derive a feasible schedule for the ThSP.

(1) Therapists
Each therapist has a certain number of skills and a certain working pattern. The skills define what kind of jobs a therapist can perform. For example a special training is required to perform reflex-zone therapies while all therapists can perform massages. In this work, skills are abstracted as qualifications. If a therapist has a certain qualification (s)he has a certain amount of skills. A therapist with higher qualifications has all skills of the lower qualifications and at least one more skill. For the ThSP we define 5 levels of qualification with lowest qualification 0 and highest qualification 4. Thus, a therapist with qualification 4 can perform all jobs, a therapist with qualification 3 all jobs without the ones requiring qualification 4 and so on.

Next to the qualification a therapist has a certain working pattern. The working pattern defines the start of a shift, possible shift ends and the time windows when a break can be taken. The type of shift that is assigned to a therapist is planned at a higher tactical level and known at the beginning of the therapist scheduling. The hospital in Freising employs full-time and part-time worker. Part-time worker work 4 hours a day and receive no break while full-time worker work 8.5 hours regularly. This includes a break of 30 minutes. However, it is possible that full-time worker work overtime. Then the maximum shift length expands to 11.75 hours and a second break of 15 has to be assigned.

The break assignment must respect national law.[1] Thus, a break can start earliest 2 hours after the shift start and end latest 1 hour before the shift end. It is not allowed that a therapist works more than 6 hours without a break. If (s)he works between 6

[1] The regulations can be found in §9 Arbeitszeitgesetz (ArbZG). [§9 labor hours act] (In German)

and 9 hours a break of at least 30 minutes must be given. For a working time that exceeds 9 hours at least 45 minutes of breaks must be given in total.

To define the break time windows, different shift types have to be defined. A shift is called a short shift if it does not exceed 4 hours. A shift is called a normal shift if it exceeds 4 hours but not 8.5 hours. And a shift is called a long shift if it exceeds 8.5 hours. Those times correspond to the possible shift ends of full-time and part-time worker at the hospital.

It would be legal to split the breaks in blocks of 15 minutes. However, for the ThSP there will only be at most two breaks. A short shift receives no break. A normal shift receives a break of 30 minutes and a long shift receives a break of 30 minutes and a break of 15 minutes. For the sake of simplicity it is defined that the 30 minutes break must precede the 15 minutes break. Therefore, the 30 minutes break is called first break or break 1 and the 15 minutes break is called second break or break 2. The resulting break time windows can be seen in Figure 2.1.

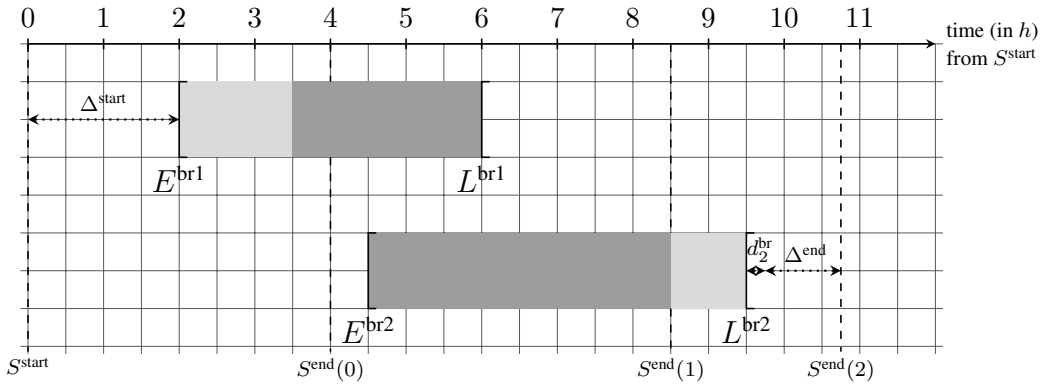

Figure 2.1: Example break time windows.

The shift depicted in Figure 2.1 starts at 0 (S^{start}) and can end after 4, 8.5 or 10.75 hours ($S^{\text{end}}(0), S^{\text{end}}(1)$ or $S^{\text{end}}(2)$ respectively). The earliest start for the first break is defined by E^{br1} and the latest by L^{br1} and for the second break by E^{br2} and L^{br2}. E^{br1} is defined by the minimum time between shift start and first break which is two hours. L^{br1} is defined by the fact that after 6 hours there must be a 30 minutes break. The second break must end one hour before the latest shift end. E^{br2} is the earliest possible point in time where it is still guaranteed that there is not more than 6 hours of consecutive work.

Nevertheless, the break time windows are not sufficient to guarantee a legal schedule. If the shift is a long shift and both breaks were assigned in the light gray areas the time of consecutive work could exceed 6 hours. The scheduler has to keep that in mind or the break time windows must be narrowed down to the dark gray areas.

(2) Therapy jobs

Besides therapist the therapy jobs are a central element of the planning process. Each job requires a certain qualification, can be started within a pre-defined time

window in a set of rooms and takes a certain time to be performed. Furthermore, there is a set of therapists by which the job is preferably done. The qualifications are the same as defined for the therapists. A job has to be done by exactly one therapist. If a job represents a group therapy it is still considered as one job. There is no case where more than one therapist is needed for one treatment. The ThSP is an individual scheduling problem and does not contain crew scheduling decisions.

The modeling of real-existing jobs as described above is straightforward. However, also breaks are modeled as jobs. Those are jobs that can be done but not necessarily have to be done (depending on the shift length). The required qualification is 0 and the time window is the entire planning horizon. Note that there is a difference between the start time window for the break jobs and the break time window for the therapists. A therapist can take a break only in the limited break time window. However, depending on the possible shifts of all therapists a break could occur during the entire planning horizon.

In addition, two dummy jobs are introduced. A dummy job representing the shift start and a dummy job representing the shift end. Those jobs can be done by every therapist and the start time window is again the entire planning horizon. The dummy jobs are not necessarily needed to describe the problem but they will be used in the model in section 2.2 and the are the starting point for solving the SP in section 3.2.

(3) Hospital layout
The hospital layout is the last part that must be defined in order to be able to solve the ThSP. The hospital layout is an abstraction of the real-world hospital. It contains information on the rooms and the walking distances between the rooms. Information on the rooms are the type of the room and the room's capacity. Three different types can be distinguished. Ward rooms, TC and the break room. The break room works as a dummy room. A shift is started and ended in the break room and all breaks are done in this room. TC rooms are the only rooms where a capacity info is needed. If a patient is treated in a ward room this is always the room where (s)he already recovers. Thus, a patient can never be assigned to somebody else's ward room. As said before the break room is a dummy room. Only TC rooms can be overbooked if too many therapies are scheduled at the same time in the same room.

The information that were described above are sufficient to solve the ThSP. An example for a very small instance is provided in order to show how therapists, therapy jobs and hospital layout work together. The instance contains 2 therapist, 3 real-existing jobs and 4 rooms (T2-J3-R4). First, all data is listed and then one optimal schedule is shown.

The planning interval is 15 minutes. Thus, one day is split into 96 time-intervals (0,...,95). The data for working patterns and therapists can be found in Table 2.1 and Table 2.2. Both therapists have the same working pattern with id=0 and the qualification 2. A shift starts in period 28 (7 o'clock) and possible shift ends are in period 44, 62 and 71. The break time window for the first break is E_{br_1}=36 to

Table 2.1: Working pattern (instance T2-J3-R4).

Working Pattern				
id	Shift Start	Shift Ends	E_{br_n}	L_{br_n}
0	28	$44, 62, 71$	$36, 46$	$54, 66$

Table 2.2: Therapists (instance T2-J3-R4).

Therapists		
id	qualification	working pattern
0	2	0
1	2	0

L_{br_1}=52 and for the second break 46-66.

The data for the jobs is found in Table 2.3. The qualification for a job with id i is denoted by q_i, the duration of i by d_i^{job}, the rooms where i can take place by r_i and the starting time window by $\mathcal{W}_i$. Further, the job type is specified and the preferred therapist for the job is listed. If no information on the job is given the job is a real-existing job. 14. If no preference information is given no preference for a therapist exists

The information for rooms is provided in Table 2.4. The room type and the room's capacity is stated. Capacity is only relevant to TC rooms. Hence, no information for other room types is provided. The travel time matrix which lists the time needed to walk from room r to room l is stated in equation 2.1.

$$c_{r,l} = \begin{pmatrix} 0 & 5 & 3 & 2 \\ 5 & 0 & 4 & 2 \\ 3 & 4 & 0 & 6 \\ 2 & 2 & 6 & 0 \end{pmatrix} \tag{2.1}$$

An optimal schedule is depicted in Figure 2.2.

Table 2.3: Jobs (instance T2-J3-R4).

Jobs						
id	q_i	d_i^{job}	r_i	$\mathcal{W}_i$	type	pref
0	2	11	$1, 2, 3$	36		0
1	2	10	1	50		1
2	2	4	$2, 3$	32		
3	0	2	0	0-95	$br1$	
4	0	1	0	0-95	$br2$	
5	0	0	0	0-95	D^{Out}	
6	0	0	0	0-95	D^{In}	

Table 2.4: Rooms (instance T2-J3-R4).

Rooms		
id	type	capacity
0	break room	
1	ward	
2	TC	1
3	TC	1

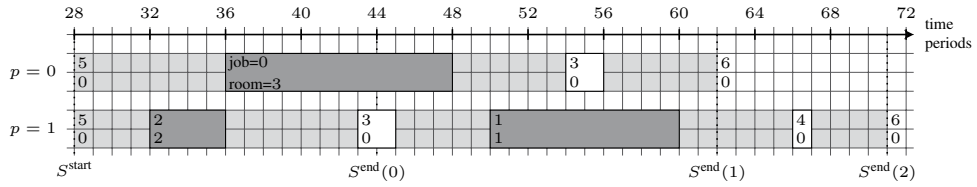

Figure 2.2: Example of optimal schedule (T2-J3-R4).

Light gray areas mark the time when a shift is active. Dark gray boxes mark jobs that are performed by a therapist and white marked boxes represent breaks. The first number in the box stands for the job's id and the second number for the room's id. Both therapists start their shifts in period 28. Therapist $p = 0$ has a normal shift ending in period 62 and therapist $p = 1$ has a long shift ending in period 71. The overall number of matches is two since therapist 0 performs job 0 and therapist 1 performs job 1. For this tiny case all jobs are done by the preferred therapists.

It must be noted that more than one optimal schedule exist for this instance. There is a lot of flexibility to move the break jobs. For instances where regular jobs have bigger time windows as well it is also likely that they can be moved. Nevertheless, the solution represents an optimal schedule which respects all working time regulations.

2.2 Integer Programming Formulation

The IP that is developed in this section serves two purposes. First, it provides an exact mathematical definition of the ThSP and second, it is an approach that can be used to solve problem instances of a size which cannot be solved by hand anymore.

The central idea of the ThSP is that apart from the assignment of therapists to jobs also a routing decision is included. Thus, a natural way would be to model the ThSP as a routing problem. The ThSP indeed can be seen as a special case of the vehicle routing problem (VRP) with time windows.

A shift can be interpreted as a tour and the set of therapists can be interpreted as a very heterogeneous fleet of vehicles where each vehicle can only perform one tour. The fleet is very heterogeneous because each therapist is different in terms of the

jobs for which (s)he is the preferred therapist due to the patients preferences. Only one tour can be performed since it is not possible to perform more than one shift within one day. The break room can be seen as the depot where the vehicles start their tour.

The main difference to classic VRPs[2] is that a job is not bound to a certain location but can be done in more than one location. Thus, the ThSP can be seen as a generalization of the VRP. However, it must not be mixed up with the generalized VRP. The generalized VRP is the problem of finding a route from the depot to a number of predefined clusters of nodes. The node sets are mutually exclusive and it is sufficient to visit only one node of each cluster (Pop et al. (2012)). Hence, even though both problems are a generalization of the VRP they address very different settings.

In the remainder of this section the IP for the ThSP is detailed. First, all necessary sets, parameters and decision variables are defined. Then the mathematical model is stated, followed by the explanation of the objective function and the constraints.

Sets

$\mathcal{P}$	=	set of therapists.
$\mathcal{P}_i$	=	set of therapists who can perform job i.
$\mathcal{I}$	=	set of real-existing therapy jobs.
$\mathcal{I}^{\text{br}}$	=	set of jobs including break jobs (br_1, br_2).
$\mathcal{I}^{br,D}$	=	set of jobs including break jobs and dummy jobs (D^{Out}, D^{In}).
$\mathcal{I}_p^{\text{br}}$	=	set of jobs that can be done by therapist p including break jobs.
$\mathcal{I}_{p,i}^{\text{Pre}}$	=	set of possible predecessor jobs for therapist p and job i.
$\mathcal{I}_i^{\text{Succ}}$	=	set of possible successor jobs for job i.
$\mathcal{I}_{p,i}^{\text{Succ}}$	=	set of possible successor jobs for therapist p and job i.
$\mathcal{I}_p^{\text{Fav}}$	=	set of jobs for which therapist p is favored by the patient.
$\mathcal{I}_r$	=	set of jobs that can be done in room r.
$\mathcal{R}$	=	set of rooms.
$\mathcal{R}_i$	=	set of rooms in which job i can take place.
$\mathcal{R}^{\text{tc}}$	=	set of TC rooms.
$\mathcal{T}$	=	set of time-intervals.
$\mathcal{W}_i$	=	set of time-intervals when job i can start.
$\mathcal{S}_p^{end}$	=	possible end times of shift for therapist p.
$E_p^{\text{br}_n}$	=	earliest start time of break $n \in \{1, 2\}$ for therapist p.
$L_p^{\text{br}_n}$	=	latest start time of break $n \in \{1, 2\}$ for therapist p.

Parameters

D^{Out}	=	dummy job out.

[2] A classification of VRPs can be found in Eksioglu et al. (2009).

Parameters (continuation)

D^{In}	=	dummy job in.
$\mathcal{S}_p^{St}$	=	start time of shift for therapist p.
d_i^{job}	=	duration of job i.
$c_{r,l}$	=	time to travel from room r to room l.
K_r^{tc}	=	capacity of therapist center room r.
D^{short}	=	duration of a short shift.
D^{norm}	=	duration of a regular shift.
d_n^{br}	=	duration of break $n \in \{1, 2\}$.
Δ^{end}	=	minimum time to shift end where no break can be scheduled.
Δ^{br}	=	maximum time-interval between br_1 and br_2.

Decision variables

$x^p_{i,j,r,l}$	=	equal to one if therapist $p \in \mathcal{P}$ treats job $i \in \mathcal{I}^{\text{br}} \cup \{D^{\text{Out}}\}$ in room $r \in \mathcal{R}_i$ before job $j \in \mathcal{I}^{\text{Succ}}_{p,i}$ is treated in room $l \in \mathcal{R}_j$, otherwise zero.
$y^{\text{end}}_{p,t}$	=	equal to one if therapist $p \in \mathcal{P}$ ends his shift at time $t \in \mathcal{S}_p^{\text{end}}$, otherwise zero.
$z^p_{i,r,t}$	=	equal to one if therapist $p \in \mathcal{P}$ treats job $i \in \mathcal{I}^{\text{br,D}}$ which is started in room $r \in \mathcal{R}_i$ at time $t \in \mathcal{W}_i$, otherwise zero.
$a_{r,i,t}$	=	equal to one if in room $r \in \mathcal{R}^{\text{tc}}$ job $i \in \mathcal{I}$ is performed at time $t \in \mathcal{W}_i$, otherwise zero.
$f_{p,i}$	=	equal to one if therapist $p \in \mathcal{P}$ treats job $i \in \mathcal{I}_p^{\text{Fav}}$, otherwise zero.

$$\text{Maximize} \sum_{p\in\mathcal{P}} \sum_{i\in\mathcal{I}_p^{\text{Fav}}} f_{p,i} \tag{2.2}$$

subject to

$$\sum_{r\in\mathcal{R}^{\text{tc}}} \sum_{t\in\mathcal{W}_i} z^p_{i,r,t} = f_{p,i} \qquad \forall\, p \in \mathcal{P}, i \in \mathcal{I}_p^{\text{Fav}}, \tag{2.3}$$

$$\sum_{p\in\mathcal{P}_i} \sum_{j\in\mathcal{I}^{\text{Pre}}_{p,i}} \sum_{r\in\mathcal{R}_j} \sum_{l\in\mathcal{R}_i} x^p_{j,i,r,l} = 1 \qquad \forall\, i \in \mathcal{I}, \tag{2.4}$$

$$\sum_{i\in\mathcal{I}_p^{\text{br}}} \sum_{r\in\mathcal{R}_i} x^p_{D^{\text{out}},i,0,r} = 1 \qquad \forall\, p \in \mathcal{P}, \tag{2.5}$$

$$\sum_{i\in\mathcal{I}_p^{\text{br}}} \sum_{r\in\mathcal{R}_i} x^p_{i,D^{\text{in}},r,0} = 1 \qquad \forall\, p \in \mathcal{P}, \tag{2.6}$$

$$\sum_{j_1 \in \mathcal{I}_{i,p}^{\text{Pre}}} \sum_{l \in \mathcal{R}_{j_1}} \sum_{r \in \mathcal{R}_i} x^p_{j_1,i,l,r} - \sum_{j_2 \in \mathcal{I}_{i,p}^{\text{Succ}}} \sum_{r \in \mathcal{R}_i} \sum_{l \in \mathcal{R}_{j_2}} x^p_{i,j_2,r,l} = 0 \qquad \forall\, p \in \mathcal{P}, i \in \mathcal{I}_p^{\text{br}}, r \in \mathcal{R}_i, \tag{2.7}$$

$$\sum_{p \in \mathcal{P}_i} \sum_{r \in \mathcal{R}_i} \sum_{t \in \mathcal{W}_i} z^p_{i,r,t} = 1 \qquad \forall\, i \in \mathcal{I}, \tag{2.8}$$

$$z^p_{D^{\text{Out}},0,S^{\text{St}}} = 1 \qquad \forall\, p \in \mathcal{P}, \tag{2.9}$$

$$\sum_{t \in \mathcal{S}_p^{\text{end}}} y^{\text{end}}_{p,t} = 1 \qquad \forall\, p \in \mathcal{P}, \tag{2.10}$$

$$\sum_{t \in \mathcal{S}_p^{\text{end}}} t \cdot y^{\text{end}}_{p,t} = \sum_{t \in \mathcal{T}} t \cdot z^p_{D^{\text{in}},0,t} \qquad \forall\, p \in \mathcal{P}, \tag{2.11}$$

$$\sum_{j \in \mathcal{I}_{p,i}^{\text{Pre}}} \sum_{l \in \mathcal{R}_j} x^p_{j,i,l,r} = \sum_{t \in \mathcal{W}_i} z^p_{i,r,t} \qquad \forall\, p \in \mathcal{P}, i \in \mathcal{I}_p^{\text{br}} \cup \{D^{\text{in}}\}, r \in \mathcal{R}_i, \tag{2.12}$$

$$\sum_{j \in \mathcal{I}_{p,i}^{\text{Succ}}} \sum_{l \in \mathcal{R}_j} x^p_{i,j,r,l} = \sum_{t \in \mathcal{W}_i} z^p_{i,r,t} \qquad \forall\, p \in \mathcal{P}, i \in \mathcal{I}_p^{\text{br}} \cup \{D^{\text{out}}\}, r \in \mathcal{R}_i, \tag{2.13}$$

$$\sum_{t \in \mathcal{W}_i} \left(t + d_i^{\text{job}}\right) \cdot z^p_{i,r,t} + c_{r,l} - \sum_{t \in \mathcal{W}_j} t \cdot z^p_{j,l,t} \leq |\mathcal{T}| \cdot (1 - x^p_{i,j,r,l}) \qquad \forall\, p \in \mathcal{P}, i \in \mathcal{I}_p^{\text{br}} \cup \{D^{\text{Out}}\}, j \in \mathcal{I}_{p,i}^{\text{Succ}}, r \in \mathcal{R}_i, l \in \mathcal{R}_j, \tag{2.14}$$

$$\sum_{i \in \mathcal{I}_r} a_{r,i,t} \leq K_r^{\text{tc}} \qquad \forall\, r \in R^{\text{tc}}, t \in \mathcal{T}, \tag{2.15}$$

$$\sum_{\tau=t}^{t+d_i^{\text{job}}-1} a_{r,i,\tau} \geq \sum_{p \in \mathcal{P}} d_i^{\text{job}} \cdot z^p_{i,r,t} \qquad \forall\, r \in \mathcal{R}^{\text{tc}}, i \in \mathcal{I}_r, t \in \mathcal{W}_i, \tag{2.16}$$

$$\sum_{t \in \mathcal{S}_p^{\text{end}}} t \cdot y^{\text{end}}_{p,t} - S_p^{\text{st}} - D^{\text{short}} - \sum_{t=E_p^{\text{br1}}}^{L_p^{\text{br1}}} z^p_{br_1,0,t} \cdot |\mathcal{T}| \leq 0 \qquad \forall\, p \in \mathcal{P}, \tag{2.17}$$

$$\sum_{t \in \mathcal{S}_p^{\text{end}}} t \cdot y^{\text{end}}_{p,t} - S_p^{\text{st}} - d_1^{\text{br}} - D^{\text{norm}} - \sum_{t=E_p^{\text{br2}}}^{L_p^{\text{br2}}} z^p_{br_2,0,t} \cdot |\mathcal{T}| \leq 0 \qquad \forall\, p \in \mathcal{P}, \tag{2.18}$$

$$\Delta^{\text{end}} + d_1^{\text{br}} \leq \sum_{t \in \mathcal{S}_p^{\text{end}}} t \cdot y^{\text{end}}_{p,t} - \sum_{t=E_p^{\text{br1}}}^{L_p^{\text{br1}}} t \cdot z^p_{br_1,0,t} \qquad \forall\, p \in \mathcal{P}, \tag{2.19}$$

$$\Delta^{\text{end}} + d_2^{\text{br}} \leq \sum_{t \in \mathcal{S}_p^{\text{end}}} t \cdot y^{\text{end}}_{p,t} - \sum_{t=E_p^{\text{br2}}}^{L_p^{\text{br2}}} t \cdot z^p_{br_2,0,t} \qquad \forall\, p \in \mathcal{P}, \tag{2.20}$$

$$\Delta^{\text{br}} + d_1^{\text{br}} \geq \sum_{t=E_p^{\text{br2}}}^{L_p^{\text{br2}}} t \cdot z^p_{br_2,0,t} - \sum_{t=E_p^{\text{br1}}}^{L_p^{\text{br1}}} t \cdot z^p_{br_1,0,t} \quad \forall\, p \in \mathcal{P}, \tag{2.21}$$

$$\sum_{t=E_p^{\text{br1}}}^{L_p^{\text{br1}}} (|\mathcal{T}| - t) \cdot z^p_{br_1,0,t} - \sum_{t=E_p^{\text{br2}}}^{L_p^{\text{br2}}} (|\mathcal{T}| - t) \cdot z^p_{br_2,0,t} \geq 0 \quad \forall\, p \in \mathcal{P}, \tag{2.22}$$

$$a_{r,i,t} \in \{0,1\} \quad \forall\, r \in \mathcal{R}^{\text{tc}}, i \in \mathcal{I}_p, t \in \mathcal{W}_i, \tag{2.23}$$

$$f_{p,i} \in \{0,1\} \quad \forall\, p \in \mathcal{P}, i \in \mathcal{I}_p^{\text{Fav}}, \tag{2.24}$$

$$x^p_{i,j,r,l} \in \{0,1\} \quad \forall\, p \in \mathcal{P}, i \in \mathcal{I}_p^{\text{br}} \cup \{D^{\text{Out}}\}, j \in \mathcal{I}_{p,i}^{\text{Succ}}, r \in \mathcal{R}_i, l \in \mathcal{R}_j, \tag{2.25}$$

$$y^{\text{end}}_{p,t} \in \{0,1\} \quad \forall\, p \in \mathcal{P}, t \in \mathcal{S}_p^{\text{end}}, \tag{2.26}$$

$$z^p_{i,r,t} \in \{0,1\} \quad \forall\, p \in \mathcal{P}, i \in \mathcal{I}_p^{\text{br,D}}, r \in \mathcal{R}_i, t \in \mathcal{W}_i \tag{2.27}$$

The objective is to maximize the number of jobs that have been done by the favored therapist. Therefore, the objective function (2.2) sums up the $f_{p,i}$ for all Therapists $p \in \mathcal{P}$ and their corresponding favored jobs $i \in \mathcal{I}_p^{\text{Fav}}$.

The constraints can be separated into 7 parts: (a) link between time-stamps and favored jobs constraint, (b) tour constraints, (c) time-stamp constraints, (d) link between tour construction and time-stamp constraints, (e) TC room capacity constraints, (f) break constraints and (g) variable domains.

(a) Link between time-stamps and favored jobs constraint

Constraint (2.3) links time-stamps and favored jobs. Whenever a job is done by a therapist who is the preferred therapist for this job $(x^p_{i,r,t} = 1)$ the variable $f_{i,p}$ is set to 1.

(b) Tour constraints

Constraint (2.4) ensures that each real-existing job is visited exactly once. This excludes break and dummy jobs. Constraints (2.5) and (2.6) impose for each therapist that exactly one tour is started and finished, respectively. In addition, it is assured that the starting and ending takes place in the break room. Constraint (2.7) preserves the flow. A certain job-room combination (i, r) including break jobs can only be reached if it is left again in direction to some other job-room combination.

(c) Time-stamp constraints

Every real-existing job gets exactly one time-stamp (2.8). The time-stamp for the tour start is the beginning of the shift S^{St} (2.9). Constraint (2.10) ensures that each shift has exactly one shift end. And (2.11) assures that this shift end is also the time-stamp of the tour end.

(d) Link between tour construction and time-stamp constraints

If a job i is done in room r after a job j was done in room l then one time-stamp must be assigned to job i (2.12). Constraint (2.13) works similarly. If a job i is done in room r and job j will follow in room l then a time-stamp must be assigned to that job. One of the model's central constraints is (2.14). It assures that 2 consecutive jobs i and j maintain the interval needed to get from the location r where i is performed to the location l where j is performed. Furthermore, it also works as the sub-tour elimination constraint. Since the duration of all non-dummy jobs is bigger than 0 for all job pairs (i, j) one of the jobs gets a bigger time-stamp. Thus, it is impossible that there is a cycle of jobs where job i has a predecessor which is a successor of one of i's successors.

(e) TC room capacity constraints

Constraints (2.15) and (2.16) collaborate. Constraint (2.15) ensures that the capacity of a TC room cannot be exceeded and (2.16) links jobs and TC room occupancy.

(f) Break constraints

For each therapist the shift length is calculated by the difference between shift end and shift start. If this shift length is bigger than the duration of a short shift the first break must be assigned to that shift (2.17). If the shift length is bigger than the normal shift length less the duration of break 1 also the second break must be assigned to that shift (2.18). Constraints (2.19) to (2.21) ensure the minimum interval among the breaks, and between the breaks and the possible shift ends. If break 2 is assigned to the shift constraints (2.22) enforces break 2 to succeed break 1.

(g) Variable domains

Constraints (2.23) to (2.27) define the domains for all decision variables.

3 Dantzig-Wolfe Decomposition

The routing model from section 2.2 can only be used to describe the ThSP. Even for solving small instances it needs considerable time. In general, VRPs are hard to solve (Lenstra and Kan (1981)) and the ThSP is a generalization of the VRP which makes it even harder to solve.

To overcome this obstacle a DW decomposition is developed in this section. Dantzig and Wolfe (1960) proposed this technique in order to solve large scale linear programs (LPs). The main idea is to exploit the structure of problems where some sets of constraints are independent from each other and are just linked together by joint constraints (Wolsey (1998, p. 185)). The original problem is divided into a so called MP and a set of SPs. The MP works as a coordinator at a superior stage. It treats the linking structure and thus, keeps the problem together. On a subordinated stage the special structure of the SPs is exploited to solve them efficiently (Lübbecke and Desrosiers (2005)).

In section 3.1, the MP is described. In scetion, 3.2, an algorithm is developed to solve the SP.

3.1 Master-problem

The routing model from section 2.2 can be reformulated as a set partitioning formulation. Additional notation and the model are described below.

Sets		
$\mathcal{N}$	=	set of all possible tours.
$\mathcal{N}_p$	=	set of all possible tours for therapist $p \in \mathcal{P}$.
Parameters		
Θ_n^{match}	=	number of matches within a tour n.
$\Theta_{n,i}^{\text{job}}$	=	equal to one, if in tour n job i is treated, otherwise zero.
$\Theta_{n,r,t}^{\text{room}}$	=	equal to one, if in tour n room $r \in \mathcal{R}^{\text{tc}}$ is occupied at time t, otherwise zero.
Decision variables		
z_n	=	equal to one if tour $n \in \mathcal{N}$ is used, otherwise zero.

$$\text{Maximize} \sum_{n \in \mathcal{N}} \Theta_n^{\text{match}} \cdot z_n \tag{3.1}$$

subject to

$$\sum_{n \in \mathcal{N}} \Theta_{n,i}^{\text{job}} \cdot z_n = 1 \qquad \forall\, i \in \mathcal{I}, \tag{3.2}$$

$$\sum_{n \in \mathcal{N}} \Theta_{n,r,t}^{\text{room}} \cdot z_n \leq K_r^{\text{tc}} \qquad \forall\, r \in \mathcal{R}^{\text{tc}}, t \in \mathcal{T}, \tag{3.3}$$

$$\sum_{n \in \mathcal{N}_p} z_n \leq 1 \qquad \forall\, p \in \mathcal{P}, \tag{3.4}$$

$$z_n \in \{0, 1\} \qquad \forall\, n \in \mathcal{N} \tag{3.5}$$

The objective function (3.1) maximizes the number of matches over all tours. The job assignment constraint (3.2) ensures that each real-existing job is done exactly once. The TC capacity constraint (3.3) assures that the TC's capacity cannot be exceeded. The convexity constraint (3.5) imposes that for each therapist at most one tour can be used in the optimal solution.

The number of possible tours $\mathcal{N}$ is huge. It might be computationally intractable to generate all tours. Hence, only a small subset of tours $\mathcal{N}' \subseteq \mathcal{N}$ is used (Lübbecke and Desrosiers (2005)). The resulting problem including only $\mathcal{N}'$ is called restricted master problem (RMP).

In order to generate new tours we employ CG and decompose the problem on therapists. Each tour can be seen as a column for the MP/RMP. A new tour is only added to the RMP when it promises an improvement of the objective function value (Desrosiers and Lübbecke (2005)). To decide whether a new tour improves the objective function value or not the linear relaxation of the RMP (L-RMP) is solved. The resulting dual-variables are used to calculate the reduced cost (RC) for the new tour. If the RC are greater than zero the tour is added to the L-RMP.

Let π_i^{job} be the dual variables for the job assignment constraint (3.2), $\pi_{r,t}^{\text{room}}$ the dual variables for the TC capacity constraint (3.3) and π_p^{conv} the dual variables for the convexity constraint (3.5). Further let δ_i^{job} be a variable indicating whether a job $i \in \mathcal{I}$ is done in a tour and let $\delta_{r,t}^{\text{room}}$ be a variable indicating whether a TC room is occupied in a certain time period. Then the RC for a tour $n \in \mathcal{N}$ results to:

$$RC_n = \Theta_n^{\text{match}} - \left(\sum_{i \in \mathcal{I}} \pi_i^{\text{job}} \cdot \delta_i^{\text{job}} + \sum_{r \in \mathcal{R}^{\text{tc}}} \sum_{t \in \mathcal{T}} \pi_{r,t}^{\text{room}} \cdot \delta_{r,t}^{\text{room}} + \pi_p^{\text{conv}} \right) \tag{3.6}$$

The tour with biggest RC for a given set of dual variables is generated in the SP which is described in section 3.2.

After obtaining a new favorable tour the L-RMP is solved again with the updated set of tours. The full course of the CG algorithm is depicted in Figure 3.1.

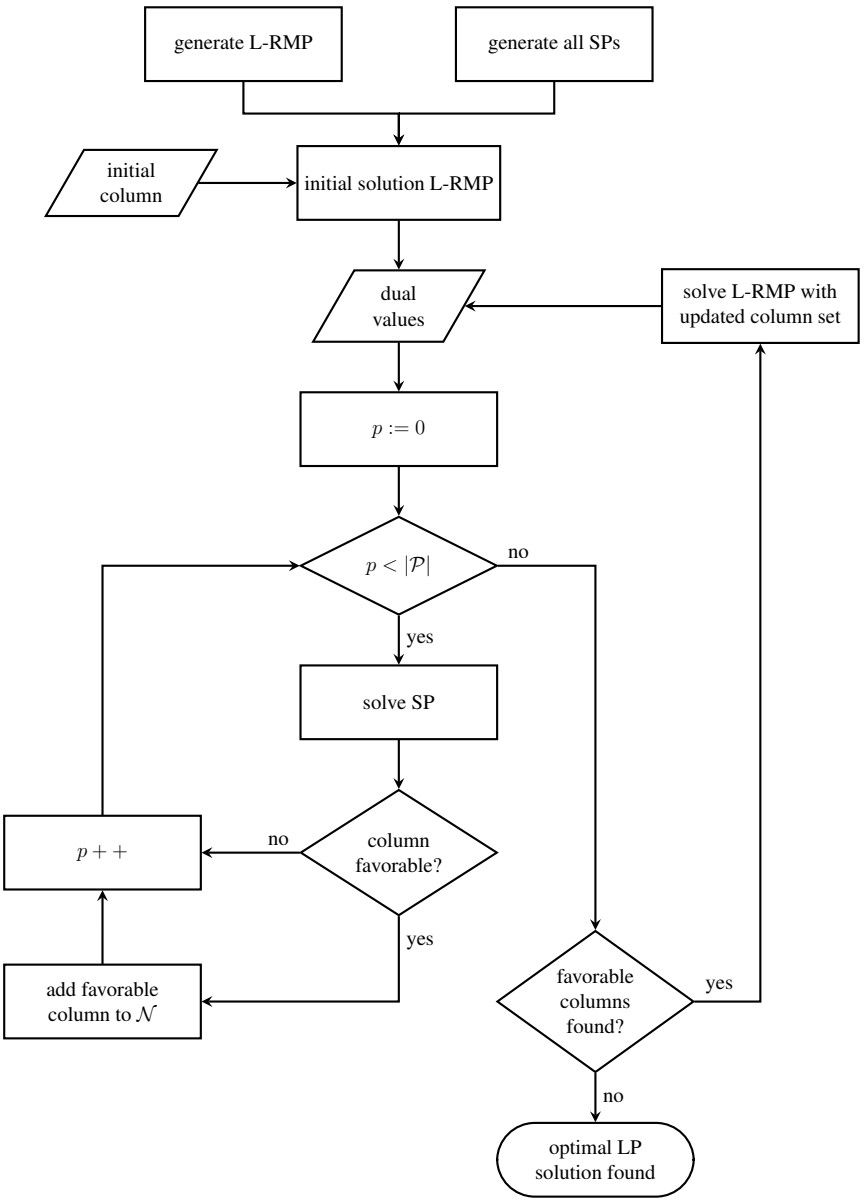

Figure 3.1: Column generation scheme for the ThSP.

First, the L-RMP is generated as well as the SPs for all therapists $p \in \mathcal{P}$. An initial column is used to solve the L-RMP and derive an initial solution. The initial column is a dummy tour that contains no matches. In addition, it is performed by a dummy therapist doing all jobs $i \in \mathcal{I}$ and without occupying any TC during the planning horizon. With this dummy tour the L-RMP is feasible since the tour does not violate any constraint. The objective function value of the initial solution takes the smallest value possible (0) since the dummy tour does not contain any match. By doing so, every other feasible combination of tours is at least as good as the initial solution. Thus, if the L-RMP is feasible there is no possibility that this dummy tour is used in an optimal solution.

In the second step, the initial solution is used to obtain the dual variables for the first iteration. For the first therapist $(p := 0)$ the SP is solved. If the new tour has positive RC and is thus favorable it is added to the set of column $\mathcal{N}$. After the new column

was added or if there was no favorable column found the procedure is repeated for the next therapist. The SP is solved for the second therapist, it is checked if the new column is favorable and if so it is added to the $\mathcal{N}$. This continues until it was iterated over all therapists ($p < |\mathcal{P}|$).

When all SPs were solved it is checked whether at least one favorable column was found or in other words if the set of columns $\mathcal{N}$ was updated. If new columns were added to the L-RMP it is solved again. New dual variables result from the new L-RMP solution and the SPs are solved again.

If within one iteration through all SPs no new column was found the optimal LP-solution is found. Nevertheless, if one would solve the L-RMP again the set of columns $\mathcal{N}$ would remain the same. Thus, also the dual variables would stay the same and with the same duals again no new columns could be obtained.

To avoid unnecessary calculation one could reduce the number of SPs that have to be solved within each iteration. A decision can be added if a certain SP can still generate new promising tours. Once a SP returns RC ≤ 0 no new tours can be generated. Thus, it would be a waste of computational effort to solve the according SP again.

Note that since the linear relaxation of the RMP is solved the CG process does not necessarily provide an integral solution. It just provides an upper bound on the number of matches. However, the generated solution can be taken as a starting point to for a branch-and-price algorithm where an integral solution can be obtained.

3.2 Subproblem

The MP can be solved easily even for a big number of columns. Likewise the process of checking if a column is favorable, adding a column to $\mathcal{N}$ or deriving the dual variables is computed in short time. The crucial part in the column generation process is to solve the SPs and generate feasible tours for the therapists.

The objective function for the SP is to maximize the tour's RC.

$$\text{Maximize } \Theta^{\text{match}} - \left(\sum_{i \in \mathcal{I}} \pi_i^{\text{job}} \cdot \delta_i^{\text{job}} + \sum_{r \in \mathcal{R}^{\text{tc}}} \sum_{t \in \mathcal{T}} \pi_{r,t}^{\text{room}} \cdot \delta_{r,t}^{\text{room}} + \pi_p^{\text{conv}} \right) \tag{3.7}$$

The RC are found when Θ_n^{match}, δ_i^{job} and $\delta_{r,t}^{\text{room}}$ are determined. Thus, an efficient algorithm must be identified to determine those variables quickly.

The easiest possibility would be to model the SP as a VRP based on the routing problem in section 2.2. Then the problem would become a VRP with a single vehicle representing the therapist. However, the SP's complexity would remains almost the same as for the original problem. On the one hand, the problem would

need only constraints for one therapist. On the other hand, the problem would have to be solved $|\mathcal{P}|$ times.

Since the routing perspective is not practical, a new approach must be developed. The approach that is used most when CG is applied to VRPs is interpreting the SPs as a SPPs (Irnich and Desaulniers (2005)). In 1992, Desrochers et al. presented this method and since then a lot of research has been done. However, the problem setting of assigning therapist to jobs and rooms is novel. Thus, no algorithm could be identified which can be applied directly to the ThSP. Therefore, the remainder of this section deals with developing a shortest path algorithm that is capable to solve the ThSP's SP.

In section 3.2.1, the underlying graph for the SPP is described. In section 3.2.2, different approaches to solve SPPs are examined. In section 3.2.3, the final algorithm to solve the CG-SP for the ThSP is presented.

3.2.1 Graph Representation of a Tour

As described above a tour can be seen as a way through a graph. Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be a directed graph where $\mathcal{V}$ is the vertex set and $\mathcal{A}$ is the arc set. The terms vertex and node are used synonymously throughout this work.

The first who dealt with a large-scale single-vehicle VRP with time-windows were Desrosiers et al. (1986). Their nodes represent jobs and each job has a time-window when the node can be reached. The arcs represent the costs that are connected with going from one node to another. This approach is highly efficient since it keeps the number of nodes and thus the number of arcs small.

Unfortunately, this is not suitable for the ThSP. It is not necessary to know only that a job was started in its time-window. It is needed to know when exactly it was started. Only then it can be calculated when a TC room is occupied. Therefore, in the ThSP each vertex in $\mathcal{G}$ represents a job-room-time combination $\langle i, r, t \rangle$. For example, job 1 can be performed in rooms 1 or 2 and start in periods 15 or 16. Then the resulting set of vertexes would be $\mathcal{V} = \{\langle 1,1,15\rangle, \langle 1,1,16\rangle, \langle 1,2,15\rangle, \langle 1,2,16\rangle\}$. Thus, the resulting graph contains much more nodes and arcs than in the approach of Desrosiers et al. (1986) and consequently it is harder to find a shortest path through the network. To overcome this difficulty the graph is reduced to only those nodes and arcs that are really needed to generate feasible solution for the SPs.

First of all, for all real-existing jobs $i \in \mathcal{I}$ all job-room-time combinations are generated. For all possible shift-ends one dummy node is created. Let $\mathcal{I}^{S^{\text{endShort}}}$, $\mathcal{I}^{S^{\text{endNorm}}}$, $\mathcal{I}^{S^{\text{endLong}}}$ and $\mathcal{I}^{S^{\text{end}}}$ be the sets for jobs representing short, normal, long and all shift ends, respectively. Further, two additional dummy nodes are created: v^{OUT} for the tour start representing the source of $\mathcal{G}$ and v^{IN} for the latest possible end of the tour representing the sink of $\mathcal{G}$. Job-room-time combination for break jobs are

only generated if the according break is possible in the SP. For example, if only a short shift is possible in the SP no breaks have to be generated and if only short or normal shifts are possible no second break has to be generated.

In order to further reduce the set of nodes only those nodes are maintained that can improve the RC. Therefore, nodes are weighted with their positive contribution to the RC. Positive contribution means that it is only considered if a job is a favored job and if the duals for the job assignment constraint are negative. For example, job 1 is a favored job for the SP's therapist and the dual value for this job is -0.5. Then the node's contribution to the RC is $1 - (-0.5) = 1.5$. Nodes are now eliminated from the graph if their weight is not bigger than 0. In that case they can never be higher than the RC. Especially at the beginning of the CG algorithm this is efficient. Directly after the start the L-RMP has only generated a few duals and the number of favored jobs is generally small. Thus, the graph is small and a solution is obtained fast.

Later on, when the L-RMP has generated many duals this does not reduce the set of vertexes that much. However, also the set of arcs can be reduced. Let v and v' be two nodes in $\mathcal{V}$. Let i_v be the job, r_v the room and t_v the starting time that is represented by a node $v \in \mathcal{V}$. An arc $(v, v') \in \mathcal{A}$ connects v and v' if $i_{v'}$ is a possible successor job of i_v and if $r_{v'}$ can be reached from r_v in a time not greater than $t_v + d^{\text{job}}_{i_v} + c_{r_v, r_{v'}}$.

However, some sub-paths are dominated by others and thus can be removed from $\mathcal{G}$. To do so, the remaining nodes in $\mathcal{G}$ are weighted with their overall contribution to the RC. That does include positive duals from the job assignment constraint and the capacity constraint for TC rooms. The only exception is the sink which gets the weight the convexity dual for the according therapist. This is a constant and cannot be changed by any node.

In order to eliminate arcs, it has to be distinguished between nodes representing TC and non-TC rooms. The more straightforward case is non-TC rooms. For every node $v \in \mathcal{V}$ all outgoing arcs are examined. Let $\mathcal{U}$ be the set of direct successors of v which have the same job-room combination. If $|\mathcal{U}| > 1$ only u with the earliest starting time keeps its arc from v. The rest of the nodes $u' \in \mathcal{U} \backslash \{u\}$ has the same contribution to the RC as u but it is not possible that u' has more possible successors than u since u's starting time is earlier. In a final step, all outgoing arcs are deleted for nodes that have no incoming arcs and therefore cannot be part of the shortest path. The process of eliminating arcs from non-TC rooms is depicted in Figure 3.2 for a fictitious graph.

In graph (a) a break can follow job 1 in the periods 3,4,5 and 6. It is sufficient to allow a break only in period 3 when following job 1. Then the arcs to the breaks in period 4,5 and 6 can be disposed which leads to graph (b). Now, there are arcs leaving breaks in $t = 4, 5, 6$ but it is not possible to get to those nodes when starting

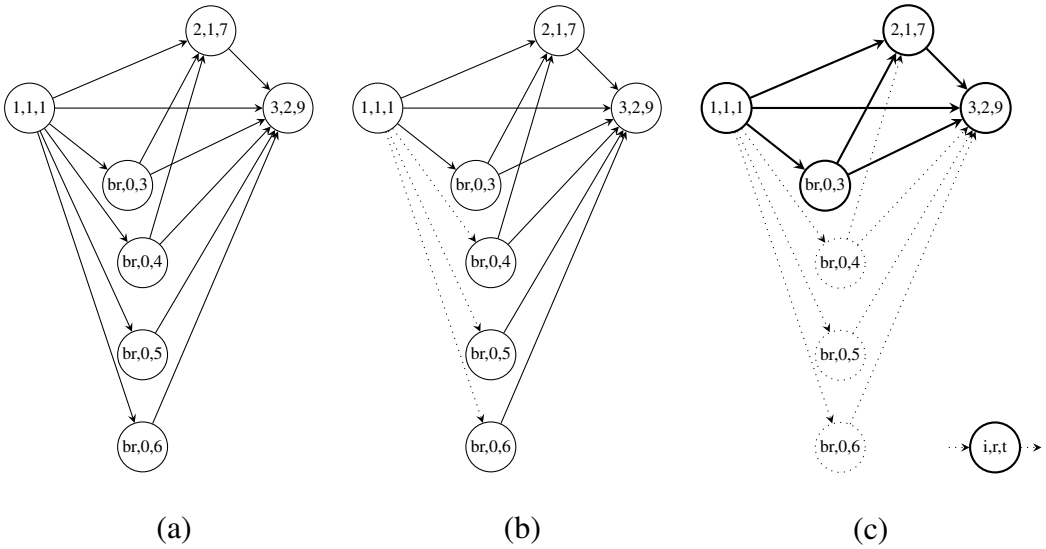

Figure 3.2: Reduction of edges (non-TC rooms).

in the source.[3] Thus, all of their outgoing arcs can be disposed as well. The graph is then reduced to (c).

The case for TC rooms works similar. Again for every node $v \in \mathcal{V}$ all outgoing arcs are examined and again $\mathcal{U}$ is the set of direct successors of v with the same job-room combination. However, this time it is not guaranteed that for all nodes in $\mathcal{U}$ the node with the earliest starting times dominates the others. For the TC case one has to consider the sum of the duals for the TC capacity constraint from the start of the job to its end. Hence, it is necessary not only to maintain the arc from v to u with the earliest starting time of all nodes in $\mathcal{U}$. It is rather necessary to maintain additionally all nodes $u \in \mathcal{U}$ with bigger RC than all nodes $u' \in \mathcal{U} \mid t_{u'} < t_u$ with smaller starting time. The process of choosing nodes u that keep their arc from v is depicted in Figure 3.3.

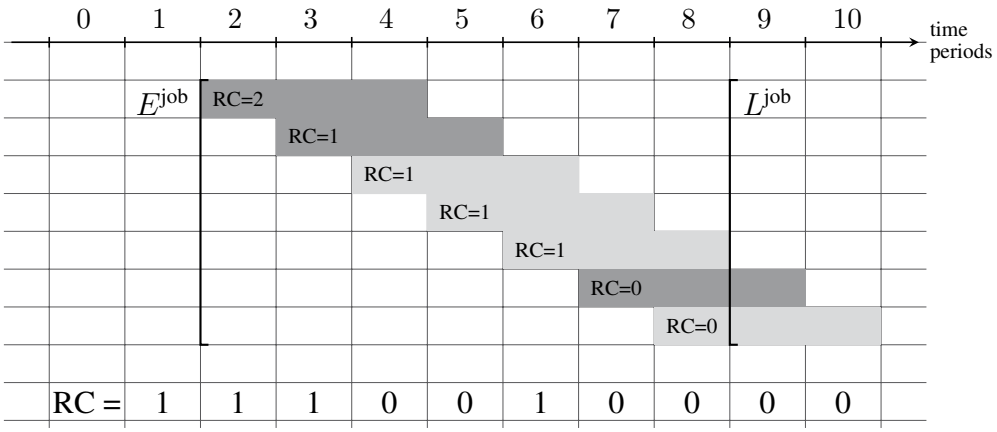

Figure 3.3: Reduction of edges (TC rooms).

A fictitious job can be started earliest in period 2 and latest in period 8. The RC for

[3] It is assumed that $(\text{br}, 0, 4)$, $(\text{br}, 0, 5)$ and $(\text{br}, 0, 6)$ do not have incoming arcs from any other node of the entire graph (not only from the small part shown in Figure 3.2).

the time periods are stated at the bottom. The arc connecting the node representing a start in period 2 has to be maintained because it is not guaranteed that a later start can reach all of its successors. The second possibility is to start the job in period 3 with RC from the TC room capacity constraint of 1. Again the incoming arc to this node has to be maintained because it is the earliest job with RC smaller than 2. The next three possibilities (start in period 4,5 and 6) do not improve the RC. Thus, their incoming arcs from v can be disposed. In period 7 the RC can be reduced again. Consequently, also this incoming arc has to be maintained. It is not possible to get a value lower than 0 therefore all arcs going to a node with later start can be disposed. Thus, only the nodes representing startings in period 2, 3 and 7 have to be maintained.

3.2.2 Shortest Path Problems

In the previous section, the structure of the graph for the SPP was examined. This section deals with the question what kind of algorithms could be used to solve SPPs and which one suits best the ThSP.
First of all, it must be defined what exactly is meant by a shortest path. Let $\Pi = \langle v_0, v_1, \ldots, v_k \rangle$ be a path containing nodes v_0 to v_k. Let $w(\Pi)$ be the weight of the path which is the sum over all arc weights:

$$w(\Pi) = \sum_{i=1}^{k} w(v_{i-1}, v_i). \tag{3.8}$$

If there exists a path from node u to v the shortest path between them is defined as the path with minimum path weight $w(\Pi)^*$. If no such path exists the weight is set to infinity (Cormen et al. (2009, p. 643)). In the ThSP-SP always the path between v^{OUT} and v^{IN} has to be determined. At least one path exists with $w(\Pi)$ of the negative convexity constraint dual (the proportional RC of v^{OUT}). Thus, the case that no path exists can be neglected for this work.

The ThSP-SP is de facto calculating the longest and not the shortest path. However, the described means can be applied as well since longest path problems can be transformed to a SPPs by inverting the sign of the arc weights (Ahuja et al. (1993, p. 102)). The shortest path weight that is found then is the longest path weight with negative sign.

After defining the shortest path the question is how it can be determined. In general, it is distinguished between label setting and label correcting algorithms. Both have in common that they are iterative approaches that assign distance or cost labels to nodes at each iteration(Ahuja et al. (1993, p. 96)). The difference between both is how they update those distance labels. Label setting algorithms generally based on Dijkstra (1959) mark one label as permanent at each iteration (Ahuja et al. (1993, p. 96), Bertsekas (1998, p. 65)). Label correcting algorithms generally based on

Bellman (1958) set all labels permanent in the last step. Until then all labels are considered as temporary (Ahuja et al. (1993, p. 96), Bertsekas (1998, p. 73)).

Label setting algorithms are considered to be more efficient than label correcting algorithms. The restriction to use a label setting algorithm is that the underlying graph does not contain cycles or that the graph has non negative arc weights (Ahuja et al. (1993, pp. 96-97)). Both is fulfilled for the ThSP-SP. However, some kind of cycling can exist within the set of nodes representing the same jobs. Hence, an algorithm is needed which is algorithmically more flexible and label correcting algorithms are considered to be more flexible (Ahuja et al. (1993, pp. 97)). Therefore, a label correcting algorithm is used to solve the ThSP-SP.

Without further restrictions the SP would generate shortest paths which are not feasible for the ThSP. For example, if two jobs have sufficiently large overlapping time windows and at least one of the jobs improves the RC it would be possible that a job is done more than one time in a tour. This situation is shown in Figure 3.4.

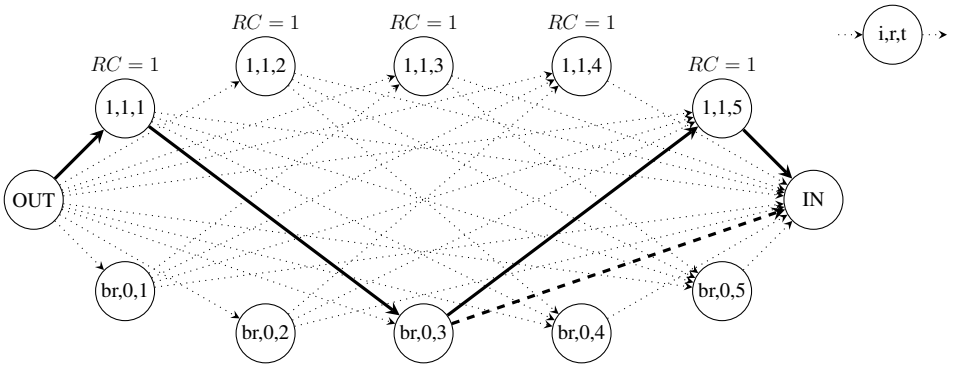

Figure 3.4: Problem with large overlapping time windows.

Job 1 and the break job have overlapping time windows. Thus, job 1 can be done two times (thick solid line). However, after visiting node $\langle 1, 1, 1\rangle$ and $\langle \text{br}, 0, 4\rangle$ the only possible successor is node $\langle \text{IN}\rangle$ (thick dashed line). Thus, it must be assured that no job is visited twice.

If the underlying graph was a job graph this could be seen as elementary SPP. An elementary SPP determines the shortest elementary path between two nodes. An elementary path is defined as a path which contains each node at most once (Irnich and Desaulniers (2005)). For the ThSP this becomes more complicated. Practically each node can at most be once in a path since the graph does not contain cycles. [4] However, the path may contain more than one node which is representing the same job. Fortunately, the same means that can be used for the elementary SPP can be used for the ThSP-SP as well.

The idea is to give the label information about if jobs have been done or if jobs still can be done. This information works as a binary resource. If a job was done and thus

[4] This is true for all SPs because it is only possible to go from one node to another node if the successor represents a later start. Thus, one can never return to a node that was already visited.

the resource was used another node which requires this resource cannot be visited. A recent overview over exact solution approaches to the resource constrained SPP is provided in Pugliese and Guerriero (2013).

The approach of assigning information about jobs to labels appeared first in 1989 when Beasley and Christofides proposed to use information on what jobs have been done. Feillet et al. (2004) state that it would be more efficient to save information on jobs that still can be done. Indeed, the information on jobs that still can be done reduces the set of possible successor nodes for a given label more than information on jobs that have been done. Thus, for the ThSP-SP the information on jobs that still can be done is used to extend labels. How exactly labels are extended is described in section 3.2.3.

However, it mus be discussed weather labels are extended to its predecessors (backward iteration) or its successors (forward iteration). That means if one starts the iteration at the source or the sink. It is considered to be most efficient if both is combined (Ceselli et al. (2009)). That means if one starts at the source and sink simultaneously. From the source labels are extended to its successors and from the sink to its predecessors. This is done until both have examined a half of the graph. Then the labels from forward and backward iteration are combined to determine the cost-optimal tour.

All in all the ThSP-SP turns out to be a kind of elementary SPP with resource constraints which is solved by means of a bidirectional label correcting algorithm. How this algorithm works in particular is described in the following section.

3.2.3 The Bidirectional Label Correcting Algorithm

The previous sections focused on the graph structure and what kind of algorithm could be used. This section focuses on how the algorithm works in detail. The process of solving the SP can be divided into the following 6 stages:

(1) creating the reduced graph,

(2) dividing the graph in half,

(3) performing the forward iteration for one half,

(4) performing the backward iteration for the other half,

(5) combining the results of forward and backward iteration and

(6) post-processing to provide information for L-RMP.

Stage 1 – creating the reduced graph

The graph is constructed as described in section 3.2.1. Only those nodes and arcs are part of the graph that could lead to positive RC.

Stage 2 – dividing the graph in half

To be able to perform a forward and backward iteration, the graph has to be divided in half. For the sake of a better understanding let the fictitious graph in Figure 3.5 be a graph that has to be divided. The labels above the nodes indicate the starting time represented by the node.

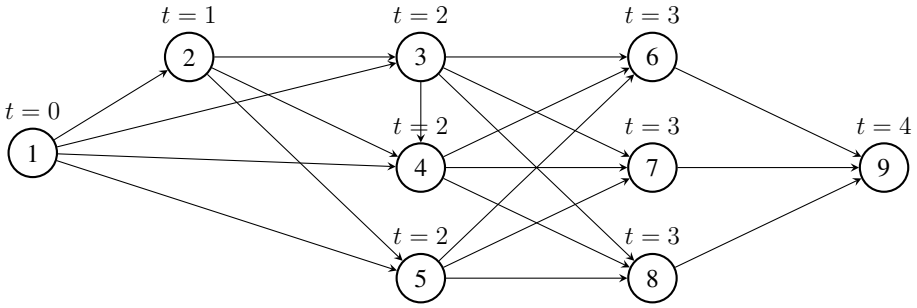

Figure 3.5: Fictitious graph that has to be split.

However the graph is divided, the following two conditions must hold. First, for every node in the graph of the forward iteration ($\mathcal{G}^{\text{forward}}$) all the node's predecessors must also be part of $\mathcal{G}^{\text{forward}}$. Second, for every node in the graph of the backward iteration $\mathcal{G}^{\text{backward}}$ all the node's successors must also be part of $\mathcal{G}^{\text{backward}}$. Otherwise, it might be that the splitting forbids beneficial tours.

A straightforward possibility to ensure that the precedence relationship between the nodes is maintained, is to divide the nodes according to the starting time they represent. That means that one defines a certain separation time t^{sep} and all nodes which have a starting time less or equal to t^{sep} are added to $\mathcal{G}^{\text{forward}}$ and remaining nodes to $\mathcal{G}^{\text{backward}}$. If this rule is applied to the graph in Figure 3.5 and the separation time is set to $t^{\text{sep}} = 2$ the resulting graphs are depicted in Figure 3.6. All nodes on

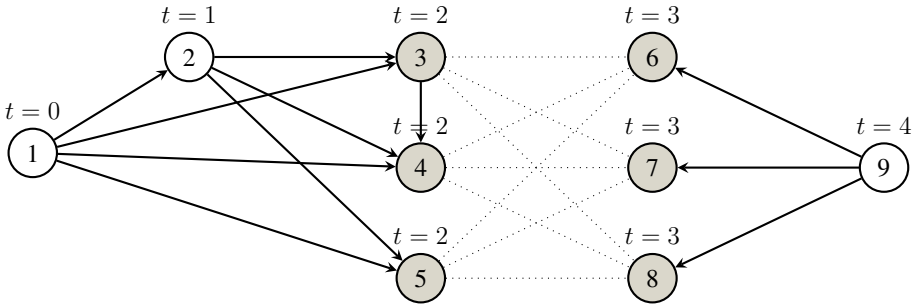

Figure 3.6: Divided graph when dividing the set of nodes.

the left hand side would be assigned to the graph for forward iteration and all nodes on the right hand side to the backward iteration. The gray shaded nodes and the dotted lines between them represent the connection between both iterations and will be used when the results from forward and backward iteration are combined.

Depending on how t^{sep} is set, the number of arcs and nodes in each half of the graph can vary considerably. For the label correcting algorithm the separation time is set to:

$$t^{\text{sep}} = \left\lceil \frac{\mathcal{S}_p^{St} + \max(t \mid t \in \mathcal{S}_p^{end})}{2} \right\rceil \tag{3.9}$$

Thus, all nodes which represent a starting time less or equal to the half of the maximum shift length are added to $\mathcal{G}^{\text{forward}}$. A discussion on the balancing of the resulting graphs can be found in section 4.2.

Stage 3 – forward iteration

When the sub-graphs are defined, the forward iteration begins for the sub-graph containing v^{OUT}. The basic idea behind the label correcting algorithm can be summarized as follows. Paths are labeled with information on their costs and resource consumption. One starts with the trivial path $\Pi = \langle v^{\text{OUT}} \rangle$ and explores iteratively new paths. This is done by manipulating two sets. The first set $\Lambda^{\text{unprocessed}}$ contains all unprocessed paths that wait to be extended. The second set Λ^{useful} contains all useful paths that have already been treated (Irnich and Desaulniers (2005)). Dominance rules are used to dispose labels from the set and accelerate the algorithm. The course of the algorithm can be seen in Algorithm 1. Thereafter, the structure of a label, the dominance rules and the label extension is detailed.

First of all, a root label λ_0 is created. The root label represents the trivial path $\Pi = \langle v^{\text{OUT}} \rangle$. The set of unprocessed paths $\Lambda^{\text{unprocessed}}$ is created which contains only λ_0 at the beginning. The set for useful paths Λ^{usefull} is empty at the beginning. Further, two binary variables are introduced indicating if a label dominates other labels (b^{dominant}) and if the label is dominated by another label (b^{inferior}).

The procedure continues as long as $\Lambda^{\text{unprocessed}}$ is not empty. The first label λ of $\Lambda^{\text{unprocessed}}$ is taken and deleted from the set. It is compared to each label λ' in Λ^{usefull}. If λ dominates λ' and λ' is the first label dominated by λ then λ' is replaced by λ. Additionally b^{dominant} is set true if λ' is the first label that is dominated by λ. If λ' is not the first label that is dominated λ' is removed from Λ^{usefull}. If λ is dominated by λ' then b^{inferior} is set true.

In case that λ has not replaced a label and is not dominated by a label already in Λ^{usefull} then λ is added to Λ^{usefull}.
Whenever λ was not dominated the path is extended to all possible successor nodes of λ's final node. This is only possible if the extension is not forbidden by some resource restriction. All labels λ' that are generated during the extension process are added to $\Lambda^{\text{unprocessed}}$.

The remainder of this forward iteration part describes in detail (a) how a label is structured, (b) what values are assigned to the root label, (c) when a label dominates and when it is dominated, (d) how a label is extended and (e) how the process

Algorithm 1: LABEL CORRECTING ALGORITHM (FORWARD ITERATION)
Input: Sub-graph $\mathcal{G}^{\text{forward}}$ and all information from nodes and arcs included in it
Output: A set of useful paths from the forward iteration

```
1  create root label λ_0
2  Λ^unprocessed := {λ_0}
3  Λ^usefull := ∅
4  b^dominant := false
5  b^inferior := false
6  while Λ^unprocessed ≠ ∅ do
7      λ ← Λ^unprocessed[0]
8      Λ^unprocessed := Λ^unprocessed \ {λ}
9      foreach λ' ∈ Λ^usefull do
10         counter := 0
11         if λ > λ' then
12             if counter := 0 then
13                 b^dominant := true
14                 Λ^usefull := {Λ^usefull ∪ {λ}} \ {λ'}
15                 counter := 1
16             else
17                 Λ^usefull := Λ^usefull \ {λ'}
18         else
19             if λ < λ' then
20                 b^inferior := true
21     if b^dominant = false and b^inferior = false then
22         Λ^usefull := Λ^usefull ∪ {λ}
23     if b^inferior = false then
24         foreach u ∈ V^Succ_{v_λ} do
25             if path can be extended to u then
26                 create new label λ'
27                 Λ^unprocessed := Λ^unprocessed ∪ {λ'}
```

of getting labels from the unprocessed paths and adding label to the useful paths can be adjusted to strengthen the algorithm.

(a) label structure

Each label λ contains the following 4 basic parameters:

- The path so far that is represented by the label: Π_λ.
- The path's last node: v_λ.
- The accumulated RC for the path so far: RC_λ.
- The set of jobs that still can be done: $\mathcal{I}_\lambda$.

In addition to those parameters, 4 binary variable are assigned to each label. The latter will be needed when the resulting sets of useful paths from forward and backward iteration are combined.

- A variable indicating if the path so far represents a short shift: $b_\lambda^{\text{ShortShift}}$.
- A variable indicating if the path so far represents a long shift: $b_\lambda^{\text{LongShift}}$.
- A variable indicating if the first break is already in the path: $b_\lambda^{\text{AlreadyBr1}}$.
- A variable indicating if the second break is already in the path: $b_\lambda^{\text{AlreadyBr2}}$.

(b) creation of root-label

The root label λ_0 gets the following parameters assigned. The path consists only of the graph's source: $\Pi_{\lambda_0} := \langle v^{\text{OUT}} \rangle$. Thus, v^{OUT} is the last node. The RC are the sum of the convexity duals $\text{RC}_{\lambda_0} := \pi_p^{\text{conv}}$.

The set of jobs that still can be done contains every real-existing job that can be reached. A job cannot be reached if the time to perform it and return to the break room exceeds the time for a short shift. If shift ends representing a short shift exist jobs corresponding to those ends are also added to $\mathcal{I}_{\lambda_0}$.

The binary variable $b_{\lambda_0}^{\text{ShortShift}}$ is set to 1 and $b_{\lambda_0}^{\text{LongShift}}$, $b_{\lambda_0}^{\text{AlreadyBr1}}$ and $b_{\lambda_0}^{\text{AlreadyBr2}}$ to 0.

(c) dominance rules

A label λ dominates a label λ' if the following conditions hold:

- The dominating label must have greater RC ($\text{RC}_\lambda > \text{RC}_{\lambda'}$).
- Both labels must have the same last node ($v_\lambda = v_{\lambda'}$).
- Both labels must be of the same shift type ($b_\lambda^{\text{ShortShift}} = b_{\lambda'}^{\text{ShortShift}} \wedge b_\lambda^{\text{ShortLong}} = b_{\lambda'}^{\text{ShortLong}}$).
- The dominating label must have at least the same jobs that still can be visited ($\mathcal{I}_{\lambda'} \subseteq \mathcal{I}_\lambda$).

If the RC are not greater the tour cannot lead to a better path through the graph. The remaining three conditions ensure that the dominating label RC_λ is not denying paths that could have been explored from $\text{RC}_{\lambda'}$.

The reversed case that a label λ' is dominated by a label λ works similar. The only difference is that it is sufficient that the RC are greater or equal such that a label is dominated ($\text{RC}_\lambda \geq \text{RC}_{\lambda'}$). The other conditions remain the same.

(d) label extension

In the forward iteration a label λ is always extended from its last node v_λ to the successors of v_λ. This holds unless resource constraints deny an extension. There are two possibilities why a successor node cannot be reached. First, the successor node represents a job that was already done. Second, the starting time represented by the node is too late and would violate working time regulations. For example, no break was assigned and it would be impossible to perform the job represented by the successor node and return to the break room before the short shift ends.

If no such restriction prohibits an extension the label is extended as follows. Let λ^{parent} be the label that is to be extended and λ^{new} the resulting label. First, the new label gets exactly the same values as the parent label:

$$\lambda^{\text{new}} := \lambda^{\text{parent}} \tag{3.10}$$

In the next step, the parameters are updated. The target node u in which direction λ^{parent} is to be extended is added to the path so far:

$$\Pi_{\lambda^{\text{new}}} := \Pi_{\lambda^{\text{new}}} \cup \{u\} \tag{3.11}$$

The proportional RC of u are added:

$$\text{RC}_{\lambda^{\text{new}}} := \text{RC}_{\lambda^{\text{new}}} + \text{RC}_u \tag{3.12}$$

If the job i_u that is represented by the destination node u is break job $br1$ or $br2$ the binary variables are set accordingly:

$$i_u = \text{br1} \rightarrow b^{\text{ShortShift}}_{\lambda^{\text{new}}} = 0, b^{\text{AlreadyBr1}}_{\lambda^{\text{new}}} = 1 \tag{3.13}$$

$$i_u = \text{br2} \rightarrow b^{\text{LongShift}}_{\lambda^{\text{new}}} = 1, b^{\text{AlreadyBr2}}_{\lambda^{\text{new}}} = 1 \tag{3.14}$$

The part that requires most calculation during the extension process is to update the set of jobs that still can be done. In a first step, job i_u is removed from the set:

$$\mathcal{I}_{\lambda^{\text{new}}} := \mathcal{I}_{\lambda^{\text{new}}} \setminus \{i_u\} \tag{3.15}$$

$$\tag{3.16}$$

Thereafter, the possible shift ends are updated. If job i_u was break $br1$ the tour can no longer represent a short shift. It must then be a normal shift until the break $br2$ is given. Thus, all jobs representing a short shift must be removed from the set and

all jobs representing a normal shift must be added to it. Additionally break $br2$ has to be added.

$$i_u = br1 \rightarrow \mathcal{I}_{\lambda^{\text{new}}} := \left\{ \mathcal{I}_{\lambda^{\text{new}}} \cup \mathcal{I}^{S^{\text{endNorm}}} \cup \{br2\} \right\} \setminus \mathcal{I}^{S^{\text{endShort}}} \tag{3.17}$$

If job i_u was break $br2$ the tour can no longer represent a normal shift. It must then be a long shift. Thus, all jobs representing a normal shift must be removed from the set and all jobs representing a long shift must be added to it.

$$i_u = br2 \rightarrow \mathcal{I}_{\lambda^{\text{new}}} := \left\{ \mathcal{I}_{\lambda^{\text{new}}} \cup \mathcal{I}^{S^{\text{endLong}}} \right\} \setminus \mathcal{I}^{S^{\text{endNorm}}} \tag{3.18}$$

By (3.17) and (3.18) all dummy and break jobs are updated. However, it is possible that some of the jobs which are in $\mathcal{I}_{\lambda^{\text{new}}}$ cannot be reached anymore since the time to get from $r_{v_{\lambda^{\text{new}}}}$ (the room that represents the last node of λ^{new}) to the room where the successor job should be performed is too short. Or it might be that a later shift end allows additional jobs to enter $\mathcal{I}_{\lambda^{\text{new}}}$. How $\mathcal{I}_{\lambda^{\text{new}}}$ is updated for real-existing jobs is shown in Algorithm 2.

Algorithm 2: UPDATE THE SET OF JOBS THAT STILL CAN BE DONE
Input: The set of jobs that still can be done $\mathcal{I}_{\lambda^{\text{new}}}$, all jobs that a therapist p can perform $\mathcal{I}_p$, the path so far $\Pi_{\lambda^{\text{new}}}$ and the latest possible start time for a given job-room-combination $t^{\text{latest}}_{j,l}$
Output: Updated set $\mathcal{I}_{\lambda^{\text{new}}}$

```
foreach j ∈ I_p ∧ v_j ∉ Π_{λ^new} do
    foreach l ∈ R_j do
        if j ∈ I_{λ^new} and t_{v_{λ^new}} + d^job_{i_{v_{λ^new}}} + c_{r_{v_{λ^new}},l} > t^latest_{j,l} then
            I_{λ^new} := I_{λ^new} \ {j}
        else
            if j ∉ I_{λ^new} and t_{v_{λ^new}} + d^job_{i_{v_{λ^new}}} + c_{r_{v_{λ^new}},l} ≤ t^latest_{j,l} then
                I_{λ^new} := I_{λ^new} ∪ {j}
```

The algorithm's first loop runs for each job j that can be performed by therapist p and which is not represented by a node already in the path $\Pi_{\lambda^{\text{new}}}$. The second loop runs over all rooms l where job j can be performed. For each job-room-combinations (j, l) the following conditions are checked. If a job in $\mathcal{I}_{\lambda^{\text{new}}}$ cannot be reached from the last node in the path the job is removed from $\mathcal{I}_{\lambda^{\text{new}}}$. This is the case when the starting time $t_{v_{\lambda^{\text{new}}}}$ of the job $i_{v_{\lambda^{\text{new}}}}$ (represented by the last node in the path) plus the duration for that job plus the time needed to travel from the room where $i_{v_{\lambda^{\text{new}}}}$ is performed to l where j is performed is bigger than the latest starting time of j in l. For jobs that are not in $\mathcal{I}_{\lambda^{\text{new}}}$ it is checked if they can be reached. If so, they are added to $\mathcal{I}_{\lambda^{\text{new}}}$.

(e) label getting and adding strategies

In Algorithm 1 always the first label is taken from unprocessed paths to be examined and it is not defined where a new label is added to unprocessed paths. A straightforward approach is to add the label in the last position. Then it would be a first-in-first out strategy. However, this strategy can be improved in terms of the overall performance of the algorithm (Bertsekas (1998, pp. 73-78)). The idea is that labels should be examined first if they are more likely to dominate othes. This would lead to a smaller overall number of generated labels and thus accelerate the algorithm.

For the case of choosing a label from unprocessed paths Bertsekas (1998, pp. 76) suggests to use a so called large label last strategy. Since a longest path has to be found for the ThSP it changes to a smallest label last (SLL) method. The first label of unprocessed paths in compared to the average label in unprocessed paths. If the RC of the label are bigger than the RC of the average label the label is used in the iteration. Otherwise, the label is taken from the top and added to the end of unprocessed paths. This continues until a large enough label is discovered. The average reduced cost $\overline{RC}$ of unprocessed paths can be calculated as follows (Bertsekas (1998, p. 77)):

$$\overline{RC} = \frac{\sum\limits_{\lambda \in \Lambda^{\text{unprocessed}}} RC_\lambda}{|\Lambda^{\text{unprocessed}}|} \tag{3.19}$$

If a new label λ' enters $\Lambda^{\text{unprocessed}}$ the resulting $\overline{RC}^{\text{new}}$ can be derived from the former $\overline{RC}^{\text{old}}$ as follows:

$$\overline{RC}^{\text{new}} = \frac{\overline{RC}^{\text{old}} + RC_{\lambda'}}{|\Lambda^{\text{unprocessed}}| + 1} \tag{3.20}$$

If a label λ' leaves $\Lambda^{\text{unprocessed}}$ the resulting $\overline{RC}^{\text{new}}$ are:

$$\overline{RC}^{\text{new}} = \frac{\overline{RC}^{\text{old}} - RC_{\lambda'}}{|\Lambda^{\text{unprocessed}}| - 1} \tag{3.21}$$

For the case of adding new labels to unprocessed paths Bertsekas (1998, pp. 76) suggests a small label first strategy. For the ThSP this turns into a largest label first (LLF) strategy. A newly generated label is compared to the top-node of unprocessed paths. If the new label has greater RC it is added to the top. Otherwise, it is added to the end.

LLF and SLL strategy are both used during in the label setting algorithm to solve the ThSP.

Stage 4 – backward iteration

In its essence the backward iteration works similar to the forward iteration as described in Algorithm 1 on page 26. The differences in the course of algorithm are that the backward iteration works on the graph $\mathcal{G}^{\text{backward}}$ and that paths are extended to possible predecessor. Thus, $u \in \mathcal{V}_{v^\lambda}^{\text{Succ}}$ in line 24 changes to $u \in \mathcal{V}_{v^\lambda}^{\text{Pred}}$. The pseudecode depiction of the backward iteration can be found in Appendix A.

Although the overall course of forward and backward iteration are identical they differ in (a) how the root label is created and (b) how labels are extended.

(a) creation of root-label

The backward iteration starts at the sink of graph $\mathcal{G}$ which is the source of sub-graph $\mathcal{G}^{\text{backward}}$. Thus, the root label λ_0 gets the following parameters assigned. The path consists only of $\mathcal{G}^{\text{backward}}$'s source: $\Pi_{\lambda_0} := \langle v^{\text{IN}} \rangle$. Hence, v^{IN} is the last node. The RC_{λ_0} are set to 0.

At the beginning only nodes representing a possible shift end can be reached. Therefore, the set of jobs that can be reached is:

$$\mathcal{I}_{\lambda_0} := \mathcal{I}^{S^{\text{endShort}}} \cup \mathcal{I}^{S^{\text{endNorm}}} \cup \mathcal{I}^{S^{\text{endLong}}}. \tag{3.22}$$

The binary variables $b_{\lambda_0}^{\text{AlreadyBr2}}$, $b_{\lambda_0}^{\text{AlreadyBr2}}$, $b_{\lambda_0}^{\text{ShortShift}}$ and $b_{\lambda_0}^{\text{LongShift}}$ are all set to 0.

(b) label extension

The label extension keeps the same as in the forward iteration for the path so far $\Pi_{\lambda^{\text{new}}}$, the reduced cost $\text{RC}_{\lambda^{\text{new}}}$ and the binary variables $b_{\lambda^{\text{new}}}^{\text{AlreadyBr1}}$ and $b_{\lambda^{\text{new}}}^{\text{AlreadyBr2}}$. However, the updating of the binary variables $b_{\lambda^{\text{new}}}^{\text{ShortShift}}$ and $b_{\lambda^{\text{new}}}^{\text{LongShift}}$ and the set of jobs that still can be done $\mathcal{I}_{\lambda^{\text{new}}}$ is different.

If the job i_u that is represented by the destination node $u^{S^{\text{end}}}$ is a job representing a short or a long shift end the binary variables are set accordingly:

$$i_u \in \mathcal{I}^{S^{\text{EndShort}}} \rightarrow b_{\lambda^{\text{new}}}^{\text{ShortShift}} = 1 \tag{3.23}$$

$$i_u \in \mathcal{I}^{S^{\text{EndLong}}} \rightarrow b_{\lambda^{\text{new}}}^{\text{ShortLong}} = 1 \tag{3.24}$$

The most challenging part again is to update the set of jobs that still can be done. First, the job that has just be done is removed from the set:

$$\mathcal{I}_{\lambda^{\text{new}}} := \mathcal{I}_{\lambda^{\text{new}}} \setminus \{i_u\} \tag{3.25}$$

If a job i_u represented by a destination node $u^{S^{\text{end}}}$ representing a shift end $\mathcal{I}_{\lambda^{\text{new}}}$ is updated as follows:

$$i_u \in \mathcal{I}^{S^{\text{end}}} \rightarrow \mathcal{I}_{\lambda^{\text{new}}} = \mathcal{I}_{\lambda^{\text{new}}} \setminus \mathcal{I}^{S^{\text{end}}} \tag{3.26}$$

Additionally breaks must be included according to the shift end.

$$i_u \in \mathcal{I}^{S^{\text{EndNorm}}} \rightarrow \mathcal{I}_{\lambda^{\text{new}}} = \mathcal{I}_{\lambda^{\text{new}}} \cup \{br1\} \tag{3.27}$$

$$i_u \in \mathcal{I}^{S^{\text{EndLong}}} \rightarrow \mathcal{I}_{\lambda^{\text{new}}} = \mathcal{I}_{\lambda^{\text{new}}} \cup \{br2\} \tag{3.28}$$

In case that break $br1$ was given break $br2$ must be made available

$$i_u = br2 \rightarrow \mathcal{I}_{\lambda^{\text{new}}} = \mathcal{I}_{\lambda^{\text{new}}} \cup \{br1\} \tag{3.29}$$

After those steps are performed, all dummy and break jobs are updated correctly. However, also for the backward iteration it might be possible that some jobs in the set cannot be reached anymore or other jobs could be reached but are not included in the set. The updating Algorithm 3 works similar to Algorithm 2.

Algorithm 3: UPDATE THE SET OF JOBS THAT STILL CAN BE DONE
Input: The set of jobs that still can be done $\mathcal{I}_{\lambda^{\text{new}}}$, all jobs that a therapist p can perform $\mathcal{I}_p$, the path so far $\Pi_{\lambda^{\text{new}}}$ and the earliest possible start time for a given job-room-combination $t^{\text{earliest}}_{j,l}$
Output: Updated set $\mathcal{I}_{\lambda^{\text{new}}}$

```
foreach j ∈ I_p ∧ v_j ∉ Π_λ^new do
    foreach l ∈ R_j do
        if j ∈ I_λ^new and t_{v_λ^new} + d^job_{j_{v_λ^new}} + c_{r_{v_λ^new},l} < t^earliest_{j,l} then
            I_λ^new := I_λ^new \ {j}
        else
            if j ∉ I_λ^new and t_{v_λ^new} + d^job_{i_{v_λ^new}} + c_{r_{v_λ^new},l} ≥ t^earliest_{j,l} then
                I_λ^new := I_λ^new ∪ {j}
```

The differences are first, that not the latest but the earliest time $t^{\text{latest}}_{j,l}$ for a given job-room-combination (j, l) is compared. And second, that the relational operators are changed. The earliest starting time $t^{\text{earliest}}_{j,l}$ has to be greater (instead of less) than $t_{v_{\lambda^{\text{new}}}} + d^{\text{job}}_{j_{v_{\lambda^{\text{new}}}}} + c_{r_{v_{\lambda^{\text{new}}},l}}$ such that a job can be removed from the set. And $t^{\text{earliest}}_{j,l}$ has to be less (instead of greater) or equal than $t_{v_{\lambda^{\text{new}}}} + d^{\text{job}}_{i_{v_{\lambda^{\text{new}}}}} + c_{r_{v_{\lambda^{\text{new}}},l}}$ such that a job has to be added to the set.

Stage 5 – combining the results from forward and backward iteration

The crucial part within the bidirectional algorithm is to join the results of forward and backward iteration in order to generate feasible tours and obtain the optimal solution. Therefore, the labels from useful paths from forward iteration ($\Lambda^{\text{usefulFW}}$) and from backward iteration ($\Lambda^{\text{usefulBW}}$) have to be combined.

However, not all labels can be combined. Only those can be joined that have a connection arc to a node in the other half of the graph. Additionally further resource

constraints must be fulfilled. Let λ^{FW} be one label out of $\Lambda^{\text{usefulFW}}$ and λ^{BW} one label out of $\Lambda^{\text{usefulBW}}$. Then the following conditions must hold:

- The shift type must be equal.

$$b_{\lambda^{\text{FW}}}^{\text{ShortShift}} = b_{\lambda^{\text{BW}}}^{\text{ShortShift}} \wedge b_{\lambda^{\text{FW}}}^{\text{LongShift}} = b_{\lambda^{\text{BW}}}^{\text{LongShift}} \tag{3.30}$$

- The number of assigned breaks must match with the shift type.

$$b_{\lambda^{\text{FW}}}^{\text{ShortShift}} = 1 \leftrightarrow \begin{matrix} b_{\lambda^{\text{FW}}}^{\text{AlreadyBr1}} + b_{\lambda^{\text{BW}}}^{\text{AlreadyBr1}} = 0 \\ b_{\lambda^{\text{FW}}}^{\text{AlreadyBr2}} + b_{\lambda^{\text{BW}}}^{\text{AlreadyBr2}} = 0 \end{matrix} \tag{3.31}$$

$$b_{\lambda^{\text{FW}}}^{\text{ShortShift}} = 0 \wedge b_{\lambda^{\text{FW}}}^{\text{ShortLong}} = 0 \leftrightarrow \begin{matrix} b_{\lambda^{\text{FW}}}^{\text{AlreadyBr1}} + b_{\lambda^{\text{BW}}}^{\text{AlreadyBr1}} = 1 \\ b_{\lambda^{\text{FW}}}^{\text{AlreadyBr2}} + b_{\lambda^{\text{BW}}}^{\text{AlreadyBr2}} = 0 \end{matrix} \tag{3.32}$$

$$b_{\lambda^{\text{FW}}}^{\text{ShortLong}} = 1 \leftrightarrow \begin{matrix} b_{\lambda^{\text{FW}}}^{\text{AlreadyBr1}} + b_{\lambda^{\text{BW}}}^{\text{AlreadyBr1}} = 1 \\ b_{\lambda^{\text{FW}}}^{\text{AlreadyBr2}} + b_{\lambda^{\text{BW}}}^{\text{AlreadyBr2}} = 1 \end{matrix} \tag{3.33}$$

If the shift is a short shift then no break must be assigned (3.31). If the break is a normal shift then exactly one first break has to be assigned but no second break (3.32). If the shift is a long shift both breaks must be assigned exactly once (3.33).

If the conditions (3.30) to (3.33) hold the labels can be combined to one shift. In order to derive this information, labels have to be compared. The process of comparing and combining labels is depicted in Algorithm 4.

Algorithm 4: GENERATE FEASIBLE TOURS BY JOINING LABELS
Input: The lists of useful labels from forward and backward iteration $\Lambda^{\text{usefulFW}}$, $\Lambda^{\text{usefulBW}}$, the duals for the convexity constraint π_p^{conv} and the root label from forward iteration λ_0
Output: The best tour λ^*

```
1 λ* = λ_0
2 foreach λ ∈ Λ^usefulFW do
3     foreach u ∈ V^Succ_{v_λ} do
4         foreach λ' ∈ Λ^usefulBW do
5             if u = v_{λ'} and conditions fulfilled then
6                 λ^combi = f^combi(λ, λ')
7                 if RC_{λ^combi} > RC_{λ*} then
8                     λ* = λ^combi
```

Let λ^* be the best tour that could be found and RC^* the corresponding RC. At the beginning of the algorithm the best tour is set to the root label from the forward iteration and thus the best RC are set to the negative value of the convexity duals. The algorithm examines each element λ in the list of useful paths. The last node of λ is extended to all its possible successors. For every successor it is checked if there is a label in useful paths from backward iteration with the same last node. If

both nodes correspond and the conditions described in (3.30) to (3.33) are fulfilled the labels are joined by the combination function f^{combi}.

In order to safe computational effort, the function hands over only a fraction of information to λ^{combi}; the RC which are needed to decide whether the best tour must be updated and the path which is needed for the L-RMP.
The resulting RC are:

$$\text{RC}_{\lambda^{\text{combi}}} = \text{RC}_{\lambda^{\text{FW}}} + \text{RC}_{\lambda^{\text{BW}}} \tag{3.34}$$

The resulting path is:

$$\Pi_{\lambda^{\text{combi}}} = \langle \underbrace{v^{\text{OUT}}, \ldots, v_{\lambda^{\text{FW}}}}_{\substack{\text{forward} \\ \text{iteration}}}, \underbrace{v_{\lambda^{\text{BW}}}, \ldots, v^{\text{IN}}}_{\substack{\text{backward} \\ \text{iteration}}} \rangle \tag{3.35}$$

If the RC of the combined label are bigger than the best RC so far the best tour is set to λ^{combi}. When all elements in $\Lambda^{\text{usefulFW}}$ are checked λ^* represents the optimal tour for the ThSP-SP.

Stage 6 – post-processing to provide information for L-RMP

In the last step a new tour n^{new} is created to be added to the L-RMP. The path Π_{λ^*} contains all nodes that were visited in the optimal tour. For the nodes representing a real-existing job the following information is extracted.

$\Theta^{\text{match}}_{n^{\text{new}}}$	=	number of matches within tour n^{new}.
$\Theta^{\text{job}}_{n^{\text{new}},i}$	=	equal to one, if in tour n^{new} job i is treated, otherwise zero.
$\Theta^{\text{room}}_{n^{\text{new}},r,t}$	=	equal to one, if in tour n^{new} room $r \in \mathcal{R}^{\text{tc}}$ is occupied at time t, otherwise zero.

When the whole CG process terminates it returns the maximum number of matches for the given ThSP. However, the scheduler in a hospital is not only interested in the maximum number of matches. In particular it is important to assign tours to the therapists. Therefore, the whole path is added to the column n^{new}, too. Then it is possible to extract the shifts represented by a certain column used in the optimal solution after the CG algorithm terminates.

Note that it is not guaranteed that the algorithm terminates with a integral solution. Since only the L-RMP is solved an upper bound on the optimal solution to the RMP is provided.

4 Computational Results

The aim of this section is to evaluate to what extend the developed approaches could be used to solve a real-world ThSP. It is examined how powerful the speed-up means such as graph reduction and LLF and SLL strategies are. Further, it is discussed where most computational effort is needed during the execution of the algorithms and how this issue could be handled.

4.1 Comparison of the Algorithms

During this thesis, two main concepts were exploited to address the ThSP: (1) the IP formulation in Section 2.2 and (2) the DW decomposition in Section 3. The DW decomposition can be subdivided according to the direction of the graph search in the SP. Forward, backward and bidirectional label correcting algorithms were developed.

In order to test the instances in terms of their capability to solve the ThSP appropriate test instances must be defined. Seven instances were created.[5] Starting with the tiny one from Section 2.1 up to an instance of real-world problem size. The instances are named by the combination of the number of therapists, real-existing jobs and rooms. For example a problem set containing 2 therapists, 3 real-existing jobs and 4 rooms is denoted T2-J3-R4.

Out of the 7 instances only the biggest is of real-world problem size. However, the instances are sufficient to draw the limitations of the several approaches.

In order to evaluate the different approaches two criteria are used. The first one is if the algorithm is able to solve the problem to optimality. The second is what time is required to solve it. Note that the scheduler at the hospital needs approximately one hour. Thus, the program should never need more time. One has to add that the DW decomposition only solve the L-RMP. That means that it is possible that fractional solutions are generated. Those solutions have to be treated further by a branch-and-price algorithm which require to solve the ThSP several times. Thus, a time limit of 5 minutes is imposed for each run for the DW approaches and a limit of 45 minutes for the integer program.

The algorithms were coded in Java SE6 environment linked to CPLEX 12.4 as an LP solver and executed on a 2.00 GHz Intel Core i7 Medion Notebook with 6 GB RAM under a 64-bit Version of Windows 7.

[5] The instances can be found on the CD-ROM attached to this thesis.

Table 4.1: Comparision of integer program and dantzig-wolfe decomposition.

	Integer Program		Dantzig-Wolfe		
			Forward	Backward	Bidirectional
Instance	CPU (s)	Solved %	CPU (s)	CPU (s)	CPU (s)
T2-J3-R4	0.4	100.00	0.2	0.2	0.1
T3-J8-R6	38.2	100.00	0.4	0.3	0.4
T4-J8-R12	1.4	100.00	0.4	0.4	0.6
T4-J16-R12	309.5	100.00	5.7	6.2	1.9
T4-J29-R12	> 2700.0	0.00	> 300.0	> 300.0	21.4
T4-J45-R12					146.2
T10-J117-R28					> 300.0

The result for the different algorithm are provided in Table 4.1. For very small instances the computation times almost do not differ. However, with increasing problem size the limitations of the different approaches can be seen. A direct comparison between IP and DW decomposition is not possible since the IP solves to optimality and the DW decomposition provides just an upper bound. A priori it is not known how often the CG algorithm would have to be solved in a branch-and-price algorithm. Hence, the overall run time to solve the problem to optimality can not be determined. However, it can be said that the IP is not able to solve instances bigger than T4-J16-R12. Thus, it could not be used to solve the ThSP in a real-world application.

The DW decomposition in its current design is also not suitable to solve real-world problems. None of the approaches could solve instance T10-J117-R28. The bidirectional algorithm is more powerful than the forward or backward algorithm. The difference in the run times for forward and backward algorithm can be explained by the graph structure. If it is possible to dominate more labels at the starting points of the algorithm the set of generated labels is kept smaller and thus the algorithm faster. If the graph structure allows more domination close to the source the forward iteration is faster. If the graph structure allows more domination close to the sink the backward iteration is faster.

However, none of these algorithms developed in this work is capable of solving real-world problems. An analysis where most computational effort is needed and how the algorithm could further be improved is provided in the subsequent section.

Although, the bidirectional label correcting algorithm is not able to solve the ThSP it already contains some components of acceleration: (1) the bidirectional structure itself, (2) the reduction of the underlying graph for the SP by eliminating nodes and arcs and (3) the LLF and SLL strategies to decide when and where a node enters the lists of unprocessed paths. It is of interest to know to what extend those means were able to improve the algorithm.

The benefit from the bidirectional structure could already be seen in Table 4.1. The bidirectional algorithm was faster and could solve more complex instances. The

benefit from LLF and SLL strategies and the reduction of the graph can be seen in Table 4.2 on page 38. Four cases are compared: (1) the algorithm contains neither LLF and SLL strategy nor the graph reduction, (2) the algorithm contains only LLF and SLL strategy (LLF+SLL), the algorithm contains only the graph reduction ($\mathcal{V}^{\text{red}} + \mathcal{A}^{\text{red}}$) and (4) the algorithm contains both the algorithm contains neither LLF and SLL strategy and the graph reduction (LLF+SLL and $\mathcal{V}^{\text{red}} + \mathcal{A}^{\text{red}}$). The two biggest instances were not tested since already instance T4-J29-R12 yields to a run time greater than 5 minutes.

Both, the label selection strategies and the graph reduction have positive influence on the run time. However, the influence of the LLF and SLL strategy is rather small in comparison to the graph reduction. For example for T4-J29-R12 the label selection strategy reduces the run time by 22% in comparison to the the case without any acceleration means. The graph reduction reduces the run time by 95%. However, when both are applied together the run time can be reduced to 97%.

Consequently, it can be said that the applied means are efficient in terms of speeding up the algorithm. However, they are not able to speed-up the algorithm to solve realistic problem size.

4.2 Analysis of Computational Effort and Approaches for Improvement

In order to be able to further improve the algorithm it must be known where exactly the huge computational effort is generated. As described in section 3 the crucial part within the CG process is the SP. The SP consists of the graph generation, the forward and backward iteration and the combining of the useful labels from forward and backward iteration. The generation of the graph and the combining of the useful paths requires less than a second. Although there might still be space for improvement the waste majority of time is needed during the forward and backward iteration.

The computation time corresponds to the number of generated labels. Table 4.2 reveals this connection. If less labels are generated the optimal solution is found faster. For small instances this does not necessarily hold since in any case only a small number of labels is needed to find the optimal solution. Then the time which is saved by generating fewer labels is outperformed by the additional computation effort for LLF and SLL strategy. However, for bigger instances the correlation holds and a real-world instance for the ThSP would be a big instance. Thus, the question is how it is possible to further reduce the number of generated labels.

The reason for combining forward and backward iteration was to reduce the possible state space. It is more likely that labels dominate each other close to the starting point when the represented path do not differ that much. If one starts from the sink

Table 4.2: Comparision of means to accelerate the algorithm.

	LLF+SLL		LLF+SLL $\mathcal{V}^{\text{red}} + \mathcal{A}^{\text{red}}$		LLF+SLL $\mathcal{V}^{\text{red}} + \mathcal{A}^{\text{red}}$	
Instance	CPU (s)	$\lvert\Lambda\rvert$	CPU (s)	$\lvert\Lambda\rvert$	CPU (s)	$\lvert\Lambda\rvert$
T2-J3-R4	0.3	3 546	0.3	3 522	0.1	211
T3-J8-R6	2.0	123 083	1.8	112 135	0.4	2 471
T4-J8-R12	1.5	101 087	1.6	93 178	0.5	5 146
T4-J16-R12	29.3	4 494 274	24.5	3 431 597	2.0	225 741	1.9	202 162
T4-J29-R12	692.1	50 077 080	540.2	35 266 165	37.8	5 093 585	21.4	2 951 706

Table 4.3: Number of labels from forward and backward iteration.

	Bidirectional	
Instance	$\|\Lambda\|$ forward	$\|\Lambda\|$ backward
T2-J3-R4	78	133
T3-J8-R6	1 706	757
T4-J8-R12	3662	1 444
T4-J16-R12	198 594	3 467
T4-J29-R12	2 940 332	11 156
T4-J45-R12	4 301 204	3 277 363
T10-J117-R28	8 303	15 394

and the source this drastically reduces the number of generated labels. However, it might be that the separation of the graph leads to unbalanced effort for forward and backward iteration. That means that in one part of the graph much more labels are generated and that the idea of the bidirectional algorithm cannot be exploited completely.

Table 4.3 shows the number of generated labels for forward and backward iteration for all instances. The values for instance T10-J117-R28 are the number of labels generated after the last iteration that was completed before the time limit was reached.[6] When looking especially at instances T4-J16-R12 and T4-J29-R12 it seems that the graph is very unbalanced. This can be explained since those instances contain much more jobs in the first half of the shift. For the big instances T4-J45-R12 and T10-J117-R28 where the jobs are distributed more evenly over the shift the unbalance is not so distinctive. Thus, there exist some unbalance between forward and backward iteration. A possibility to impose more balance could be to set the separation time according to the number of jobs or nodes in each half. However, this it is not promising to lead to an improvement such that a real-world problem could be solved.

In case that both parts of the iteration would be balanced still too much labels would be generated. This number has to be reduced. A label is only generated when it is not dominated by another label in useful paths. As described in section 3.2.3 a label has to fulfill the following for conditions to dominate another label.

- The dominating label must have greater RC.
- Both labels must have the same last node.
- Both labels must be of the same shift type.
- The dominating label must have at least the same jobs that still can be visited.

The RC must be greater. Otherwise, labels would dominate which have a negative influence on the objective function. The last three conditions impose that only those

[6] The number of labels is small since only a few numbers of SPs with sparse graph could be solved. In the moment when the L-RMP gets an objective function value greater than zero a lot of duals are generated. Then the SP becomes intractable.

Table 4.4: Comparison of domination rules.

	domination old		domination new	
Instance	CPU (s)	$\|\Lambda\|$	CPU (s)	$\|\Lambda\|$
T2-J3-R4	0.1	211	0.1	167
T3-J8-R6	0.4	2 471	0.4	2 016
T4-J8-R12	0.6	5 146	0.5	2 528
T4-J16-R12	1.9	202 162	1.2	53 809
T4-J29-R12	21.4	2 951 706	4.5	566 258
T4-J45-R12	146.2	7 579 003	105.1	1 747 093
T10-J117-R28	> 300.0		> 300.0	

labels are dominated that do not lead to distinct tours. However, this could be done in a different manner. The condition that both labels must have the same last node can be relaxed. In order to assure that no promising paths are dominated the condition that both have the same follower jobs must be expand.

For the forward iteration it works in the following manner. If a job is a follower job for both labels it is checked if the dominating label can reach the job at least as early as the dominated label. Otherwise the following situation could occur. A label λ_1 has dominated a label λ_2. The optimal path after λ_1 would be to visit the jobs 1, 2 and 3. All jobs can be visited from λ_1 since it has the same set of follower nodes as λ_2. However, it might be that job 1 is reached in moment of time where it is too late to visit job 2. Then the optimal path cannot be constructed. To overcome this problem it must be checked if a job can be reached at least as early as for the dominated label. Then there is no effect on the following path. The domination rules described in section 3.2 are not affected by such a situation. If both labels have the same last node they have the same job-room-time combination and thus they have the same possible starting times for all follower nodes (assumed that both labels have the same set of possible successor jobs).

A comparison of this new domination rule (domination new) and the old one (domination old) can be found in Table 4.4. The new domination rule is more efficient than the old one. For every instance less labels are generated and thus the run time could be reduced. The number of labels is reduced more than the run time could be reduced. The reason is that the new domination rule requires more computational time. However, this is outperformed by the reduction of the number of labels. Unfortunately, again this improvement was not sufficient to solve T10-J117-R28. It seems that the ThSP cannot be solved to optimality for realistic problem size. It might be that more complex and more effective domination rules exist which consider not only the next nodes but a series of nodes or sub-paths. However, those domination rules would require additional computational effort that could exceed the benefits.

A different approach is to relax some of the domination criteria. The resulting solution would not necessarily be optimal. However, it is not needed that the SP is

solved to optimality. It is necessary if the SP returns positive RC. Such a heuristic solution might be generated in appropriate time.

The condition that the RC must be greater cannot be relaxed without imposing that inferior paths displace superior ones. An option is to allow labels to dominate even though not all jobs can be reached or reached in the same time as for the dominated label. The question is when and under which conditions such a domination rule can be applied. When adding the label to the list of unprocessed paths it is unknown how the path will develop. A priori it cannot be said if RC of a certain value or a certain resource consumption will lead to a preferable path when the label is further extended.

Since the resulting path of a label is unknown one could randomly reduce the number of labels added to unprocessed paths. This approach is similar to a greedy randomized adaptive search procedure as described for example in Resende (2009). In such an algorithm the element that is treated next by the algorithm is chosen from a so called restricted candidate list. The list is restricted because it does contain only the "best" candidates. Randomly one element out of the best candidates is chosen.

This setting is not directly applicable to the label correcting algorithm. However, the central idea can be used. As described above it is not possible to judge how good a label is. Thus, there is no list of "best" candidates. For the label correcting algorithm a label can be seen as "best" if it is not dominated. Then a random value is generated indicating if the label is really added to unprocessed paths.

A risk within this dominance rule is that paths are not branched which would lead to positive RC and no other path with positive RC can be obtained. A possibility to counteract this is by imposing that the rule is only applied to labels whose predecessor labels have more than a certain number of possible successor nodes. For example if a label cannot be extended to more than 10 nodes the rule is not applied to the resulting label. If an algorithm with this domination rule is able to solve the SPs quickly an additional mechanism can be used to improve the chance that positive RC are generated. The problem could be solved several times at least until positive RC are found. Only the best solution is kept and then send to the L-RMP. However, a limit on the maximum number of repetition has to be imposed. Otherwise, the algorithm would get stuck in an infinite loop when the best RC are zero.

Let h^{succ} be the number of successor nodes that must be exceeded that the algorithm can be applied. Let h^{rand} be the random number which has to be undercut that the algorithm can be applied and let h^{count} be the number of repetitions each SP is solved.

For the arbitrary chosen values of $h^{\text{succ}} = 10$, $h^{\text{rand}} = 0.1$ and $h^{\text{count}} = 4$ the resulting solutions are as found in Table 4.5. For the smaller instances more labels are generated during the execution of the randomized algorithm. The graph for those

Table 4.5: Results from the randomized algorithm.

	exact		randomized		
Instance	CPU (s)	$\|\Lambda\|$	CPU (s)	$\|\Lambda\|$	solve %
T2-J3-R4	0.1	211	0.2	762	100.00
T3-J8-R6	0.4	2 471	0.5	5 318	100.00
T4-J8-R12	0.6	5 146	0.5	6 721	100.00
T4-J16-R12	1.9	202 162	1.1	40 084	88.20
T4-J29-R12	21.4	2 951 706	3.5	362 407	92.75
T4-J45-R12	146.2	7 579 003	141.0	7 457 027	45.08
T10-J117-R28	> 300.0		> 300.0		

instances is small and each node does not have many successors. Thus, the domination is not applied very often. In addition, the SPs are solved several times and thus the same problems are generated again even though they were already solved to optimality. For larger instances it seems that the number of generated labels and the run time decreases. However, no reliable statement can be made concerning the randomized label correcting algorithm. The results differ too much depending on the instance. The results for instance T4-J29-R12 are satisfying. It is solved quickly and the objective function value is close to optimal. In contrast, solving instance T4-J45-R12 requires almost the same computational effort as for the exact approach and the solution is far from optimal. Again, the biggest instance could not be solved. However, it should be noted that more columns were generated until the time limit was reached.

It might be that the randomized bidirectional label correcting algorithm is able to solve the ThSP for real-world instances. The quality of the solution depends on how well the parameters (h^{succ}, h^{rand} and h^{count}) fit the problem instance. In order to figure out how parameters can be adjusted optimally to the instance more research has to be done.

The resulting solutions might not solve the problem to optimality. However, a solution close to optimal could also be a considerable improvement of the daily scheduling at hospitals.

5 Conclusion

This thesis introduced the hospital-wide ThSP. To the best of our knowledge it was the first time that scheduling and routing decisions were combined in a hospital related context. The objective was to maximize the matches of patients and their preferred therapist for a certain treatment. An IP formulation and a DW decomposition were developed. The SP was solved using a bidirectional label correcting algorithm to find a SPP through a graph representing possible shifts. It was examined to what extend those approaches were capable to solve problems of real-world application size. Unfortunately, it seems that the ThSP cannot be solved to optimality for such instances. Although some progress has been made using the bidirectional label correcting algorithm. This provides only a starting point for further research.

In order to address the ThSP heuristically a randomized version of the bidirectional label correcting algorithm was developed. This algorithms is promising to perform well on the instances when the adjusting parameters are set accordingly. However, an extensive survey and real-world data from hospitals are needed to finally judge if the randomized algorithm can produce appropriate schedules.

In general, further research should focus on solving the SP since most computational effort was bound at this stage. The SP should be solved more efficiently and less often. The latter can be reached if the set of initial columns for the L-RMP is improved. For example a heuristic could be used to construct a feasible tour. The resulting dual variable would be better than the ones generated by the dummy tour. The second aspect is that the SP not necessarily has to be solved by a shortest path algorithm. This approach was chosen since the problem should be solved to optimality. However, if one wants to solve the ThSP heuristically also other algorithms capable to construct a tour could be developed.

If once the CG algorithms can be solved in a timely manner just an upper bound is obtained. That means the solution might not be integral. In order to generate usable schedules for the hospital the ThSP could be embedded in a branch-and-price algorithm.

A completely different approach would be to treat the problem entirely heuristically. Depending on the hospital's current planning a heuristic schedule could still be an improvement. An advantage would be that heuristic solutions are generated quite fast. On the other hand, if the same solution is used as a starting point for the CG it is promising to obtain better results.

Apart from the issues on solving the problem efficiently one could also think about

different objectives and constraints. It might be that other hospitals want to equally distribute the workload among the therapists. An interesting problem is how walking times would then be considered. Walking times definitely belong to working time. However, it must be avoided that therapists walk unnecessary long ways just to have an equally distributed workload. Another possible objective could be the minimization of the walking times in order to minimize time where no revenue is generated for the hospital. Also multi-criteria objective function could be used.

From a constraint point of view one could consider that for certain treatments different material is needed (e.g. gymnastic ball). Thus, the treatment can only be executed when the equipment is available. This holds for all resources that could be taken into account. It might also be that more than one therapist is needed for a certain treatment. Hence, those therapists would have to execute this treatment at the same time.

The ThSP might also be applicable to other logistic settings where a certain task could be done in more than one place. A capacity could be added to a therapist and one can think of the therapist as a vehicle. If the problem instance is not too big already the algorithm presented in this thesis could be applied.

All in all, by addressing the hospital-wide ThSP a novel research problem got formulated. This thesis provides a starting point and various options exist for the research community to continue and expand this work in future.

Bibliography

Ahuja, Ravindra K., Thomas L. Magnanti, James B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Bard, Jonathan F., Yufen Shao, Ahmad I. Jarrah. 2013a. A sequential grasp for the therapist routing and scheduling problem. *Journal of Scheduling* 1–25.

Bard, Jonathan F., Yufen Shao, Huan Wang. 2013b. Weekly scheduling models for traveling therapists. *Socio-Economic Planning Sciences* **47**(3) 191–204.

Beasley, John E., Nicos Christofides. 1989. An algorithm for the resource constrained shortest path problem. *Networks* **19**(4) 379–394.

Bellman, Richard. 1958. On a routing problem. *Quarterly of Applied Mathematics* **16** 87–90.

Bertsekas, Dimitri P. 1998. *Network Optimization: Continuous and Discrete Models*. Athena Scientific, Belmont, Massachusetts, USA.

Blum, Karl, Sabine Löffert, Matthias Offermanns, Petra Steffen. 2013. Krankenhaus Barometer Umfrage 2013. [Hospital barometer survey 2013.] (In German).

Braaksma, Aleida, Nikky Kortbeek, Gerhard Post, Frans Nollet. 2012. Integral multidisciplinary rehabilitation treatment planning. URL `http://doc.utwente.nl/80926/`.

Ceselli, Alberto, Giovanni Righini, Matteo Salani. 2009. A column generation algorithm for a rich vehicle-routing problem. *Transportation Science* **43**(1) 56–69.

Ceselli, Alberto, Giovanni Righini, Emanuele Tresoldi. 2013. Combined location and routing problems for drug distribution. *Discrete Applied Mathematics* doi: http://dx.doi.org/10.1016/j.dam.2013.07.016.

Chien, Chen-Fu, Fang-Pin Tseng, Chien-Hung Chen. 2008. An evolutionary approach to rehabilitation patient scheduling: A case study. *European Journal of Operational Research* **189**(3) 1234–1253.

Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. 2009. *Introduction to Algorithms*. 3rd ed. The MIT Press.

Dantzig, George B., Philip Wolfe. 1960. Decomposition principle for linear programs. *Operations Research* **8**(1) 101–111.

Desrochers, Martin, Jacques Desrosiers, Marius Solomon. 1992. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research* **40**(2) 342–354.

Desrosiers, Jacques, Yvan Dumas, François Soumis. 1986. A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows. *American Journal of Mathematical and Management Sciences* **6**(3&4) 301–325.

Desrosiers, Jacques, Marco E. Lübbecke. 2005. A primer in column generation. Guy Desaulniers, Jacques Desrosiers, MariusM. Solomon, eds., *Column Generation*. Springer, 1–32.

Dijkstra, Edsger W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik* **1**(1) 269–271.

Eksioglu, Burak, Arif Volkan Vural, Arnol Reisman. 2009. The vehicle routing problem: A taxonomic review. *Computers Industrial Engineering* **57**(4) 1472–1483.

Feillet, Dominique, Pierre Dejax, Michel Gendreau, Cyrille Gueguen. 2004. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks* **44**(3) 216–229.

Griffiths, Jeffrey Deacon, Janet Elizabeth Williams, Richard Max Wood. 2012. Scheduling physiotherapy treatment in an inpatient setting. *Operations Research for Health Care* **1**(4) 65–72.

Hans, Erwin W., Mark Houdenhoven, Peter J.H. Hulshof. 2012. A framework for healthcare planning and control. Randolph Hall, ed., *Handbook of Healthcare System Scheduling*. Springer.

Hulshof, Peter J.H., Nikky Kortbeek, Richard J. Boucherie, Erwin W. Hans, Piet J.M. Bakker. 2012. Taxonomic classification of planning decisions in health care: a structured review of the state of the art in or/ms. *Health Systems* **1** 129–175.

Irnich, Stefan, Guy Desaulniers. 2005. Shortest path problems with resource constraints. Guy Desaulniers, Jacques Desrosiers, Marius M. Solomon, eds., *Column Generation*. Springer.

Klinikum Freising. 25.11.2013. Leitbild Klinikum Freising. [Role model of Freising hospital.] (In German). URL `http://www.klinikum-freising.de/klinikum/leitbild.html`.

Lanzarone, Ettore, Andrea Matta, Gianlorenzo Scaccabarozzi. 2010. A patient stochastic model to support human resource planning in home care. *Production Planning Control* **21**(1) 3–25.

Lenstra, J. K., A. H. G. Rinnooy Kan. 1981. Complexity of vehicle routing and scheduling problems. *Networks* **11**(2) 221–227.

Liu, Ran, Xiaolan Xie, Vincent Augusto, Carlos Rodriguez. 2013. Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care. *European Journal of Operational Research* **230**(3) 475–486.

Lübbecke, Marco E., Jacques Desrosiers. 2005. Selected topics in column generation. *Operations Research* **53**(6) 1007–1023.

Mankowska, DorotaSlawa, Frank Meisel, Christian Bierwirth. 2013. The home health care routing and scheduling problem with interdependent services. *Health Care Management Science* 1–16.

Pop, Petrică C., Imdat Kara, Andrei Horvat Marc. 2012. New mathematical models of the generalized vehicle routing problem and extensions. *Applied Mathematical Modelling* **36**(1) 97–107.

Pugliese, Luigi Di Puglia, Francesca Guerriero. 2013. A survey of resource constrained shortest path problems: Exact solution approaches. *Networks* **62**(3) 183–200.

Resende, Mauricio G.C. 2009. Greedy randomized adaptive search procedures. Christodoulos A. Floudas, Panos M. Pardalos, eds., *Encyclopedia of Optimization*. Springer US, 1460–1469.

Schimmelpfeng, Katja, Stefan Helber, Steffen Kasper. 2012. Decision support for rehabilitation hospital scheduling. *OR Spectrum* **34**(2) 461–489.

Shao, Yufen, Jonathan F. Bard, Ahmad I. Jarrah. 2012. The therapist routing and scheduling problem. *IIE Transactions* **44**(10) 868–893.

Statistisches Bundesamt. 2010. Demografischer Wandel in Deutschland. Heft 2: Auswirkungen auf Krankenhausbehandlungen und Pflegebedürftige im Bund und in den Ländern. [Demographic change in Germany. Number 2: Impact on hospital treatments and persons in need of nursing in the Federation and the States.] (In German).

Statistisches Bundesamt. 2011. Fachserie 12 Reihe 6.2.2: Diagnosedaten der Patienten und Patientinnen in Vorsorge- oder Rehabilitationseinrichtungen. [Diagnostic data of patients in provision and rehabilitation centers.] (In German).

Statistisches Bundesamt. 2013. Press release 359/13: 51 Millionen Operationen und medizinische Prozeduren bei stationären Patienten 2012. [51 million surgeries and medical procedures for in-patients in 2012.] (In German).

Wolsey, L. A. 1998. *Integer programming*. Wiley-Interscience, New York, NY, USA.

Appendix

A Additional Algorithms

Algorithm 5 depicts the course of the backward iteration for the label setting algorithm described in Section 3.2.3.

Algorithm 5: LABEL CORRECTING ALGORITHM (BACKWARD ITERATION)
Input: Sub-graph $\mathcal{G}^{\text{backward}}$ and all information from nodes and arcs included in it
Output: A set of useful paths from the forward iteration

```
1  create root label λ_0
2  Λ^unprocessed := {λ_0}
3  Λ^useful := ∅
4  b^dominant := false
5  b^inferior := false
6  while Λ^unprocessed ≠ ∅ do
7      λ ← Λ^unprocessed[0]
8      Λ^unprocessed := Λ^unprocessed \ {λ}
9      foreach λ' ∈ Λ^useful do
10         counter := 0
11         if λ > λ' then
12             if counter := 0 then
13                 b^dominant := true
14                 Λ^useful := {Λ^useful ∪ {λ}} \ {λ'}
15                 counter := 1
16             else
17                 Λ^useful := Λ^useful \ {λ'}
18         else
19             if λ < λ' then
20                 b^inferior := true
21     if b^dominant = false and b^inferior = false then
22         Λ^useful := Λ^useful ∪ {λ}
23     if b^inferior = false then
24         foreach u ∈ V^Pred_{v_λ} do
25             if path can be extended to u then
26                 create new label λ'
27                 Λ^unprocessed := Λ^unprocessed ∪ {λ'}
```

Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich, dass ich die vorliegende Arbeit selbständig angefertigt habe. Die aus fremden Quellen direkt und indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Ich weiß, dass die Arbeit in digitalisierter Form daraufhin überprüft werden kann, ob unerlaubte Hilfsmittel verwendet wurden und ob es sich - insgesamt oder in Teilen - um ein Plagiat handelt. Zum Vergleich meiner Arbeit mit existierenden Quellen darf sie in eine Datenbank eingestellt werden und nach der Überprüfung zum Vergleich mit künftig eingehenden Arbeiten dort verbleiben. Weitere Vervielfältigungs- und Verwertungsrechte werden dadurch nicht eingeräumt. Die Arbeit wurde weder einer anderen Prüfungsbehörde vorgelegt noch veröffentlicht.

München, 29. November 2013

Unterschrift