Integrated planning and scheduling in operational patient management

Atle Riise

2015

Series of dissertations submitted to the
Faculty of Mathematics and Natural Sciences, University of Oslo
No. 1697

ISSN 1501-7710

Cover: Hanne Baadsgaard Utigard.
Print production: John Grieg AS, Bergen.

Produced in co-operation with Akademika Publishing.
The thesis is produced by Akademika Publishing merely in connection with the thesis defence. Kindly direct all inquiries regarding the thesis to the copyright holder or the unit which grants the doctorate.

Abstract

This thesis addresses a class of particularly challenging real world optimisation problems that occur in operational patient management in hospitals. These problems are found in surgery scheduling, as well as in scheduling for various non-surgical treatment or diagnostic services. As a class, these problems can be identified by the following properties: They concern the scheduling of a (potentially large) set of patients, where one or several activities must be scheduled for each patient. A set of resources is required for each activity, and there are many alternative such sets to choose from (the problems are *multi-modal*). Furthermore, the problems involve both planning and scheduling decisions; both a day and an exact start time must also be chosen for each activity. These problems are subject to resource capacity constraints and a variety of time constraints, often including both minimum and maximum time lags, as well as time windows. Common objective functions involve both the quality of treatment and an efficient use of hospital resources. In general, these problems have a high computational complexity, and in many cases even simpler subproblems have been shown to be NP-hard.

In this thesis we show that a majority of these real world problems can be modelled in a unified way, without artificial simplification, and that structurally different problem instances can be solved efficiently based on such a model. We develop several efficient search methods, both heuristic and exact, and test these on real world problem instances. We also investigate the potential of 'same program multiple data' parallelisation for use in algorithms for our chosen problem class.

The scientific contribution of this thesis is presented through five papers. Three of these were published in (or submitted to) 'Level 2' journals[1].

[1]Norwegian classification of journals; Level 2 is the top level.

In Paper I, we consider inpatient surgery admission planning. The paper employs a meta-heuristic algorithm, based on iterated local search and variable neighbourhood descent, and demonstrates its effectiveness on realistic problems instances from a Norwegian hospital. The paper also presents a search space analysis for different neighbourhood operators and fitness surfaces, and offer conclusions on their suitability for local search algorithms. The problem involves integrated planning and scheduling on a planning horizon of three weeks. The paper was published in the Journal of Heuristics.

For Paper II, we investigated the potential of utilising massively parallel computation hardware for solving permutation-like optimisation problems. At the time, hardware for 'same program multiple data' parallelisation, like the General Purpose Graphical Processing Unit (GPGPU), had been applied to many scientific computation tasks. There were, however, very few applications for local search methods, and we wanted to investigate how this emerging technology can enable novel ways to design such methods. The paper shows how massively parallel neighbourhood evaluation can facilitate the simultaneous application of many independent improving moves. This gives considerable improvements in performance, in addition to the one achieved from the parallel evaluation in itself. Paper II was published in the Journal of the Operational Research Society.

In Paper III, we present a generic model for the 'generalised surgery scheduling problem', expressed as an extension of the multi-mode resource constrained project scheduling problem with minimum and maximum time lags. We show that this model covers many of the problem variations that are addressed in the literature. The paper also presents an algorithmic framework for iterative schedule construction and improvement, based on sampling and modification of the project insertion order, respectively. The methods use a sequential schedule generation scheme that is modified to handle the rich set of model constraints. The model can be applied without modification to surgery scheduling problems that arise in three very different planning situations. Our numerical experiments show, based on realistic data from a Norwegian hospital, that the presented method provides high quality solutions for these problems, in a short time. Paper III was accepted for publication in Computers & Operations Research in July 2015.

The general model of Paper III is also applicable to the scheduling of other kinds of treatment, examination, or control activities. Indeed, Paper IV applies

the same underlying model to another real world application; multi-mode appointment scheduling in outpatient clinics. In this paper, we exploit the multi-period nature that is typical for this class of problems. We show how one may construct a three-level version of logic based Benders decomposition to solve large, real world problem instances to optimality in a very short time. Paper IV was submitted to the European Journal of Operational Research in August, 2015.

Finally, Paper V gives an introduction to some topics in hospital resource management and patient scheduling. It was published as a chapter in the *Handbook of Healthcare Delivery Systems*.

Preface

This thesis is submitted in partial fulfilment of the requirements for the degree of Philosophiae Doctor (PhD), at the Faculty of Mathematics and Natural Sciences, at the University of Oslo.

The work has been carried out in association with two research projects, both sponsored by the Research Council of Norway. The first, 'Optimisation methods in health care planning software' (HOSPITAL), had surgery scheduling as one of its main topics. This project was initiated by the candidate, together with colleague Martin Stølevik, and included collaboration with Bærum hospital, and the software vendors DIPS and Gatsoft. DIPS is a market leader in Norway in activity planning software for hospitals. The second project, 'Tools for unified activity planning and scheduling in hospitals' (AKTIV), has a broader scope. It considers not only surgery, but also other kinds of treatment and diagnostic services. This project was also initiated by the candidate. The project involves a close collaboration with the hospitals Drammen hospital and the University Hospital of Northern Norway, as well as with DIPS. The collaboration in these projects has provided application knowledge and insight into the optimisation problems that we address in this thesis. The hospitals have also provided the data that we have used in our experiments.

For the duration of this thesis, the candidate has been employed by SINTEF ICT, at the Department of Applied Mathematics.

Thesis structure

This thesis consists of two parts. Part I provides an introduction, and gives an overview of our contribution: In Chapter 1 we introduce the problem domain, and the main research challenges. A more detailed description of the class of optimisation problems that we consider is given in Chapter 2. Chapter 3 provides

an introduction to common applications, and an overview of related work. The scientific contributions are summarised in Chapter 4, with a discussion of each paper. Finally, we conclude in Chapter 5, and discuss some interesting directions for further research. Part II contains the scientific papers.

Acknowledgements

As the work with this thesis has taken some time, many people have been involved in the process. I would like to thank my main supervisor, Professor Edmund K. Burke for offering to take me on, and for his insights and contributions throughout these years. I would also like to thank Professor Carlo Mannino, who entered the roles of co-supervisor, colleague, and friend in the latter part of the process. I have learned a lot from our discussions, and hope to continue to do so. A big thanks also to my colleagues in the Department of Applied Mathematics, and especially the good people of the Optimisation group, for maintaining such a friendly and constructive working environment. A special thanks to Oddvar Kloster and Leonardo Lamorgese, with whom I've worked closely in the AKTIV project and had many interesting discussions. Also thanks to Geir Hasle, who employed me at SINTEF all those years ago, and whose input has always been appreciated. Also, I would like to thank my friends for being just that; I'm looking forward to more frequent interactions. Special thanks to Kristin and Arild, who beat me to it. They have been offering encouragement as well as practical help throughout. Finally, I would like to thank my family. Most of all, my wonderful girlfriend and cohabitant Kristin, whose patience and loving support has been indispensable to me. I deeply value our relationship. That post-thesis life starts now! Also, thanks to my daughter, Tanvi. A child when all this began, she has always been a wonderful person and a joy in my life. And to Rufus the dog, for all those long days at the office. Any and all mistakes in this thesis are his.

Contents

Part II Scientific Results

Part I
Background

Chapter 1

Introduction

Hospitals are under pressure to provide high quality care within limited budgets. At the same time, national health care expenditures are increasing, world wide [109]. Hospitals are therefore looking for ways to reduce cost, and one way of doing this without reducing the quality of care is through better use of critical and expensive resources. Hospital resource management and patient scheduling are topics that have been much studied in the scientific field of *operations research* [44, 83, 124, 46, 33]. These topics contain optimisation problems on different time scales and levels of detail, often categorised as *strategic*, *tactical*, or *operational* [15, 26, 42, 104, 67, 83, 54]. On the tactical level one finds decision problems related to patient mix and overall dimensioning of resources, usually on a long time horizon (year). Based on such overall decisions, the tactical decision level is concerned with the allocation of resources to different *specialities*[1], resulting for example in a *master surgery schedule* for operating rooms [16, 134, 47, 99]. These tactical resource allocations in turn impose constraints on the operational daily planning and scheduling of treatment activities.

It is this last class of operational problems that we consider in this thesis. It is critical for the overall performance of a hospital that such problems are solved efficiently [57, 26, 70]. Some of these problems, such as surgery scheduling, have received a lot of attention in the operations research literature. Still, however, real

[1]Each hospitals is organised in a different way. Here, and in the following, we use the term 'speciality' to denote an organisational unit that specialises in a certain branch of medicine or surgery (such as for example a 'Department of orthopaedic surgery'). Each speciality is typically responsible for the treatment of a certain group of patients.

world applications continue to raise hard research challenges. Also, the situation remains that most hospitals do their scheduling manually, or with software tools that do not take advantage of these research results. This gap between academic research and actual application is to a large extent a result of the complexity of hospital operations, and the diversity of planning situations and problem definitions that can be found in hospitals. This diversity is reflected in the literature, which covers a large variety of such planning and scheduling problems. In addition, there is often a certain degree of simplification of the problem at hand, which reduces the applicability of the research results to real world applications. These are some of the reasons why advanced production scheduling tools are less common in hospitals than in, say, the production industry. There is, therefore, still a significant benefit to be achieved if this gap can be bridged, both in terms of economics and in terms of general quality of care in hospitals.

The work in this thesis is motivated by the recognition of several factors:

1. There is a a need for richer, more realistic, modelling of real world activity planning and scheduling problems [67]. If research in this field is to support software applications for activity planning and scheduling, such a model has to be general enough to express most problems, without excessive customisation for each application.

2. These problems are typically of a high computational complexity - several have been shown to be NP-hard (see [59] for an introduction to complexity theory). Since most real world instances are also quite large, they are challenging to solve to a sufficient quality within a reasonable time limit. This has led to a number of problem simplifications, both in actual hospital management and in the research literature. Solving realistic models for these problems, without making such simplifications, therefore poses a considerable research challenge.

3. One simplification that is frequently used in the literature is to consider planning and scheduling as two separate, consecutive, steps. We use the term *planning* here to imply a choice of day, and a choice of resources, for each activity. By *scheduling* we mean a choice of activity start times[2,3].

[2]This, of course, also implies a sequencing of activities on each resource, for each day.

[3]Note that we still use the term 'surgery scheduling' as a name for a class of problems. In

This two-step approach of planning and scheduling is usually motivated by the need to reduce computational complexity, but has been shown to give inferior solutions. It is therefore important to integrate planning and scheduling decisions, and this is a key aspect of the presented work. We will elaborate on this in Section 2.1.

We choose to address a general, but challenging, class of real world activity planning and scheduling problems, in which planning and scheduling decisions are integrated, and multiple alternative resources must be chosen for each activity.

This thesis aims to answer the research questions:

- *How can this class of rich and computationally complex problems be modelled realistically, in a generic way?*
- *How can one design search methods that can find high quality solutions to realistic problem instances from different planning situations, within a time limit that is acceptable in the practical planning situation?*

We aim to answer these questions through a series of research papers, and through the discussion in this introductory part of the thesis. As will become apparent, our aim has been to find the right methods to solve actual, real world problems, rather than to adapt each problem to fit a certain type of method.

that context, the word 'scheduling' does not refer only to a choice of start times; indeed, surgery scheduling problems often contain planning decisions. With this in mind, however, it should be clear from the context what the meaning is in each case.

Chapter 2

Scope and problem description

2.1 The activity planning and scheduling problem

Operational activity planning and scheduling in hospitals include a vast range of optimisation problems. These are found in many different planning situations, with different characteristics in terms of the number patients, the number of activities per patient, the number of resources and the length of the planning horizon. For example, a short term surgery scheduling problem may consider several activities per patient (including the surgery itself), and many resources (preparation rooms, operating rooms, surgeons, anaesthesiologists, nurses, an intensive care unit, mobile equipment, cleaning staff, and so on). On the other hand, a long term scheduling of elective outpatients[1] for a gastroenterology clinic may consider very many patients and several types of resources (for example doctors, rooms and equipment), but perhaps only one

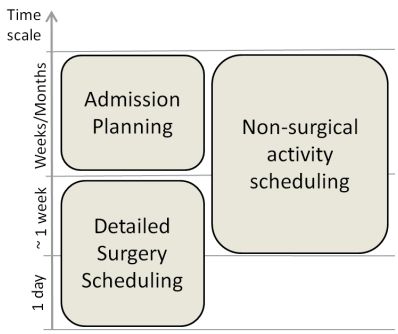

Figure 1: APSP applications.

[1] An *outpatient* is a patient that shows up for an appointment, and then leaves the hospital on the same day, without being admitted. *Inpatients* are patients that are admitted to the hospital, for example on the day before their surgery.

activity per patient. In this thesis, we consider real world problems that involve many treatment activities, and where a *day*, a *start time* and a *set of resources* have to be chosen for each activity. In the following discussion, we will refer to such problems collectively and generically as the *activity planning and scheduling problem* (APSP). Such problems are ubiquitous in hospitals, and include surgery scheduling on different time scales, as well as scheduling of non-surgical treatment and diagnostic activities (see Figure 1). We will define this class of problems in more detail in Section 2.2. As an introduction, however, it can be characterised by the following properties:

- Many patients are to be scheduled, and for each patient there may be one or several treatment activities, often with time windows and time related preferences. There may also be time constraint between the activities in each project.
- Each activity requires one or several resources, and there are several possible combinations of resources that can perform the activity. One such combination, (a *mode*), must be chosen for the activity. The activity duration often depends on the mode, because it depends on the experience or capabilities of the mode's resources.
- Most resources are renewable, and may have a setup time. In general, the APSP may also contain non-renewable resources, but we have so far not encountered a problem where non-renewable resources are critical.
- Resources are available only part of the time. The availability is often governed by a combination of factors, such as tactical resource allocations, working hours, lunch breaks, scheduled meetings, etc. For most problems that we consider, the availability of all resources can be partitioned into disjoint time intervals, often one per day. This multi-interval structure is an important property of these problems.
- There may be compatibility constraints between activities and resources, based on skills, and even between resources that cannot be used together.
- There may be constraints or preferences to use the same resources across several activities for the same patient.

- Both planning and scheduling decisions are included, which means that in addition to the choice of resources, both a day, and a precise start time must be chosen for each activity.

The last point deserves some elaboration. In the literature, surgery scheduling that cover more than one day is often considered as a two-step process [56, 57]. The first step considers planning decisions, while the second step concerns scheduling, based on the planning step solution. In the planning step, a day of surgery, and possibly an operating room, is chosen for each patient on the waiting list. This is often called admission planning, but is also known as *advance scheduling* [98, 114], *operating theatre planning* [68, 30] or *surgery loading*[72]. The scheduling step of this approach, sometimes called *allocation scheduling* [98], is to create a schedule for each day. This may happen at a much later point, closer to the day of surgery, and typically involve more activities for each patient, and more resources.

The advantage of this two-step approach is that it reduces the problem complexity, even if both steps are still, at least in their general versions, NP-hard [72, 25]. The drawback of the two-step approach, however, is that the assignments from the planning step may not permit a reasonably good, or even a feasible, schedule to be found in the scheduling step. This is, of course, because the scheduling is bound by the previous planning decisions, which was made without consideration of the scheduling step's constraints and objectives. For example, as pointed out by Cardoen et al [24], the quality of the surgery sequence that can be achieved in the second, detailed, surgery scheduling step is highly dependent on the assignment to days and rooms that was made in the admission planning step. Similar observations are made in [22].

The same two-step approach to planning and scheduling have also been used in other application areas, for the same reasons, and with the same inherent problems. See, for example, [100] for a discussion of integrated production planning and scheduling in manufacturing. They show that the inclusion of the scheduling problem is necessary to give good enough information about feasibility and costs at the time of planning, especially when production systems are operating at near full capacity.

These observations are also consistent with our experience from Norwegian hospitals. While at first glance it seems that many hospital in practice follow the

two-step approach, this is not completely true. For example, even if the admission plan only contains a choice of day for each patient, the admission planner actually considers sequencing and scheduling preferences to ensure that this assignment will give good day schedules at some later point. This is often done indirectly, based on experience or some agreed rules of thumb. One example is a *time-of-day* preference to schedule certain patients, such as children or patients with diabetes, early in the morning. Rather than formally including this scheduling preference in the problem, the planner may use a rule of thumb that limits the number of children to plan per day. However, while these rules of thumb to a certain degree represent scheduling considerations in a manual planning situation, they are still only approximations of the actual scheduling preferences and constraints. The resulting plans will therefore often be suboptimal, or even infeasible. When we model real world surgery planning and scheduling problems, we therefore do well to replace these rules of thumb with the actual time and resource considerations, and to solve the actual combined planning and scheduling problem.

Motivated by these realities of the application, we take the view that APSPs are best solved by integrating the planning and scheduling decisions, for both short and long term problems. In conjunction with multiple modes and general time constraints, this makes the APSP very hard to solve. Solving these difficult problems efficiently is the main focus of this thesis.

2.2 Modelling

We believe that the diversity of real world APSPs is best handled by establishing a generalised model. Like some other authors [120, 127, 128], we find it useful to formulate such a model based on a well-known problem definition. In Paper III, we present such a general model that extends the multi-mode resource constrained multi-project scheduling problem with minimum and maximum time lags, which in turn extends the classical resource constrained project scheduling problem (RCPSP). Based on the taxonomy of Cardoen et al [26], Paper III shows that this model covers most of the relevant aspects found in surgery scheduling for elective patients. While originally formulated for surgery scheduling, the model also applies to other APSPs. An exact mathematical model can be found in [126]. In this section we provide a shorter, less formal, definition of the APSP.

First, we have to introduce the basic RCPSP and some of its common exten-

sions. The RCPSP concerns the scheduling of the activities $\mathcal{N}^p$ of a single project p. Each activity demands a certain amount of each of a set of renewable resources $\mathcal{R}$, and each resource has a limited, constant, capacity. Preemption is not allowed. Precedence constraints (minimum time lags of size zero) can be defined between individual pairs of activities. The problem may be modelled as a directed activity-on-node *project graph*, where an artificial start node represents the start of the project, and another artificial end node represents the completion of the project. Each precedence constraint can be expressed as an arc in this graph, from the predecessor to the successor. The objective is to minimize the project makespan (the total duration of the project), which is bounded from below by the longest path in the project graph, from the start node to the end node. Several text books offer introductions to the RCPSP; see for example [2] or [107]. In the classification notation of Brucker et al [21], the RCPSP is labelled $PS|prec|Cmax$. Even in this basic form, the RCPSP can be shown to be NP-hard [17]. As a generalisation of this problem, the APSP is also NP-hard.

A number of generalisations and extensions to the RCPSP have been studied over the years; see [76] for a recent survey of these. We shall present three of these extensions: the multi-project extension, the multi-mode extension, and the extension with generalised precedence constraints. In the multi-project RCPSP, a set of projects $\mathcal{P}$ are to be scheduled. Each project can be represented with its own project graph, as there are no precedence constraints between activities in different projects. All projects share the same pool of resources, $\mathcal{R}$ [122].

The multi-mode extension requires that a set of resources is chosen for each activity. This is modelled using the concept of *modes*: For each activity i, a set of feasible modes $\mathcal{M}^i$ are defined out of which exactly one must be selected. Each mode defines a set of resources $\mathcal{R}^m \subseteq \mathcal{R}$ that together can perform the activity, and a demand μ_r^m for each $r \in \mathcal{R}^m$. Each mode also defines an associated activity duration [52]. This extension is the multi-mode resource constraint project scheduling problem (MRCPSP). In the three field notation of [21], the problem is labelled $MPS|prec|Cmax$.

The third extension comes from replacing the precedence constraints with more general minimum and maximum time lags between the activities in a project. These constraints can be expressed as arcs with non-negative and non-positive weights, respectfully, in the project graph. This problem is called the RCPSP with

minimum and maximum time lags (RCPSP/Max), or the RCPSP with generalised precedence relations (RCPSP-GPR). In the three field notation of [21], the problem is labelled $PS|temp|Cmax$.

Taking the above extensions together, one has the multi-mode resource constrained multi-project scheduling problem with minimum and maximum time lags (MRCMPSP/Max in the following). The APSP is similar to this problem, but it also has some additional constraints, as will be explained below. The problem defines a set of projects $\mathscr{P}$, where each project represents the treatment of one patient. There may be time windows, both for projects and individual activities, that constrains their earliest start and latest completion. As for many other real world scheduling problems, there may be instances where not all projects can be feasibly scheduled. It is therefore necessary to choose a subset of projects, $\tilde{\mathscr{P}} \subseteq \mathscr{P}$, to schedule. This means that there is always at least one feasible solution to the APSP, where $\tilde{\mathscr{P}} = \emptyset$. The penalty for not scheduling a patient is a component of the objective function, with individual weights for each patient depending on the urgency of their treatment. The set of feasible modes for each activity depends on resource skills and other resource/activity (or resource/resource) compatibility constraints that may apply. Skills are typically linked to the ability to perform a certain medical procedure, or to use a certain type of equipment. For each resource, a set of non-overlapping time intervals are defined, in which the resource is available with a certain capacity. These *resource availability intervals* can be made exclusive for certain patients, to model that certain blocks of resource time have been reserved for individual surgeons, or for certain specialities.

The APSP also contains some constraints that we believe are not previously studied in the literature. One of these are general *mode consistency* constraints, which links the choices of modes for activities in the same project. This is used to model, for example, that if a certain operating room is used in the preparation of a patient, that same operating room should be used for the actual surgery, as well as in the following room cleaning activity. This is a generalisation of the *same mode constraints* in [50]. Another new extension is the *project disjunction* constraint, which expresses the fact that a resource, say an operating room, can only be used for one project at the time. This comes from the fact that as long as all activities for one patient are not completed in the operating room, no activity concerning another patients can happen in the same room (even if there is time and capacity

available on the resource).

Let us illustrate the various concepts and constraints of the APSP by a simplified example. Consider a problem where $\mathcal{R} = \{a, b\}$, where the resources a and b have availability intervals $\mathcal{K}^a = \{k_1^a, k_2^a\}$ and $\mathcal{K}^b = \{k_1^b\}$, respectively. Let k_1^a span the time interval $[4, 12]$, which falls on day 1 of the planning period. On day 2, k_2^a spans $[16, 22]$ and k_1^b spans $[13, 19]$. All the resource availability intervals have capacity 1. Let $\mathcal{P} = \{p_1, p_2, p_3, p_4\}$, where each project $p \in \mathcal{P}$ has two activities, $\mathcal{N}^p = \{i_p, j_p\}$, with minimum and maximum time lags as illustrated to the left in Figure 2. Both activities have two modes, one that uses resource a and one that uses resource b. Both activities have duration 2 in both modes, and all resource demands are 1. Mode consistency constraints require both activities to use the same resource. There are also project disjunction constraints for both resources.

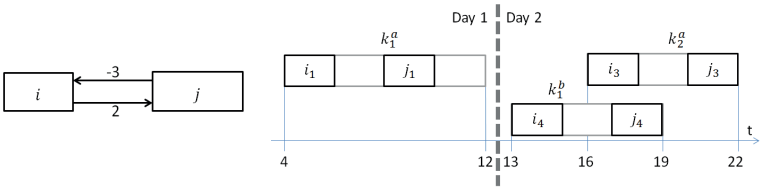

Figure 2: To the left, the project graph structure that is common to all projects in the example. j has to start at least 2, and at most 3, time units after the end of i. To the right, the schedule resulting from the insertion order $\{p_1, p_4, p_3, p_2\}$. For this schedule, $\tilde{\mathcal{P}} = \{p_1, p_3, p_4\}$.

Now, assume that we try to build a schedule by inserting one project at the time, in the order $\{p_1, p_4, p_3, p_2\}$. Assume also that our algorithm tries to schedule each project as early as possible. The resulting schedule is shown in Figure 2. The project p_1 was scheduled first, and as early as possible. The modes using resource a were chosen because this gave the earliest finish time. The same consideration was made for project p_4. Note that p_4 could not be scheduled on resource a in the resource availability interval k_1^a, because that would require i_4 to be scheduled between i_1 and j_1, which would violate the project disjunction constraint on a. Nor could p_4 be scheduled with i_4 after j_1 in k_1^a and with j_4 in k_1^b, because this would violate the mode consistency constraint that require both activities to use

the same resource. Next, project p_3 could not be inserted with i_3 at the end of k_1^a and j_3 in k_2^a, because this would violate the maximum time lag constraint. Both activities were therefore scheduled in k_2^a. There was no room left to schedule the last project, p_2, and so $\tilde{\mathscr{P}} = \{p_1, p_3, p_4\} \subset \mathscr{P}$.

The APSP has some additional characteristics that should also be noted. APSPs can have many projects, often hundreds. These are quite small, with between one and, say, ten or fifteen activities. In any given project, there are usually only a few activities that can be performed in parallel, since most of them involve the patient. Figure 3 shows an example of a project graph for detailed surgery scheduling.

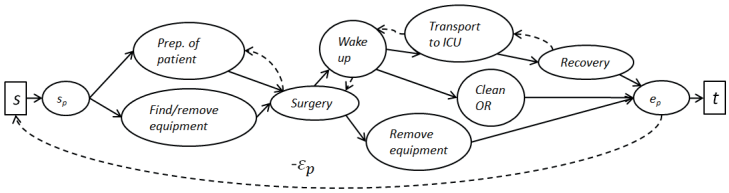

Figure 3: Example project graph for a single surgery patient, with artificial start and end nodes. Solid (dashed) arcs represent minimum (maximum) time lags. Arc lengths are zero, except for the arc (e_p, s), which constraints the project to finish no later than ε_p.

In general, the MRCPSP/Max may have start-to-start minimum and maximum time lags that can depend on the modes of both activities. In practice we assume, as in [40], that start-to-start time lags only depend on the duration of the 'first' (predecessor) activity. The reason is that in these real world problems, minimum and maximum time lag constraints are actually between the completion of the predecessor activity and the start time of the successor activity. When these are converted to start-to-start time lags, the length only depends on the duration (and thus the mode) of the predecessor activity. The time constraints between activities in a project are often very tight, and many activities must be scheduled back to back (see Figure 3). As mentioned in Section 2.1, the resource availability intervals can often be aggregated into disjoint time intervals, giving these problems a distinct multi-interval structure. In practice, there is often one such aggregate interval per working day (as in the example in Figure 2). The APSP can contain

both renewable and non-renewable resources, although we have only encountered renewable resources in the cases we have studied. Finally, while the objective of the MRCMPSP/Max is to minimise makespan, APSPs can define a range of different objectives. Typical objective components are overtime costs, hospitalization costs, intervention costs, room or doctor utilisation, patient's waiting time, and patient or personnel preferences, among others. Some components are non-regular, in the sense that to schedule an activity earlier is not always better. Examples are preferences to schedule certain patients early in the day, or surgeon overtime costs; moving a patient to an earlier day may lead to a worse solution in terms of these objectives.

2.3 Uncertainty

The focus in this thesis is on solving highly complex, but deterministic, real world APSPs. We do not, therefore, handle uncertainty explicitly, even if there are several sources of uncertainty in hospital scheduling applications. There are several reasons for this choice. First, like many other authors, we consider the scheduling of elective patients only, and assume that the treatment of these is protected against that major source of uncertainty that arises from the stochastic arrival of emergency care patients. This assumption holds for many planning and scheduling problems, in which dedicated resources (and time) are reserved for elective care patients.

Second, to handle the uncertainty in the problem, one first has to be able to solve the deterministic version. As noted in [135], previous work on stochastic surgery scheduling often address unrealistically simple problem formulations, apply simplified recourse actions, or consider only short horizons [43, 147]. As the (deterministic) APSP is NP-hard, it may be expected that exact methods like stochastic programming or robust optimisation may not perform well on realistic, larger, problem instances. Also, in our experience, data are not available to provide probability distributions of a sufficient quality, especially for stochastic programming (see Section 3.1). Another method is to create candidate schedules for a deterministic version of the problem, and to evaluate them, for example by use of discrete event simulation, for a finite number of stochastically generated scenarios. As shown in [135], such methods require that good solutions can be found, fast, for the deterministic version of the problem.

Some authors use hedging in the duration estimates to absorb the effect of duration variations. For example, Charnietski [31] introduced an estimate $\mu + h\sigma$, where μ is the mean duration, σ the standard deviation, and h a parameter of the algorithm which was tuned using simulation (see also [150, 69]). Other authors base their duration estimates on a percentile of observed durations [129, 130]. The hedging approach is, in our experience, consistent with how many hospitals do their activity scheduling. They treat activity durations as deterministic, but may add some slack to hedge for unforeseen delays. In long term admission planning, they may also plan with an artificially reduced future capacity for certain resources, in order to reserve capacity for patients that may need treatment urgently, but that are not yet known at the time of planning. In detailed surgery scheduling, with a planning horizon of a few days, the scheduling is often done without any regard to unforeseen arrivals or cancellations; these are simply handled by reactive rescheduling as they occur. As noted in [26], the majority of hospitals that have been previously studied handle disruptions in a similar way. Again, once the hedged duration estimates are made, one has a deterministic problem which needs to be solved efficiently.

Chapter 3

Applications and related work

APSPs appear in many hospital applications, both in surgical and non-surgical treatment, as well as in diagnostic services. This chapter gives an introduction to these applications, and an overview of the existing literature for each of them. The relevant surgery scheduling literature is covered in Section 3.2, while Section 3.3 provides a comprehensive survey of non-surgical applications. We show that while the number of such non-surgical APSP applications exceed those that we have been able to address explicitly in our contributions, their fundamental similarities indicate that our results should have relevance for all of them.

In addition to this application oriented survey, we also present an overview of work that is related to our more generic modelling and solution approaches. Since parts of our contribution use a project scheduling model, Section 3.4 presents an overview of the related project scheduling literature. Finally, Section 3.5 gives a brief introduction to an emerging hardware technology for massively parallel computing, and its applications to search methods.

Throughout this chapter, we focus mainly on the literature that is directly relevant to the APSP as defined in Chapter 2. We also limit ourselves to papers that describe the use of *search methods* (excluding, for example, papers that only use discrete event simulation to evaluate simple dispatching rules). For readers with a limited background knowledge about search methods, we first offer a brief introduction in Section 3.1.

3.1 Search methods

Simply put, an optimisation problem is defined by a set of variables, a set of constraints, and an objective function that is to be optimised. A solution to such a problem consists of a chosen value for each of the variables. A solution is feasible (legal) if these values satisfy all the constraints. A solution is optimal if there exist no other feasible solutions for the problem with a better objective value. An optimisation problem may have more than one objective function (a *multi-objective* problem), in which case all solutions in the Pareto set are considered optimal. See, for example, [41] for an introduction to multi-objective optimisation.

Throughout this thesis, we will refer to a number of commonly used search methods. This section provides some background for understanding these references, as well as pointers to relevant introductory texts. By the term *search method*, we here refer to a method that provides an optimal, or near optimal, solution to an optimisation problem. We will distinguish between *optimisation* (or *exact*) methods and *heuristic methods*. Optimisation methods, if successful, either find a globally optimal solution to the problem or proves that no feasible solution exists. Heuristic methods search for good solutions, but are unable to prove if the best found solution is optimal or not, and are unable to prove infeasibility. Heuristic methods are often used when the problem is too hard to solve with optimisation methods within the required time limit.

One class of exact approaches is based on *linear programming*, where the optimisation problem is expressed as a set of linear constraints, and a linear objective function. Linear programming includes *integer linear programming* (all variables are integer)[1], *mixed integer linear programming* [148] (contains both real and integer variables) , and *binary linear programming* (all variables are binary). See, for example, the text book by Dantzig and Thapa [38] for an introduction. With linear programming, one uses established algorithms, such as Dantzig's Simplex algorithm, to find an optimal solution at a vertex of the polytope that the linear constraints define in variable space. For versions with integer variables, the overall algorithm often applies a branch-and-bound tree search to fix these variables to integer values. Authors that apply such methods generally use commercially available software tools to express the model and to solve it. We use such an

[1] Note that an *integer programming problem* is not in general linear. Usually, however, the term *integer programming* is used in lieu of *integer linear programming*.

approach in Paper IV, and for calculating objective value lower bounds in Paper III.

Stochastic programming [37, 86, 14] and *robust optimisation* [10, 13] are optimisation methods for optimisation under uncertainty. Both are based on linear programming.

Logic based Bender's decomposition was introduced by Hooker [79, 81] as a generalisation of a decomposition approach for mixed integer models that was proposed by Benders in 1962 [11]. This exact approach is sometimes used in scheduling applications where the problem can be naturally decomposed into a master and a slave problem. We propose a three level version of this approach in Paper IV.

Column generation is used to solve large linear programs efficiently. It is often also used in a heuristic setting, in the sense that it can provide good, non-optimal, solutions when the problem is too hard to solve to optimality in a reasonable time. See, for example, [45] for an introduction.

Another exact approach is *constraint programming* [1]. This method uses a tree search that at each node instantiates a variable with one of the values in its remaining domain. Following each instantiation, constraint propagation is used to reduce the domains of other variables. The search backtracks if a variable's domain becomes empty.

Among heuristic methods, we will distinguish here between methods that iteratively improve a single solution, and population based methods that combine known solutions to create new ones. In the first category, we find well-known metaheuristics such as *tabu search* [63], *guided local search* [146], *greedy randomized adaptive search procedure* (*GRASP*) [125], *iterated local search* [96], *variable neighbourhood search* [73], *variable neighbourhood descent* [74], and *simulated annealing* [144]. These are all based on some variation of *local search*. Local search is an iterative improvement method based on an investigation of the local *neighbourhood* of the *current solution*. This neighbourhood is defined by a *neighbourhood operator* (or *move operator*) that represents a certain modification of the current solution. All, or a subset of, these neighbour solutions are evaluated with respect to their objective value, and one of them is chosen as the next current solution. In this way, the local search iteratively traverses a search space topology, or a *neighbourhood graph*, defined by the neighbourhood operators. One

can visualise this search by superimposing this graph on a *fitness surface* defined by the objective value at each node (solution). The local search continues until no improving neighbours exists. This happens when the search reaches a local optima, or a plateau, on the fitness surface. Metaheuristics based on local search implement various tactics to avoid that the search gets stuck in such situations, but instead moves on to investigate new regions of the search space. Several text books provide excellent introductions to local search and metaheuristics [62, 82]. The algorithms in Papers I and II are both based on local search.

Population based metaheuristics updates a population (or pool) of solutions by adding new solutions that are created by combining existing solutions. Examples are evolutionary methods (*genetic algorithms*, *genetic programming*) [51], *scatter search*, and *path relinking* [64]. Some metaheuristics are inspired by swarm intelligence models, such as ant colony optimisation [48] and particle swarm optimisation [89].

For project scheduling, heuristic search methods are often used together with a so called *schedule generation scheme* (SGS). In such methods, the heuristic search modifies a high level solution representation, typically an activity insertion order. The SGS is then used to construct the corresponding schedule by inserting one activity at the time in that order. For an introduction to such methods, see, for example, [107].

Note that despite the above attempt at a loose classification, these methods are frequently combined to create various hybrid methods.

3.2 Surgery scheduling

In this section we first give a brief introduction to the surgery scheduling problem. We then provide some references to the general surgery scheduling literature, followed by a more in-depth discussion of those papers that are relevant for the APSP.

Surgery scheduling problems exist at various time scales and levels of detail. For example, long term admission planning for inpatient elective surgery can span weeks or months. Elective patients are planned based on a waiting list, containing the referrals of each patient. Each referral has been evaluated by the appropriate specialists and contains information about the urgency of the surgery, the required resources and time, and other information that is relevant for the planner. Such ad-

mission planning problems typically consider only one activity for each patient - the surgery itself - and consider only one or two types of resources, such as operating rooms and surgeons [110, 68, 114]. Closer to the day of surgery we find more detailed surgery scheduling problems, including more activities and resources for each patient, but on a shorter time horizon (typically a week). Finally, before the day of surgery, one finds very detailed scheduling problems, that consider all relevant activities and the corresponding resources. A standard definition of these different planning situations is not possible, as the boundaries between them are not always clear, and there are significant differences from hospital to hospital.

Surgery scheduling is done in interaction between the specialities and the surgery department. Typically, the surgical speciality 'owns' the patients and the surgeon resources, while the surgery department owns the operating rooms, teams, anaesthesiologists, equipment, and so on. The capacity of each operating room has often been reserved for different surgical specialities or individual surgeons on different time blocks, in a *master surgery schedule*. This is called *block scheduling*, while the case where all rooms are open to all surgeons is called *open scheduling* [98, 72, 114, 70]. In principle, all critical activities and resources in the surgery department should be included in the surgery scheduling problem, to make sure that the resulting schedule is workable. In practice, however, the surgery scheduling is often simplified. For example, during admission planning, only some of the surgery department's resources are considered, typically the operating rooms. Only the surgery itself is scheduled. The remaining activities and resources are considered only implicitly, based on experience or rules of thumb, or not at all. Similar simplifications are often made in more detailed surgery scheduling. The result is that the schedule may not be workable in practice, and that this is only discovered close to the day of surgery. This leads to unnecessary rescheduling. We believe that the reason for this simplification is, at least partly, due to the lack of efficient surgery scheduling tools that can handle the full problem complexity. When formulating a surgery scheduling problem for a given application, therefore, it is important to handle all the critical aspects of the problem that *should* be included, not only those that are explicitly included in the present manual scheduling.

Given the importance of surgery scheduling to the overall performance of the hospital, it is not surprising that these problems have received a lot of attention

in the operations research community, and that the current literature is large and diverse. Several recent literature surveys are available [83, 26, 42, 104, 67, 54]. From these surveys, it can be seen that a great variety of problem definitions and solution methods have been studied. Despite this diversity, the number of previous publications that are directly relevant to the complex APSP is much more limited, especially for problems with longer planning horizons. As mentioned in Section 2.1, this is probably due to the inherent computational complexity of these problems.

Consider first long term admission planning, which for realistic problem instances typically spans several weeks or even months, and where the schedule may at any time contain hundreds of patients. We address such problems in Papers I and III. We are not aware of any previous work that considers such multi-mode surgery admission problems, where planning and scheduling are integrated.

On shorter time horizons, however, we do find some previous papers that consider integrated planning and scheduling. The following authors all consider a one week scheduling horizon, which is very common in hospitals [145]: Van Huele and Vanhoucke [142] consider integrated planning and scheduling combined with physician rostering. They model this problem using mixed integer programming. They demonstrate the effect of the surgery related constraints on the physician roster, and the effects of physician related constraints on the surgery schedule. In [143], the same authors develop a set of constructive heuristics for solving the same problem. Molina-Pariente et al [106] and Vijayakumar et al [145] both consider multi-mode planning and scheduling on a planning horizon of up to five days. In [145], the problem is modelled as an unequal-sized, multi-bin, multi-dimensional dual bin-packing problem. Both papers [106, 145] first employ a mixed integer linear program which cannot be solved fast enough. They therefore go on to construct specific schedule construction heuristics for this problem. Marques et al [101] consider a version of the problem where the (elective) surgery is the only activity for each patient. The operating room is the only resource to be chosen, but surgeons that are preassigned to each surgery still impose capacity constraints. They also require that only surgeons from the same speciality can use an operating room on the same day. They formulate the problem as an integer program which they solve with a 30 000 seconds time limit. For larger instances, this approach does not provide an optimal solution within this time limit, in which

case they provide a fast heuristic that effectively improves the best found integer solution, when such exists. The resulting schedules are compared favourably with those that were created manually in the hospital. The same authors later tackle the same problem [103], but this time with a genetic algorithm. In [102], they consider a bi-objective version of the problem. The two objectives represent the number of surgeries that are scheduled, and the utilisation of the operating rooms, respectively. They present dedicated construction and improvement heuristics. Doulabi et al [49] present a column generation based method, where constraint programming is used to solve the subproblem. They assume that all available operating rooms are identical on each day, which in a many real world settings may not be realistic. The problem uses an open scheduling strategy, but each surgeon is required to be in at most one operating room during each day. Also, a maximum number of hours of surgery is allowed for each surgeon per day. Surgeons are preassigned to each surgery, which is the only activity that is considered. The authors test their algorithm on randomly generated data and show that the column generation approach is more efficient than a corresponding compact formulation. Bulgarini et al [22] combine scheduling on a one week horizon with a long term assignment of patients to subsequent weeks, using a mixed integer programming approach. They show that this improves over a myopic scheduling for only the first week, which tends to systematically favour the most urgent patients without taking into account medium or long term effects on the plan. Doctors are preassigned to rooms. They do not schedule the patients in subsequent weeks because the computational effort is too large, even if they consider a small surgery department. Similar approaches have been used in manufacturing production planning and scheduling, for the same reasons [100]. The authors do not report how long time they take to solve this problem, but conclude that in order to solve realistically sized instances, it would be necessary to develop a heuristic approach.

There are three papers that take a more generic modelling approach, similar to what we do in Paper III: Roland et al [127, 128] model weekly problems as an extension to the RCPSP. They use genetic algorithms, where a schedule generation scheme is used to construct complete schedules from the chromosome representation of each new solution. Pham and Klinkert [120] view the problem as a *multi-mode blocking job shop problem*. They solve test instances using mixed integer programming, with a one hour time limit. We will discuss these papers in

some detail when we describe the contribution of Paper III, on page 47.

Detailed, one day surgery scheduling problems are often solved as the second ('allocation scheduling') step of the traditional two-step approach [140, 113, 114], as described in Section 2.1. However, it can also be a refinement of an existing schedule (including activity start times), that has previously been created on a longer horizon. As proven by Cardoen et al [24], the one day surgery scheduling problem can also be NP-hard. These problem can be quite detailed. The set of activities may include, for example, the preparation of a patient for surgery, preparation of equipment, removal of unnecessary equipment, the surgery, waking the patient, cleaning the operating room and equipment, transporting the patient to the recovery room or intensive care unit, and the recovery. These activities will be linked by (tight) minimum and maximum time lag constraints. The involved resources can include preparation rooms, operating rooms, operation teams, surgeons, anaesthesiologists, equipment, the recovery room, the intensive care unit, post-operative beds, and others. Most authors include only small subsets of these activities and resources.

For a complete overview of earlier papers on one day surgery scheduling problems, we refer to the aforementioned survey papers. The most up to date of these, and perhaps the most comprehensive, is that of Demeulemeester et al [42], which is from 2013. In what follows, we discuss some central papers from this literature. We also provide an overview of more recent publications. Cardoen et al [24] address a one day scheduling problem for a surgical outpatient clinic. The problem arises on the day before surgery. Prior to this, the patients have only been told on which day to show up, based on the earlier advance planning step. Surgeons are preassigned to operating rooms. The authors propose various algorithms based on integer programming and a dedicated branch and bound, and show that the most competitive is an *iterated integer programming* approach, where iteratively, a substantial number of variables are fixed before the (sub)problem is solved again. In [25], the same authors solve the same problem using a *branch and price* approach where they use dynamic programming to solve the pricing problem. They consider several objective components, but optimise a single linear combination of these. They show that the dynamic programming solver performs better than a corresponding mixed integer approach for the subproblem. Jebali et al [85] use a mixed integer formulation to demonstrate that integrating planning (allocation of

patients to rooms) and scheduling is important, even for a one day problem. They show that this gives better results than dividing planning and scheduling into two consecutive steps. This improvement, however, comes at a high cost in terms of CPU time. Augusto et al [3] propose a heuristic based on Lagrangian relaxation of their integer programming formulation, combined with dynamic programming. They consider several activities, including the recovery. For the surgery itself, only operating room resources are considered, and these are assumed to be identical. The authors show that there is a benefit, in terms of makespan, to let a patient recover in the surgery room whenever there is no free capacity in the recovery room. Ghazalbash et al [61] consider a one-day multi-mode problem with an open scheduling strategy, where rooms, equipment, surgeons, and surgeons in training have to be chosen. The surgery is the only activity. They use a mixed integer approach, and compare the results favourably with the current practice in the hospital. Zhao and Li [152] consider a one day scheduling problem where the surgery is the only activity, and the operating room is the only resource that is chosen. They model the cleaning and preparation activities between surgeries as a sequence-dependent setup time for these rooms. They solve quite large daily problems (28 ORs, 60-80 patients) in between 4-9 minutes, using constraint programming. Analysing the robustness of their deterministic solutions under a stochastic variation in the surgery durations, they conclude that the deterministic model is sufficient for solving this scheduling problem. All patients are elective. Xiang et al [149] present an ant colony algorithm for a problem that considers both pre-, peri-, and postsurgical phases, and that integrates skill and availability constraints for nurses, based on the nurse roster. They consider elective patients only, and base their model on the job shop scheduling problem. Pulido et al [123] treat uncertain activity durations using stochastic programming. They also present two decomposition based heuristics, in which, iteratively, some or most of the variables in the stochastic problem are fixed before solving. They show that these produce good results, and that they are computationally more tractable than the stochastic program. Their largest test instances have 11 surgeries. Meskens et al [105] use constraint programming to handle several types of real world constraints, including team preferences. Wang et al [147] consider the allocation of patients to rooms under uncertain durations, and use a column generation approach to minimise overtime and the risk of cancellations. A similar

problem is tackled by Deng et al [43], whose integer programming approach uses 'chance constraints' to minimize the total cost of opening operating rooms, subject to restrictions on the probability of surgery delays and overtime. Saremi et al [135] present a hybrid between integer programming and tabu search for solving a stochastic surgical outpatient scheduling problem. The problem contains preparation, surgery, and recovery activities. They use the integer programming approach to provide a good solution to the deterministic problem, and then use this as the initial solution for their tabu search. The tabu search uses simulation over 30 randomly chosen scenarios to evaluate each candidate solution under uncertainty. They show that the high quality initial deterministic solution is essential for this approach to be competitive. For performance reasons, especially for larger problem instances, they also propose to replace the integer programming approach with a relaxed mixed binary program, combined with a repair heuristic to fix non-integer variables before the tabu search phase.

3.3 Scheduling of non-surgical activities

Most hospitals have laboratories and clinics that perform non-surgical activities, including treatment, diagnostic procedures, or control procedures. Some of these have planning and scheduling problems that fall under our definition of the APSP in Chapter 2. The literature for such applications is much more limited than the surgery scheduling literature. Also, we have found no survey of this literature, and so we provide a more comprehensive overview here.

Let us first note that the substantial literature on *appointment scheduling* is not directly relevant here. 'Appointment scheduling' is usually taken to mean the design of template schedules, into which patients are typically booked on a first come, first serve basis [83]. Surveys of the appointment scheduling literature can be found for example in Cayirli and Veral [29], and in Gupta and Denton [70]. From these, one can see that the appointment scheduling literature covers topics such as the optimal choice of block start times, the number of patients to schedule in each block, the estimation of service durations, the choice of scheduling rules to apply, and so on. These problems are usually modelled as *single server* problems, considering a single resource (typically a doctor), and a single activity for each patient [8]. Much of this work uses queuing theory and discrete event simulation, while some also apply search methods [53]. Since such appointment scheduling

problems do not consider the planning and scheduling of actual, known patients, they do not fall within the scope of this thesis.

Non-surgical APSPs are primarily found in cancer treatment scheduling, rehabilitation treatment scheduling, and in the scheduling for various diagnostic services. We shall see that these problems are very similar. Therefore, although our contributions do not explicitly address all of them, our results should be widely applicable.

The largest literature concerns the scheduling of cancer treatment programs, including radiotherapy or chemotherapy, where each patient's program contains many treatments that are to be carried out over several days or weeks. There are also some preparatory activities, but these are usually not considered in the literature. As the schedule typically covers several weeks, it is always partially filled with patients that have already been scheduled. New treatment requests are either scheduled continuously, in an online fashion, or in batches once or twice a day. The main objective of these problems is often to minimize the degree of deviation from each patient's prescribed treatment program. An efficient solution of these problems are of great medical value, since such deviations have a direct impact on the health and survival prospects of the patient [34]. Other objective components include the number of scheduled patients, the degree of compliance with official waiting time targets, resource utilisation, and resource overtime. Few hospitals use advanced scheduling technology, and those papers that consider real world applications invariably report big improvements over current practice.

3.3.1 Radiotherapy treatment scheduling

Let us consider first the radiotherapy treatment scheduling problem, in which the problem is to schedule treatment activities on one or more linear accelerator machines (*linacs*). These are often modelled as job shop problems; for example, Kapamara et al [87] define the problem as a dynamic and stochastic job shop problem. They review various methods that have been successfully applied to such problems, including branch and bound approaches, simulated annealing, tabu search, GRASP, and genetic algorithms. They conclude that the problem might be best tackled by a metaheuristic, built around tabu search or a genetic algorithm. Petrovic et al [119] propose two construction algorithms that attempt to schedule each patient (in prioritised order) as soon as possible after release date, or as late as possible before the due date, respectively. They show that the latter algorithm

has some advantages because it gives better results for palliative (pain relief) patients. Only treatment activities are considered, and these are assigned to days. The authors do not include scheduling decisions. The problem is considered deterministic, and patients that have already been planned are not reassigned. In [118], Petrovic et al generalise these two construction algorithms. They also propose a GRASP method which performs slightly better for cases with the average number of patients or less. However, the two construction algorithms perform better when the patient load is high. Conforti et al [34] present the first exact method for this problem. New patients with different priorities are scheduled on a one week rolling planning horizon. The authors present two similar integer programming models: one which simply insert new patients, and one which also reschedules existing patients (without removing any of these from the schedule). Patients that cannot be scheduled on the first week, are kept on the waiting list and will be a part of the problem for the next week. The authors argue that their single server model can be easily extended to include several machines, which would make their problem multi-modal. This would not, however, introduce any of the complications that arise in typical APSP instances, where modes include several resources of different types; it would simply correspond to increasing the capacity of the single linac. They apply their models to a small real case and show significant improvements in waiting times and the number of new scheduled patients per week, compared to the current practice in the hospital. In [36], they extend their model to include patient availability. In [35], they consider what they call a 'non-block' setting, where patients are not allocated to slots, but have individual treatment durations. Here, however, they do not choose start time, but a *shift* for each patient. There are two shifts per day in their case study. Again, their problem is deterministic and the planning horizon is one week. Their exact approach produces optimal plans for a collection of test instances, within a run time of 12 minutes. The largest instance has 20 patients.

A few authors also consider stochastic aspects of the radiotherapy treatment scheduling problem. Legrain et al [94] present an online algorithm where both a linac and a time slot is chosen for each treatment of each patient. They plan on a long horizon, but schedule only one patient at the time, without rescheduling the previously scheduled patients. They use stochastic programming to handle uncertainty in treatment durations and the arrival of future urgent patients. Their

experiments verify that this approach is superior to the greedy scheduling heuristic that the hospital in their case study uses today. Pérez et al [116] consider scheduling of patients for nuclear medicine treatment on a one day scheduling horizon. Nuclear medicine is a sub-speciality of radiology that involves several activities related to each procedure, where each activity requires different, and multiple, resources. These activities must be carried out in a given sequence. The problem is multi-modal. Furthermore, it is stochastic and online, since patients arrive on a short notice and are scheduled immediately, one by one. They present two integer programs for this online problem. The first represents the deterministic problem, while the second is a two-stage stochastic integer program. They use discrete event simulation to evaluate both methods, and demonstrate that the stochastic program gives better results, especially for high load scenarios. It also performs better than the scheduling algorithm that is currently used in the clinic, for a range of different evaluation criteria: the number of treated patients, a patient preference satisfaction ratio, and patient waiting time. Cares et al [27] address the diversity in formulations of the radiotherapy treatment scheduling problem, and present a standardised online benchmark generator [28].

Petrovic and Elkin [117] introduce a genetic algorithm for the radiotherapy pre-treatment scheduling problem, where patients are scheduled for an initial assessment before starting their radiotherapy treatment. The problem contains several activities per patient, and is multi-modal. It seems that the same doctor is used several times during the treatment of the same patient. It is not clear if it is a constraint that only one doctor is involved. If so, however, this is a practical example of the general mode compatibility constraints that we introduced in Paper III.

3.3.2 Chemotherapy treatment scheduling

The chemotherapy treatment scheduling problem is very similar to the radiotherapy treatment scheduling problem. However, it typically has more variation in activity durations and resource demand, and activities can use multi-capacity resources [71]. A chemotherapy treatment plan is often given in treatment periods separated by rest periods. The exact structure of each treatment plan is decided by a team of oncologists, and depends on the kind of cancer that is being treated.

Sadki et al [132] consider the planning of chemotherapy patients on a group of oncologists, while at the same time determining their working schedules. The

time resolution for both decisions are based on a shift, and there are two such shifts per day. The problem is therefore a planning problem, without detailed scheduling decisions. The authors formulate a mixed integer program for the problem. As this cannot be solved in a realistic time, however, they also present a heuristic algorithm, using a hybrid of local search and mixed integer programming for relaxed problem formulations. They claim that this approach gives considerably better results than current practice, but admits that they use information about all patients for the whole scheduling horizon, while in reality this information becomes known only gradually. They go on to address this issue in [133], where the problem is solved on a one week rolling horizon, and where future patients are anticipated based on an observed probability distribution. They validate this approach through simulation, and show that they achieve a more balanced bed load than the current practice in the clinic under study. The patients are preassigned to an oncologist, and oncologists are preassigned to shifts. Turkcan et al [141] do planning and scheduling in two sequential steps. In the planning step, a first date of treatment is assigned to each new patient, on a one week rolling horizon. The objective is to minimize unnecessary treatment delays. The scheduling problem decomposes into one problem per day, in which they assign start times and resources (chairs and nurses) to each patient. Here, the objective is to minimize overtime. They solve both problems using a mixed integer approach, and show that this can be solved fast enough to be useful in practice. The PhD thesis of Hahn-Goldberg [71] addresses the scheduling of chemotherapy outpatients on a single day. She presents a scheduling method called 'dynamic template scheduling', based on constraint programming. She uses historical data to predict future arrivals of treatment requests, and solves the resulting deterministic problem. Last moment changes are handled using a dedicated, reactive, shifting algorithm.

Note that the RCPSP based model for generalised surgery scheduling problems that we present in Paper III should be able to express radiotherapy and chemotherapy scheduling problems. The project and activity concepts, and the multi-period resource availability structure should fit the problems well. The special constraints concerning program conformity can be modelled by a combination of minimum and maximum time lag constraints. However, the relevant objectives would have to be added where such constraints are to be relaxed into soft preferences. Also, the results of Paper IV should be very relevant for these applications.

The three-level decomposition approach described therein should be applicable, with the proper adjustment to the master problem where activities are assigned to days. Our research should therefore be of relevance also to radiotherapy and chemotherapy scheduling, even if we have not addressed these problems specifically in our contribution. We also observe that none of the previous work discussed above integrate planning and scheduling on a scheduling horizon of more than one day, for a multi-mode problem, while scheduling more than one patient at the time. Our results should therefore be applicable to richer, more complex, versions of these problems than those previously studied.

3.3.3 Rehabilitation treatment scheduling

Yet another APSP application is the scheduling of various therapeutic activities, for example in connection with rehabilitation. While not as medically critical as the cancer treatments discussed above, an effective scheduling of such activities contributes significantly to patient satisfaction and hospital economy. The problem consists of scheduling a set of (possibly different) treatment activities for each patient over a period of time. There may be minimum (and possibly maximum) time lags between some of them. These therapeutic treatment activities tend to be personnel-intensive. Activity durations are as a rule deterministic. Patients often prefer to use the same therapist across sessions.

Chien et al [32] solve such a problem using a genetic algorithm, combined with a schedule generation method that decodes the chromosomes. They base their work on a *hybrid shop scheduling problem*. Their version of the problem is multi-modal, and they choose start times for each activity. All machines (resources) have capacity one. In addition to some minimum time lag constraints between pairs of activities, there is a maximum time span constraint for the completed treatment of each patient. They validate their algorithm by comparing results for small problem instances with the results from a corresponding mixed integer program. This is a problem formulation that fits our APSP description perfectly. Another paper that integrates planning and scheduling is by Braaksma et al [18]. However, here they use an online approach, where each new request is scheduled immediately, using an integer programming approach. There is no rescheduling of previously scheduled patients. The authors aim to optimise various patient and treatment related preferences, as well as resource utilisation. They use discrete event simulation to test their approach for a small but real hospital clinic.

Ogulata et al [111] plan physiotherapy treatment on a one week rolling horizon. Three problems are solved sequentially: the choice of patients to include for the week, the assignment of patients to physiotherapists, and an assignment of patients to two-hour periods on a day during the week. The last step is scheduling-like, but they do not include any resource capacity constraints, and so can produce plans that overload the physiotherapists. The point of the last step is only to balance the load between periods. The introduction of these periods is in itself an improvement over current practice, since they drastically reduce patient waiting time on the day (in the current practice, all patients show up in the morning, draw a number, and wait for their turn). Griffiths et al [65] present a planning tool. They use a three stage local search based approach to schedule patients for physiotherapy in a rehabilitation clinic. The objective is to minimize the violation of constraints, and each sequential stage considers a separate group of constraints in decreasing order of importance. A combination of simulated annealing and tabu search is used on the third stage. Schimmelpfeng et al [137] consider a problem of scheduling all treatment activities in a rehabilitation hospital on a horizon of several weeks (one month in their tests). They present a mixed integer model that defines the problem, where resources, a day, and a time slot are assigned to each patient. Assignments of groups for group therapy are also made. This monolithic model cannot be solved, however, for medium or large size instances. They therefore propose a hierarchy of three models that are solved sequentially: In model 1, patients are assigned to days. In model 2, they are given a time slot, while in model 3 they are assigned to resources (and, possibly, to therapy groups). Obviously, this approach reduces the complexity of the problem. However, it has the same weaknesses as we discussed in Section 2.1 related to the separation of planning and scheduling decisions. Using small test instances, they quantify this loss to be about 5-10% in the number of scheduled patients, compared to solutions for the monolithic model.

Again, many of these problems, and in particular those without special group therapy preferences, bear strong resemblance with our generic APSP model. This indicates that our results are of relevance also for rehabilitation scheduling problems.

3.3.4 Diagnostic services

In addition to giving treatment, hospitals also have various clinics that provide diagnostic services. Some of these have APSP-like scheduling problems that can involve a large number of elective patients. These problems can have multiple activities per patient. They can also be multi-modal, where each mode involves several resources, some of which may have setup times. Activity durations may depend on the examination procedure, the age and medical condition of the patient, and often also on the chosen mode. Patients are inserted, often from a waiting list, into a schedule that may span weeks or months, and which already contains appointments that have been fixed and communicated to the respective patients. Typical objectives are to schedule as many patients as possible within the planning period, maximise resource utilisation, and to prioritise the patients in time according to their respective medical urgency. As these problems often involve outpatients, start times have to be chosen and communicated to the patients several weeks before the appointment date. This is different from the typical admission planning for inpatients. However, since we in the APSP integrate planning and scheduling also for inpatient admission planning (even if only the date is communicated to the patient), the underlying optimisation problems that we consider for the two applications are very similar.

The only previous paper that we have found that explicitly addresses a diagnostic clinic is that of Serrano et al [139], in which the flow of patients through a gastroenterological clinic is studied on a one day horizon. They model the problem as a flexible job shop problem with three operations (activities): bed assignment, examination, and recovery. Their approach makes use of several different dispatching rules, which they show improves patient waiting time, length of stay, and resource utilisation, compared to the current practice in the clinic. There are other papers concerning diagnostic services in the appointment scheduling literature (for example [115]), but as explained above, these are not directly relevant to the APSP. The lack of relevant literature about APSPs in diagnostic services is a bit surprising, since we do find such problems in hospitals, at least in Norway. The reason may be the same as for admission planning for inpatients: that the combined planning and scheduling for many patients on a long time horizon is computationally very demanding. Yet, in Paper IV, we present an exact method that solves such a typical multi-mode outpatient appointment planning and scheduling

problem to optimality in a very short time.

3.3.5 General approaches

There are also a few studies that are more general in terms of application area. Podgorelec and Kokol [121] present a genetic algorithm for a general activity scheduling problem that is relevant to various applications, such as examinations, laboratory tests or rehabilitation therapies. They choose the start time for each activity. They consider scheduling both on equipment and personnel, and also choose which resources to use for each activity (i.e. the problem is multi-modal). They test their algorithm on one small synthetic instance, without offering any insight into the quality of the results. They use machine learning to tune the parameters of the algorithm.

Also, Gartner and Kolisch [60] aim to optimise the flow of patients in the entire hospital, based on definitions of all clinical pathways. They include planning decisions, but not scheduling.

3.4 Project scheduling

As mentioned in Section 2.2, parts of the work in this thesis use a model that extends the classical resource constrained project scheduling problem (RCPSP). In this section, we therefore provide an overview of the most relevant related work in project scheduling. The literature for the RCPSP is substantial, and good introductions to the problem and relevant solution methods can be found in, for example, Neumann et al [107] or Artigues et al [2]. See also Kolisch and Hartmann [91, 92] for an overview of scheduling heuristics and heuristic search methods. Lombardi and Milano [95] provide a survey of optimisation methods for multi-mode RCPSP's, where they highlight the challenges in combining resource allocation and scheduling.

In [76], Hartmann gives an overview of papers that have addressed different extensions to the RCPSP. The most relevant earlier work concern the single-project MRCPSP/Max (we have not found any previous papers for the multi-project version). This is sometimes also called the multi-mode RCPSP with generalised precedence relations (MRCPSP-GPR). As a generalisation of the RCPSP, the problem is NP-hard. Since the feasibility of a schedule can be verified in polynomial time, the problem of deciding whether a MRCPSP/Max instance is solvable is NP-complete [78]. Some authors use a *sequential* (or *decomposition*)

approach [107], where the problem is decomposed into a mode assignment problem (MAP) and a RCPSP/Max subproblem. First the MAP is solved to give a feasible mode assignment, and then each such assignment is evaluated by solving the resulting RCPSP/Max. Both of these subproblems are in general NP-hard [40]. The MAP is NP-complete when the number of non-renewable resources exceeds one [90]. Note, however, that for most APSP instances all resources are renewable. The feasibility problem for the RCPSP/Max is NP-complete [7, 107].

The literature for the MRCPSP/Max is very limited, and we provide an overview in the following. De Reyck et al [40] claim to be the first to address this problem, in 1999. They use the sequential approach, presenting a tabu search method for the MAP, and using a truncated branch-and-bound method to solve the RCPSP/Max subproblem. They compare this with several other approaches based on enumeration or local search, and demonstrate the tabu search's superior performance on a set of artificially generated test instances. They analyse the performance also as a function of test instance characteristics, and show that instances with less availability for renewable resources are more difficult to solve. Also, instances with a medium level of availability for non-renewable resources prove the most difficult. Heilmann [77] proposes a multi-pass construction algorithm. In contrast to [40], Heilmann uses the *integration* approach, where a mode is chosen for one activity at the time, as an integral part of the schedule generation. For each pass of the algorithm, the order in which activities are inserted, and the choice of mode for each activity, are both chosen by roulette selection. These samplings are based on probabilities given by activity-priority rules and mode-priority rules, respectively, and the combination of such rules vary from one pass of the algorithm to the next. As in [58], Heilmann's schedule construction algorithm performs backplanning steps to handle maximum time lag violations. He also employs a series of feasibility tests after each activity insertion, in order to interrupt the construction of an infeasible schedule as early as possible. Heilmann tests his method on randomly generated instances and shows that the combination of many different priority-rules contributes to its performance. He also shows improved performance compared to the tabu search of De Reyck et al [40], but only for shorter computation time limits. Heilmann's algorithm produces feasible solutions for all test instances, which is not the case for De Reyck et al's tabu search. These results are not surprising. We too have observed that the sequential

approach (which is used in [40]) is not well suited for heuristic schedule construction. The reason is that for a given MAP solution, the chances of finding a feasible solution to the corresponding RCPSP/Max subproblem is quite small. Hence, a lot of computational effort is wasted. The integration approach has a much better chance of producing feasible schedules. Similar observations are made in [107]. It is interesting to note, however, that the above decomposition can work well with exact methods, for example when using logic based Benders' decomposition (see Paper IV). In this case, the performance is highly dependent on the strength of the relaxed scheduling constraints that are used in the MAP master problem, as well as the cuts that are added when the RCPSP/Max slave problem is proven infeasible. One could imagine that if similar mechanisms are used with care, they could also strengthen a heuristic sequential approach. In [78], Heilmann introduces the first exact method for the MRCPSP/Max, based on a branch-and-bound procedure. The search maintains a *fictitious schedule*, which is a solution to a *minimal problem*, where activity durations, resource demands, and minimum and maximum time lags are calculated as a minimum across all activity modes that are still available for each activity. Branching rules are chosen dynamically; at each node in the search tree, a choice is made whether to choose a mode for an activity, or to resolve a resource conflict in the current fictitious schedule. The branching rule to use next depends on the criticality of each decision, based on the maximum makespan lower bound across all branching alternatives. Computational results demonstrate that this approach is superior to the tabu search in [40], both in terms of the number of problems that are solved to feasibility, and in terms of gaps to lower bounds. The largest test instance has 50 activities and 5 modes for each activity. Sabzehparvar and Seyed-Hosseini [131] present a geometrical formulation of the problem, inspired by rectangle packing models. This formulation is based on the requirements that multidimensional volumes that are spanned by the activity duration and the demand of different resources, do not overlap. The approach uses a continuous representation of time, which is advantageous for problems with a high time resolution. On the down side, the approach performs best when the demand is distributed uniformly in time, and where there are not too many resources in each mode. They show that their formulation is more efficient than that of [40] when there are less than tree resources in the problem. Barrios et al [6] present an approach in which they first solve a simpler problem: a mode

assignment problem where the objective function is some easily calculable approximation of the makespan for the corresponding MRCPSP/Max problem. This simpler problem is solved using a genetic algorithm. The resulting solutions are then converted and used as initial solutions for the MRCPSP/Max problem, which is again solved by a genetic algorithm, using the integration approach. They integrate different useful mechanisms into their method, such as repair heuristics and suitable schedule justification methods. Barrios et al show that the resulting overall method is competitive with the state-of-the-art algorithms for medium and large size problem instances, and that it performs better than the tabu search in [40]. In their discussion of future research opportunities, they mention that the mode assignment could be combined with a solving of the RCPSP/Max, in a decomposition based method. In [5], the same authors follow a very similar approach to the one in [6], but in this case the simplified assignment problem is solved using a simulated annealing algorithm. They also propose a method for improving the solutions of this simplified problem by attempting to modify the choice of mode for each activity in turn. They show that this gives an improvement for the overall algorithm. At the second level, like in [6], they solve the MRCPSP/Max using a genetic algorithm, but this time a different one. Like in [6], Bagherinejad and Majd [4] also use a double genetic algorithm. They, however, first solve the approximate mode assignment problem, and then enter a second phase where they only solve the RCPSP/Max resulting from using the best assignment solution. They do not seem to reconsider the mode choices at the second level. Calhoun et al [23] present a tabu search for a goal programming formulation of the MRCPSP/Max. Their move operators work directly on the mode assignment variables. The evaluation of each move involves the choice of a new start time for the activity in question. When the move leads to a violation of time constraints, a penalty is added to the objective value. If such a move is performed, then only moves that may resolve this violation are allowed until the schedule is once again feasible. While interesting, this approach would probably not work well for most APSPs, because of the combination of tight time lag constraints, mode consistency constraints and project disjunction constraints. For the same reasons, local search methods that operate directly on a full schedule representation, like the one proposed by Dauzère-Pérès et al [39] for the multi-resource job shop scheduling problem, would also be unsuitable for the APSP.

Many algorithms for the simpler RCPSP/Max [58] and the MRCPSP [97] combine the use of a schedule generation scheme (SGS) with some higher level logic that determines the order in which activities are inserted. In an integration setting, the mode for each activity is chosen as a part of the schedule generation. For problems with maximum time lag constraints, the SGS includes *unscheduling* or *backplanning* steps [58]. For the MRCPSP/Max, only the priority based sampling method of Heilmann [77] uses an SGS. This is the approach that we take in Paper III, although our algorithm also considers the insertion order among projects, and our SGS is modified to handle all the constraints of the APSP.

3.5 Parallelisation

The success of any search method depends partly on its software implementation, and on how efficiently this implementation uses the available hardware. Some of the work in this thesis exploit traditional task-parallelism on multi-core CPU's. However, during the last few years, various hardware platforms for massively parallel 'same program multiple data' (SPMD) computations have also become more easily available for scientific computing. These include Field Programmable Gate Arrays (FPGAs), multiple-core CPUs, and General Purpose Graphical Processing Units (GPGPUs) [19]. The GPGPU lends itself as an easily accessible computation platform, having over the last few years evolved from a highly specialised graphics processor unit (GPU) to become an affordable general purpose computation device that can be installed in most computers. This development has been supported by a parallel development of higher level programming models, languages, and tools [112, 108, 66]. The GPGPU's computational efficiency, in terms of floating point operations per time unit, is today significantly higher than that of top end CPUs, and the trend is that this gap is increasing [151].

The GPGPU has been successfully applied to increase computational efficiency in a range of different areas, such as sorting, searching, graph algorithms, string matching [136], numerical solving of differential equations, computer vision, and numerical linear algebra [112]. In Paper II, we addressed the question of how this technology could be best exploited in local search algorithms for permutation-like problems. At the time, very little work had been reported about the use of GPGPUs for search methods in general, and in particular for local search based algorithms. Previous use of this technology was mostly for evolu-

tionary algorithms, which lend themselves very naturally to SPMD parallelisation. In particular, parallel fine-grained (cellular) genetic algorithms have been adapted to the GPU architecture over a number of years [151, 153]. Some authors have also used the GPU for genetic programming [75, 93]. An excellent recent introduction to the use of GPGPUs for local search can be found in [20]. The survey in [138] also gives a good overview of existing GPGPU implementations for combinatorial optimisation.

Chapter 4

Contribution

In this chapter, we summarise the contributions of each paper. We also discuss possible extensions and future research opportunities.

PAPER I: LOCAL SEARCH FOR THE SURGERY ADMISSION PLANNING PROBLEM

A. Riise and E.K. Burke. *In Journal of Heuristics, Springer US, 17(4) (2011), pp. 389-414*

In this paper, we consider an admission planning and scheduling problem, where an operating room, a day, and a start time have to be chosen for each patient. Doctors are preassigned to each patient, but have associated capacity constraints: each surgeon can only participate in one surgery at a time. We present a metaheuristic approach based on a combination of iterated local search and variable neighbourhood descent. Two neighbourhood operators, *relocate* and *two-exchange*, are used. A high level solution representation is defined by a sequence of surgeries for each room/day combination. The relocate operator moves a surgery from its current position in such a sequence to another position, possibly in the sequence of another room and/or day. Correspondingly, the two-exchange operator swaps the positions of two surgeries. Following such an operation, and before the neighbour solution can be evaluated, a heuristic is used to create a schedule that is feasible with respect to surgeon capacities.

The paper presents computational results for a realistic set of test instances. These were generated based on the characteristics of the admission planning problem in Bærum hospital, a medium sized Norwegian hospital. In addition, the paper provides a search space analysis for each of the two neighbourhood operators, and for three different objective functions: *surgeon overtime*, *patient waiting time*, and a linear combination of the two. Fitness landscape ruggedness and a fitness-distance correlation between local optima are analysed (see [82] for an introduction to these concepts). The paper shows that although all three fitness surfaces are comparatively smooth for both operators, with a high one-step fitness correlation value, they have quite different characteristics and pose different challenges for local search algorithms. The waiting time objective seems to present a fitness surface that is quite suitable for guiding the local search algorithm. The overtime fitness surface, on the other hand, contains distinct levels of similar objective values, and it is difficult for our local search algorithm to move from one level to another. This indicates that more powerful move operators or diversific-

ation mechanisms could be necessary for problem instances where the overtime objective is present.

As we discussed in Chapter 2, planning and scheduling should be integrated for long term admission planning, to ensure that the admission plan can later be refined into a feasible and high quality surgery schedule. One of the main contributions of this paper is that we present, and analyse, a search method for such a joint admission planning and scheduling problem. We are not aware of any other paper that tackles a similar problem on more than a one week planning horizon. The presented local search based method could be generalised to work on the generic APSP model presented in Paper III. Such an iterative improvement algorithm could use the SGS developed in Paper III instead of the tailored schedule construction heuristic used in Paper I. The search space analysis methodology used in Paper I would be useful in the design of such an algorithm. Although tight time lag constraints would make such an algorithm unsuitable for many APSPs, it could work well for problems without such constraints.

PAPER II: ON PARALLEL LOCAL SEARCH FOR PERMUTATIONS

A. Riise and E.K. Burke. *In Journal of the Operational Research Society, Palgrave Macmillan, 65(5) (2014)*

Motivated by the arrival of easily available hardware for massively parallel computing, such as the GPGPU, Paper II investigates some ways in which this technology can be exploited in methods based on local search.

One obvious benefit is the significant speed-up that can be gained by evaluating a large number of neighbourhood solutions in parallel. Many papers have now been published to this effect. On the other hand, this is not necessarily an improvement over a local search that employs intelligent neighbourhood filtering techniques. However, we show that the ability to evaluate large neighbourhoods very efficiently enables another mechanism for performance improvement. At each iteration we can select and apply many improving moves, simultaneously. The idea is that once all neighbours are evaluated, and several are found to be improving, it would be a waste of computation effort to apply only one of them. Instead we select a set of improving moves and apply them all. The requirement is that these moves are independent, in the sense that the improvement that is gained from each move is independent of the application of any of the others. To select an optimal set of such independent moves amounts to solving the weighted version of the maximum stable set problem, which is known to be NP-hard on general graphs [88]. Since this choice is made at every iteration, efficiency is critical. We therefore select these moves using a simple heuristic, rather than solving the selection problem to optimality.

We ran a series of experiments on various Travelling Salesman Problem (TSP) instances to evaluate the benefits of applying several improving moves in parallel (*parallel moves*) instead of applying only the best move at each iteration (*sequential moves*). Since both algorithms are stochastic, we performed many runs for each test instance. Figure 1 shows a *qualified run time distribution* (QRTD) [82] for the problem instance 'd657'. This is the fraction of all runs that reached a certain solution quality (in this case within 0.5% of the optimal value), as a function of computation time. The figure shows that the algorithm using parallel

moves is considerably faster in reaching this objective value. Combined, the parallel neighbourhood evaluation and the simultaneous application of independent improving moves give speed-ups of up to several hundred times compared to a classical, sequential, best improvement search.

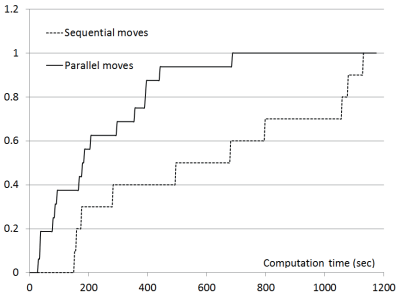

Figure 1: QRTD for the algorithms with 'parallel' and 'sequential' moves, for the problem instance d657, at a 0.5% gap from the optimum.

The above neighbourhood exploration and move applications are embedded in an iterated local search, which is performed on the CPU. We demonstrate how the use of an adaptive diversification strength contributes to the search efficiency of the algorithm. We also show how a targeted exchange of good partial solutions between the current and best found solutions improves the efficiency of the iterated local search. The idea is that even if the current solution is not better than the best solution, it may contain good partial solutions (sub-tours in the TSP) that could improve the best solution further. Equally, the best solution can contain good partial solutions that when inserted into the current solution makes that the new best solution. These are both very general mechanisms, and their effects on search efficiency are independent of the implementation of the low level local search.

Since the time of writing Paper II[1], many papers have reported the use of this technology for local search methods. Many authors simply implement a parallel neighbourhood exploration on the GPGPU, and demonstrate a speed-up compared to a sequential, CPU based, implementation. In Paper II we go further, and explore how this parallel neighbourhood exploration can enable a parallel application of multiple improving moves. This, and the other efficiency enhancement techniques described above, are main results of the paper. The original aim of this research was to exploit GPGPU based SPMD parallelisation to solve the admission planning problem more efficiently. We presented preliminary results for

[1]Note that the paper was originally submitted on on 21^{st} February 2012, and not on 24^{th} January 2013, as it says on the published paper.

this in the Meta 2010 conference. However, as we started to design a general model for the APSP (see Sections 2.1 and 2.2), we came to the conclusion that the most promising algorithms have a complexity that makes them unsuitable for SPMD parallelisation. Therefore, as we presented this general APSP model in Paper III, we proposed instead an iterative improvement method that uses CPU based task parallelism. However, as new types of processing hardware becomes available (such as the recent Intel Xeon Phi Coprosessor), their potential should be explored. It is likely that this line of research will contribute significantly to the efficiency of future planning and scheduling methods.

PAPER III: MODELLING AND SOLVING GENERALISED OPERATIONAL SURGERY SCHEDULING PROBLEMS

A. Riise, C. Mannino, and E.K. Burke. *In Computers & Operations Research, Available online 29 July 2015*

In this paper, we present a general model for a generalised surgery scheduling problem, based on a rich generalisation of the resource constrained project scheduling problem (RCPSP). The model is multi-modal, multi-project, and includes time windows as well as minimum and maximum time lag constraints. The model also contains some extensions to the RCPSP that we believe have not been previously reported in the literature, such as *project disjunction* constraints and *mode compatibility* constraints between activities (see Section 2.2). Originally, a mixed integer approach was explored [126]. This was, not unexpectedly, unable to provide good solutions for problem instances of realistic size, even within quite generous time limits. The paper therefore presents an algorithm based on the use of a schedule generation scheme (SGS; see Section 3.4). We use an adaptation of the SGS with *backplanning* that has been previously used for the RCPSP/Max [58], and for the MRCPSP/Max [77]. The SGS is modified to tackle all APSP constraints, including project disjunctions and mode consistency constraints. It uses the integration approach, choosing modes as an integrated part of the schedule generation. The high order solution representation consists of a project insertion order; the order of insertion of each project's activities are chosen as a part of the SGS.

Initial solutions are generated by a repeated sampling of project insertion orders. For each insertion order, the SGS is used to construct the corresponding schedule, and the insertion order and the schedule are stored together in a pool. In parallel, a dedicated improvement method repeatedly selects a promising solution from the pool, and modifies the corresponding project insertion order. Simply put, those projects that contribute the most to the objective value are moved to an earlier position in the insertion order. The SGS is then used to construct a new schedule, which is added back to the pool together with the modified insertion order. An online learning mechanism constantly assigns CPU time to the construction and improvement algorithms, based on their recent success in the search. When the search is stopped, the best solution in the pool is returned.

Our computational experiments use realistic problem instances from Bærum hospital, for three very different planning situations: The 'one day' surgery scheduling problem contains up to nine activities per patient, while the 'one week' problem contains only three. Both problems are multi-modal, require several resources for most activities, have minimum and maximum time lag constraints, project disjunction constraints and mode compatibility constraints. In addition, a long term admission planning and scheduling problem is solved. This has only one activity per patient, but each test instance has more than 700 patients. This problem is very similar to the admission planning problem that we considered in Paper I, except that a number of different resources are chosen, rather than just the operating rooms. Also, the planning horizon is longer (up to three months), and the objective components are slightly different.

The paper presents lower bounds for the various objective components, created by means of mixed integer linear programming. These lower bounds are used in the evaluation of the computational results. We show that the presented model and search method are able to produce high quality solutions for the three structurally very different problems, and within run time limits that are acceptable in the respective planning situations (2-5 minutes).

Our model was initially aimed at generalising surgery scheduling problems on all time scales and levels of detail. Indeed, based on the comprehensive reviews of Cardoen et al [26] and Demeulemeester et al [42], we show that the model covers most of the aspects of deterministic surgery scheduling problems that have been previously studied. However, the model is equally applicable to the planning and scheduling of other treatment activities, as long as such problems fall within the description of the APSP in Chapter 2.

We are not aware of any other paper that tackles such complex problems on time horizons longer than a week, or that presents such a general model for deterministic surgery scheduling problems. In Chapter 3, we mentioned three papers that take a similar approach to a generic modelling of surgery scheduling problems, although their models are all less general than ours. They all address problems with at most a one week scheduling horizon. In [127] and [128], Roland et al base their model on the RCPSP. In both papers, they use a SGS-based approach, where the activity insertion order is iteratively modified by a genetic algorithm. Their work differ from ours in various respects: First, in both papers

they consider only one activity per patient: the surgery. By contrast, our one week problem contains three activities per patient (surgery, recovery, and room cleaning), and these have minimum and maximum time lag constraints between them. This increases the complexity of our problem significantly. Second, while the model in [127] is multi-modal, each mode (for the surgery) contains the choice of only one kind of resource: the operating room (the surgeons are preassigned to each activity). By contrast, in our problem definition each surgery mode contains both an operating room and a surgeon, and each cleaning activity mode contains an operating room and the cleaning personnel. Again, this increases the complexity of our problem. Pham and Klinkert [120] model the one week problem as a *multi-mode blocking job shop problem*. The authors include the concept of *blocking constraints* that allow an activity to occupy a resource even after it has been completed, if it is not yet possible to start the next activity for the patient. Our model can also express this aspect, by a combination of time constraints and project disjunction constraints. However, we do not have such blocking constraints in any of our test problems. Pham and Klinkert solve their test instances using mixed integer programming, with a one hour time limit. Our approach, on the other hand, is heuristic. In our experiments we use a time limit of five minutes for these problems, which in our experience is a more practical time limit for real world surgery scheduling on a one week horizon.

The research done in this paper forms the foundation for further research in the AKTIV project, where the presented model is used to express general APSP problems. Also, the algorithmic framework has been generalised as a generic blackboard framework. This uses online learning and task parallelism to facilitate collaboration between different algorithmic components. The algorithms developed in this paper are included in this framework. The presented model is also the foundation for the research described in Paper IV.

PAPER IV: RECURSIVE LOGIC-BASED BENDERS' DECOMPOSITION FOR MULTI-MODE OUTPATIENT SCHEDULING

A. Riise, C. Mannino, and L. Lamorgese. *Submitted to European Journal of Operational Research, August 2015*

In this paper we address a long term appointment scheduling problem that can be found in hospital outpatient clinics. Again, this is a special case of the APSP: a multi-mode problem with integrated planning and scheduling. In our case study, the problem has only one activity per patient. Each activity requires a doctor, a room, and a *scope* (a piece of equipment). The scopes have setup times. Activity durations depends on the procedure, age, and medical condition of the patient, and on the chosen doctor's experience. The problem is defined using the model concepts that we developed in Paper III.

This type of APSPs lend themselves naturally to decomposition. In particular, they have several properties that make them suitable for logic-based Benders' decomposition (LBD), as introduced by Hooker [79, 81]. Firstly, the most important objective is typically waiting time, relative to priority based due dates. This objective depends only on the day on which each patient is scheduled, and it is therefore natural to use the assignment of patients to days as our master problem. Often, as in our case study, the remaining slave problem is a feasibility problem. Furthermore, this slave problem has a clear multi-period (daily) structure, where the schedule for each day is independent of the schedules for other days. The slave problem, which may be very large due to the long planning horizon, can therefore be split into a set of much smaller subproblems, one for each day.

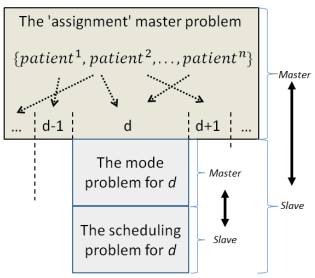

Figure 2: The three-level decomposition of a problem with n patients. The d's enumerate the days.

While this decomposition is effective, we show that it is not in itself enough to solve problem instances of a realistic size. We therefore apply a second decomposition of each daily subproblem into a master *mode problem*, where the mode for

each appointment is chosen, and a slave *scheduling problem* (see Figure 2). The scheduling problem determines if there exists a feasible start time for each activity, given the mode choices as they are fixed in the master solution. For each of these two (recursive) applications of logic-based Benders' decomposition, we present the corresponding Benders' reformulation. We also present novel inequalities that are added to strengthen the initial masters, as well as the cuts that are added from the respective slave problems. Critical elements, such as the breaking of mode-related symmetries, are explained.

The paper presents computational results from a case study at the gastroenterology laboratory at the University Hospital of North Norway, a fairly typical Norwegian hospital outpatient clinic. The numerical results demonstrate the efficiency of the proposed three-level decomposition. Based on real data from both daily and monthly (bulk) planning situations, we show that the three-level decomposition is vastly superior to the corresponding two-level version. Indeed, while the algorithm using the two-level decomposition cannot solve the monthly scheduling problems within the one hour time limit[2], the proposed three-level decomposition solves them to optimality within 5 minutes[3]. For daily scheduling, both algorithms solve all problems to optimality, but the one using the three-level decomposition is about 5 times faster on average. It also has a much smaller variation in computation time, from instance to instance.

LBD has been applied to a variety of hard discrete optimisation problems, including general planning and scheduling [84, 80, 9]. To the best of our knowledge, however, this paper presents the first application of LBD to APSPs, or to similar patient scheduling problems. The presented method is also, as far as we are aware, the first exact method that has been reported to solve such multi-mode appointment planning and scheduling problems, on a horizon of several weeks, and within a time frame that is realistic in a real planning situation. Finally, this paper is one of very few studies of multi-level LBDs. We have found only two other examples in the literature. Both solve problems that are very different from the APSP: the scheduling of computation tasks on the Cell BE processor [12], and a stochastic facility location and vehicle assignment problem [55].

[2]Except for one easy instance
[3]Except for one instance

The presented method is obviously applicable to the admission planning problem, and to other APSPs with a clear daily structure. For example, it should be possible to tackle many radiotherapy or chemotherapy scheduling problems with a minimum of adaptation. More generally, the overall structure of the recursive decomposition may be applied also for other methodologies, or even hybrid combinations of such. Depending on the choice of methods, the decomposition may be slightly different, for example including also a separate *sequencing* subproblem. Even so, such a method should exploit the general recursive decomposition technique, and the various constraint strengthening techniques that were developed in Paper IV.

PAPER V: SCHEDULING AND SEQUENCING

E.K. Burke, T. Curtois, T.E. Nordlander, and A. Riise. *In Handbook of Healthcare Delivery Systems, CRC Press (2010)*

This book chapter gives a basic introduction to some of the planning, sequencing and scheduling problems that are found in hospitals. These include personnel rostering, patient mix planning, optimisation of master surgery schedules, and operational surgery scheduling. Our contribution here was mainly to give a textbook introduction to the overall scope and diversity of operational surgery scheduling problems.

Chapter 5

Summary and outlook

In this thesis, we have considered a class of computationally hard optimisation problems where activities related to the treatment or examination of individual patients are to be scheduled on limited hospital resources. Common applications include surgery planning and scheduling on various time horizons, scheduling of non-surgical treatment such as radiotherapy or chemotherapy, and scheduling in different diagnostic or therapeutic services.

While essentially very similar, these problems are treated separately in the literature. Even within a more limited domain, such as surgery scheduling, there is a big variation in problem definitions. The development of a model that can capture the majority of these problems has been one of our goals in this thesis. The main challenge, however, has been to solve realistic APSP instances, within practical time limits. One of the reasons that these problems are hard to solve, is that they integrate planning and scheduling decisions. While this often gives better schedules, it also increases problem complexity. Such an integration has therefore been largely avoided in the literature, especially for problems with longer planning horizons. This is also the case for planning and scheduling in other domains.

Addressing these research challenges, we have made several contributions to the literature: Through the included papers, we have shown that real world APSPs can be solved, to optimality or a proven high quality, within run time limits that are reasonable for the respective planning situations. Both exact and heuristic search methods have been proposed. We have also shown that it is possible to capture structurally different problems with a generalised model. One of the proposed search methods, presented in Paper III, is very general, and can be applied

directly to very different problems without any customisation. While the methods of Papers I and IV are designed for more specific problems, these too can be generalised to cover a larger set of possible problem definitions. For example, an obvious next step is to extend the recursive decomposition algorithm of Paper IV to solve radiotherapy or chemotherapy scheduling problems.

However, since our APSP model captures such a wide range of real world problems, the fact remains that most methods will be more suitable for some problem instances than for others. To achieve a truly general scheduling solver, we have therefore designed a generic blackboard framework, in which many search methods can be included. A choice of method(s) can be based on an analysis of the problem instance. Also, online learning is used to allocate CPU time to different methods that work in parallel. The framework thus enables a collaboration between different search methods, and allows the emphasis on each method to change during the search. Several of the search methods that were created as a part of the thesis work, some of which were never published, are either directly integrated in this framework, or serve as inspiration and guidance for the ongoing development.

In the continuation of this research, we are facing several challenges. One challenge comes from problems with non-regular components in the objective function. Such problems are difficult to solve because one can no longer assume that it is optimal to start each activity as early as possible. That is, an optimal schedule is not necessarily *active* [107]. This adds another level of computational complexity to these already hard problems. Non-regular objectives, such as an 'earliness-tardiness' cost for activity start times, often occur in *rescheduling*. This happens in applications where new information arrives more or less continuously (new patient arrivals, cancellations, updated patient priorities, personnel illness, etc.), and where it is preferred that the subsequent rescheduling changes activity start times as little as possible. We are currently researching a new search method for APSPs with non-regular objectives. This method is based on a recursive decomposition similar to the one in Paper IV. It uses a hybrid combination of heuristic day assignment and exact methods for the mode assignments, sequencing, and scheduling subproblems.

Another future topic, now that we have established working methods for deterministic APSPs, is to handle stochastic aspects such as variations in activity

durations, no-shows, personnel illness, and so on. This is very challenging, both because there is a shortage of good data, and because even deterministic APSPs are computationally hard to solve. A pragmatic approach is therefore needed. It would be natural to use a method based on hedging on activity durations and/or resource availability, using discrete event simulation to tune the hedging parameters for each application (see Section 2.3). Approaches that use a scenario based evaluation of each candidate schedule should also be investigated, and are obvious candidates for parallelisation. These different approaches can also be combined. Although computationally expensive, simulations should in principle include rescheduling as the most realistic recourse action. This would obviously require very efficient rescheduling methods.

Finally, note that while we have focused on hospital applications, the APSP model is formulated in general project scheduling terms, with a rich set of constraints. The results in Papers III and IV are therefore quite general, and thus applicable also to planning and scheduling problems in other domains.

Bibliography

[1] K. Apt. *Principles of constraint programming*. Cambridge University Press, 2003.

[2] C. Artigues, S. Demassey, and E. Néron. *Resource-constrained Project Scheduling: Models, Algorithms, Extensions and Applications*. Control Systems, Robotics and Manufacturing. ISTE, London, UK, 2008.

[3] V. Augusto, X. Xie, and V. Perdomo. Operating theatre scheduling with patient recovery in both operating rooms and recovery beds. *Computers & Industrial Engineering*, 58(2):231–238, 2010.

[4] J. Bagherinejad and Z. R. Majd. Solving the mrcpsp/max with the objective of minimizing tardiness/earliness cost of activities with double genetic algorithms. *The International Journal of Advanced Manufacturing Technology*, 70(1-4):573–582, 2014.

[5] F. Ballestín, A. Barrios, and V. Valls. Looking for the best modes helps solving the mrcpsp/max. *International Journal of Production Research*, pages 1–15, 2012.

[6] A. Barrios, F. Ballestín, and V. Valls. A double genetic algorithm for the mrcpsp/max. *Computers & Operations Research*, 38(1):33–43, 2011.

[7] M. Bartusch, R. H. Möhring, and F. J. Radermacher. Scheduling project networks with resource constraints and time windows. *Annals of Operations Research*, 16(1):199–240, 1988.

[8] S. Batun and M. A. Begen. Optimization in healthcare delivery modeling: Methods and applications. In *Handbook of Healthcare Operations Management*, pages 75–119. Springer, 2013.

[9] J. C. Beck. Checking-up on branch-and-check. In *Principles and Practice of Constraint Programming–CP 2010*, pages 84–98. Springer, 2010.

[10] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust optimization*. Princeton University Press, 2009.

[11] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4(1):238–252, 1962.

[12] L. Benini, M. Lombardi, M. Mantovani, M. Milano, and M. Ruggiero. Multi-stage benders decomposition for optimizing multicore architectures. In M. Perron and M. Trick, editors, *Fifth International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Oprimization Problems*, pages 36–50, 2008.

[13] D. Bertsimas, D. B. Brown, and C. Caramanis. Theory and applications of robust optimization. *SIAM review*, 53(3):464–501, 2011.

[14] J. R. Birge and F. Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[15] J. T. Blake and M. Carter. Surgical process scheduling: a structured review. *J Soc Health Syst*, 5(3):17–30, 1997.

[16] J. T. Blake, F. Dexter, and J. Donald. Operating room managers' use of integer programming for assigning block time to surgical groups: A case study. *Anesth Analg. 2002 Jan;94(1):143-8, table of contents.*, 94(1):143–148, 2002.

[17] J. Blazewicz, J. K. Lenstra, and A. H. G. R. Kan. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5(1):11–24, 1983. 0166-218X.

[18] A. Braaksma, N. Kortbeek, G. Post, and F. Nollet. Integral multidisciplinary rehabilitation treatment planning. *Operations Research for Health Care*, 3(3):145–159, 2014.

[19] A. R. Brodtkorb, C. Dyken, T. R. Hagen, J. M. Hjelmervik, and O. O. Storaasli. State-of-the-art in heterogeneous computing. *Sci. Program.*, 18(1):1–33, 2010.

[20] A. R. Brodtkorb, T. R. Hagen, C. Schulz, and G. Hasle. Gpu computing in discrete optimization. part i: Introduction to the gpu. *EURO Journal on Transportation and Logistics*, 2(1-2):129–157, 2013.

[21] P. Brucker, A. Drexl, R. Möhring, K. Neumann, and E. Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112(1):3–41, 1999. 0377-2217.

[22] N. Bulgarini, D. Di Lorenzo, A. Lori, D. Matarrese, and F. Schoen. Operating room joint planning and scheduling. In *Proceedings of the International Conference on Health Care Systems Engineering*, pages 127–138. Springer, 2014.

[23] K. M. Calhoun, R. F. Deckro, J. T. Moore, J. W. Chrissis, and J. C. Van Hove. Planning and re-planning in project and production scheduling. *Omega*, 30(3):155–170, 2002.

[24] B. Cardoen, E. Demeulemeester, and J. Beliën. Optimizing a multiple objective surgical case sequencing problem. *International Journal of Production Economics*, 119(2):354–366, 2009. 0925-5273.

[25] B. Cardoen, E. Demeulemeester, and J. Beliën. Sequencing surgical cases in a day-care environment: An exact branch-and-price approach. *Computers & Operations Research*, 36(9):2660–2669, 2009. 0305-0548.

[26] B. Cardoen, E. Demeulemeester, and J. Beliën. Operating room planning and scheduling: A literature review. *European Journal of Operational Research*, 201(3):921–932, 2010. 0377-2217.

[27] J. Cares, M.-C. Riff, and B. Neveu. Genera: A benchmarks generator of radiotherapy treatment scheduling problem. In P. M. Pardalos, M. G. C. Resende, C. Vogiatzis, and J. L. Walteros, editors, *Learning and Intelligent Optimization*, volume 8426 of *Lecture Notes in Computer Science*, book section 30, pages 353–361. Springer International Publishing, 2014.

[28] J. Cares, M.-C. Riff, and B. Neveu. Genera: A benchmarks generator of radiotherapy treatment scheduling problem, 2015. `http://www.inf.utfsm.cl/~{}jcares/genera/`.

[29] T. Cayirli and E. Veral. Outpatient scheduling in health care: A review of literature. *Production and Operations Management*, 12(4):519–549, 2003.

[30] S. Chaabane, N. Meskens, A. Guinet, and M. Laurent. Comparison of two methods of operating theatre planning: Application in belgian hospital. In *2006 International Conference on Service Systems and Service Management*, 2006.

[31] J. Charnetski. Scheduling operating room surgical procedures with early and late completion penalty costs. *Journal of Operations Management*, 5(1):91–102, 1984.

[32] C.-F. Chien, F.-P. Tseng, and C.-H. Chen. An evolutionary approach to rehabilitation patient scheduling: A case study. *European Journal of Operational Research*, 189(3):1234–1253, 2008.

[33] CHOIR. Categorized bibliography for operations research/management science in health care. retrieved 23rd march 2015, from: http://www.choir-ut.nl/, 2015.

[34] D. Conforti, F. Guerriero, and R. Guido. Optimization models for radiotherapy patient scheduling. *4OR*, 6(3):263–278, 2008.

[35] D. Conforti, F. Guerriero, and R. Guido. Non-block scheduling with priority for radiotherapy treatments. *European Journal of Operational Research*, 201(1):289–296, 2010.

[36] D. Conforti, F. Guerriero, R. Guido, and M. Veltri. An optimal decision-making approach for the management of radiotherapy patients. *OR Spectrum*, 33(1):123–148, 2011.

[37] G. B. Dantzig. Linear programming under uncertainty. *Management science*, 1(3-4):197–206, 1955.

[38] G. B. Dantzig and M. N. Thapa. *Linear Programming 1: 1: Introduction*, volume 1. Springer Science & Business Media, 1997.

[39] S. Dauzère-Pérès, W. Roux, and J. B. Lasserre. Multi-resource shop scheduling with resource flexibility. *European Journal of Operational Research*, 107(2):289–305, 1998. 0377-2217.

[40] B. De Reyck and W. Herroelen. The multi-mode resource-constrained project scheduling problem with generalized precedence relations. *European Journal of Operational Research*, 119(2):538–556, 1999.

[41] K. Deb. Multi-objective optimization. In *Search methodologies*, pages 403–449. Springer, 2014.

[42] E. Demeulemeester, J. Beliën, B. Cardoen, and M. Samudra. Operating room planning and scheduling. In *Handbook of Healthcare Operations Management*, volume 184 of *International Series in Operations Research & Management Science*, book section 5, pages 121–152. Springer New York, 2013.

[43] Y. Deng, S. Shen, and B. Denton. Chance-constrained surgery planning under uncertain or ambiguous surgery duration. *Available at SSRN 2432375*, 2014.

[44] B. T. Denton. *Handbook of Healthcare Operations Management: Methods and Applications*, volume 184. Springer Science & Business Media, 2013.

[45] G. Desaulniers, J. Desrosiers, and M. M. Solomon. *Column generation*, volume 5. Springer Science & Business Media, 2005.

[46] F. Dexter. Bibliography of operating room management articles. retrieved 23rd march 2015, from: http://www.franklindexter.com/, 2015.

[47] F. Dexter and A. Macario. Changing allocations of operating room time from a system based on historical utilization to one where the aim is to schedule as many surgical cases as possible. *Anesth Analg*, 94(5):1272–1279, 2002.

[48] M. Dorigo and M. Birattari. Ant colony optimization. In *Encyclopedia of Machine Learning*, pages 36–39. Springer, 2010.

[49] S. H. H. Doulabi, L.-M. Rousseau, and G. Pesant. A constraint programming-based column generation approach for operating room planning and scheduling. In *Integration of AI and OR Techniques in Constraint Programming*, pages 455–463. Springer, 2014.

[50] A. Drexl, R. Nissen, J. H. Patterson, and F. Salewski. Progen/[pi]x - an instance generator for resource-constrained project scheduling problems with partially renewable resources and further extensions. *European Journal of Operational Research*, 125(1):59–72, 2000. 0377-2217.

[51] A. E. Eiben and J. E. Smith. *Introduction to evolutionary computing*. Springer Science & Business Media, 2003.

[52] S. E. Elmaghraby. *Activity networks: Project planning and control by network models*. Network analysis (Planning). Wiley, New York, 1977.

[53] S. A. Erdogan and B. Denton. Dynamic appointment scheduling of a stochastic server with uncertain demand. *INFORMS Journal on Computing*, 25(1):116–132, 2013.

[54] S. A. Erdogan, B. T. Denton, J. Cochran, L. Cox, P. Keskinocak, J. Kharoufeh, and J. Smith. Surgery planning and scheduling. *Wiley Encyclopedia of operations research and management science*, 2011.

[55] M. M. Fazel-Zarandi, O. Berman, and J. C. Beck. Solving a stochastic facility location/fleet management problem with logic-based benders' decomposition. *IIE Transactions*, 45(8):896–911, 2013.

[56] H. Fei, C. Combes, C. Chu, and N. Meskens. Endoscopies scheduling problem: a case study. In *IFAC Symposium on Information Control Problems in Manufacturing*, 2006.

[57] H. Fei, N. Meskens, and C. Chu. A planning and scheduling problem for an operating theatre using an open scheduling strategy. *Computers & Industrial Engineering*, 58(2):221–230, 2010.

[58] B. Franck, K. Neumann, and C. Schwindt. Truncated branch-and-bound, schedule-construction, and schedule-improvement procedures for resource-constrained project scheduling. *OR Spectrum*, 23(3):297–324, 2001.

[59] M. R. Garey and D. S. Johnson. *Computers and Intractability – A Guide to the Theory of NP-completeness*. W. H. Freeman & Co. New York, 1979.

[60] D. Gartner and R. Kolisch. Scheduling the hospital-wide flow of elective patients. *European Journal of Operational Research*, 233(3):689–699, 2014.

[61] S. Ghazalbash, M. M. Sepehri, P. Shadpour, and A. Atighehchian. Operating room scheduling in teaching hospitals. *Advances in Operations Research*, 2012, 2012.

[62] F. Glover and G. Kochenberger. *Handbook of Metaheuristics (International Series in Operations Research & Management Science)*. Springer, 2003.

[63] F. Glover and M. Laguna. *Tabu search*. Springer, 1999.

[64] F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. *Control and cybernetics*, 29:653–684, 2000.

[65] J. D. Griffiths, J. E. Williams, and R. M. Wood. Scheduling physiotherapy treatment in an inpatient setting. *Operations Research for Health Care*, 1(4):65–72, 2012.

[66] K. GROUP. Opencl web site as per 29th march, 2015, 2015. `https://www.khronos.org/opencl/`.

[67] F. Guerriero and R. Guido. Operational research in the management of the operating theatre: a survey. *Health Care Management Science*, 14(1):89–114, 2011.

[68] A. Guinet and S. Chaabane. Operating theatre planning. *International Journal of Production Economics Planning and Control of Productive Systems*, 85(1):69–81, 2003. TY - JOUR.

[69] S. Gul, B. T. Denton, J. W. Fowler, and T. Huschka. Bi-criteria scheduling of surgical services for an outpatient procedure center. *Production and Operations management*, 20(3):406–417, 2011.

[70] D. Gupta and B. Denton. Appointment scheduling in health care: Challenges and opportunities. *IIE Transactions*, 40(9):800–819, 2008.

[71] S. Hahn-Goldberg. *Dynamic optimization addressing chemotherapy outpatient scheduling*. Thesis, University of Toronto, 2014.

[72] E. Hans, G. Wullink, M. van Houdenhoven, and G. Kazemier. Robust surgery loading. *European Journal of Operational Research*, 185(3):1038–1050, 2008.

[73] P. Hansen and N. Mladenović. Variable neighbourhood search. In E. K. Burke and G. Kendall, editors, *Search Methodologies - Introductory Tutorials in Optimization and Decision Support Techniques*, volume 1, pages 211–238. Springer, 2005.

[74] P. Hansen, N. Mladenović, and J. Moreno Pérez. Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175(1):367–407, 2010.

[75] S. Harding and W. Banzhaf. Fast genetic programming on gpus. In *Genetic Programming*, pages 90–101. Springer Berlin Heidelberg, 2007.

[76] S. Hartmann and D. Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207(1):1–14, 2010. 0377-2217.

[77] R. Heilmann. Resource–constrained project scheduling: a heuristic for the multi–mode case. *OR Spectrum*, 23(3):335–357, 2001.

[78] R. Heilmann. A branch-and-bound procedure for the multi-mode resource-constrained project scheduling problem with minimum and maximum time lags. *European Journal of Operational Research*, 144(2):348–365, 2003.

[79] J. N. Hooker. *Logic-based Methods for Optimization*. Wiley, 2000.

[80] J. N. Hooker. Planning and scheduling by logic-based benders decomposition. *Operations Research*, 55(3):588–602, 2007.

[81] J. N. Hooker and G. Ottosson. Logic-based benders decomposition. *Mathematical Programming*, 96(1):33–60, 2003.

[82] H. H. Hoos and T. Stützle. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, 2005.

[83] P. J. H. Hulshof, N. Kortbeek, R. J. Boucherie, E. W. Hans, and P. J. M. Bakker. Taxonomic classification of planning decisions in health care: a structured review of the state of the art in or/ms. *HS*, 1(2):129–175, 2012.

[84] V. Jain and I. Grossmann. Algorithms for hybrid milp/cp models for a class of optimization problems. *INFORMS Journal on Computing*, 13:258–276, 2001.

[85] A. Jebali, A. B. Hadj Alouane, and P. Ladet. Operating rooms scheduling. *International Journal of Production Economics Control and Management of Productive Systems*, 99(1-2):52–62, 2006. TY - JOUR.

[86] P. Kali and S. W. Wallace. *Stochastic programming*. Springer, 1994.

[87] T. Kapamara, K. Sheibani, O. Haas, C. Reeves, and D. Petrovic. A review of scheduling problems in radiotherapy. In *Proceedings of the Eighteenth International Conference on Systems Engineering (ICSE2006), Coventry University, UK*, pages 201–207, 2006.

[88] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[89] J. Kennedy. Particle swarm optimization. In *Encyclopedia of Machine Learning*, pages 760–766. Springer, 2010.

[90] R. Kolisch. *Project scheduling under resource constraints: efficient heuristics for several problem classes*. Springer Verlag, 1995.

[91] R. Kolisch and S. Hartmann. Heuristic algorithms for solving the resource-constrained project scheduling problem: classification and computational analysis. In *Project Scheduling - Recent Models, Algorithms and Applications*, pages 147–178. Kluwer Academic Publishers, Boston., 1999.

[92] R. Kolisch and S. Hartmann. Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, 174(1):23–37, 2006.

[93] W. Langdon and W. Banzhaf. A simd interpreter for genetic programming on gpu graphics cards. In *Genetic Programming*, pages 73–85. Springer Berlin Heidelberg, 2008.

[94] A. Legrain, M.-A. Fortin, N. Lahrichi, and L.-M. Rousseau. Online stochastic optimization of radiotherapy patient scheduling. *Health care management science*, pages 1–14, 2014.

[95] M. Lombardi and M. Milano. Optimal methods for resource allocation and scheduling: a cross-disciplinary survey. *Constraints*, 17(1):51–85, 2012.

[96] H. R. D. Lourenco, O. C. Martin, and T. Stutzle. Iterated local search. In *Handbook of Metaheuristics*, pages 321–353. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[97] A. Lova, P. Tormos, and F. Barber. Multi-mode resource constrained project scheduling: Scheduling schemes, priority rules and mode selection rules. *Inteligencia artificial*, 30(10):69–86, 2006.

[98] J. M. Magerlein and J. B. Martin. Surgical demand scheduling: a review. *Health Services Research*, 1978.

[99] C. Mannino, E. Nilssen, and T. Nordlander. A pattern based, robust approach to cyclic master surgery scheduling. *Journal of Scheduling*, 15(5):553–563, 2012.

[100] C. T. Maravelias and C. Sung. Integration of production planning and scheduling: Overview, challenges and opportunities. *Computers & Chemical Engineering*, 33(12):1919–1930, 2009.

[101] I. Marques, M. E. Captivo, and M. V. Pato. An integer programming approach to elective surgery scheduling. *OR spectrum*, 34(2):407–427, 2012.

[102] I. Marques, M. E. Captivo, and M. V. Pato. A bicriteria heuristic for an elective surgery scheduling problem. *Health care management science*, pages 1–16, 2014.

[103] I. Marques, M. E. Captivo, and M. Vaz Pato. Scheduling elective surgeries in a portuguese hospital using a genetic heuristic. *Operations Research for Health Care*, 3(2):59–72, 2014.

[104] J. H. May, W. E. Spangler, D. P. Strum, and L. G. Vargas. The surgical scheduling problem: Current research and future opportunities. *Production and Operations Management*, 20(3):392–405, 2011.

[105] N. Meskens, D. Duvivier, and A. Hanset. Multi-objective operating room scheduling considering desiderata of the surgical team. *Decision Support Systems*, 55(2):650–659, 2013.

[106] J. Molina-Pariente, V. Fernandez-Viagas, and J. Framinan. Integrated operating room planning and scheduling problem with assistant surgeon dependent surgery durations. *Computers & Industrial Engineering*, 2015.

[107] K. Neumann, C. Schwindt, and J. Zimmermann. *Project Scheduling with Time Windows and Scarce Resources*. Springer-Verlag, New York, 2 edition, 2003.

[108] NVIDIA. Cuda zone web site, as per 29th march, 2015, 2015. `https://developer.nvidia.com/cuda-zone`.

[109] OECD, 2014. `http://www.oecd.org/health`.

[110] S. N. Ogulata and R. Erol. A hierarchical multiple criteria mathematical programming approach for scheduling general surgery operations in large hospitals. *Journal of Medical Systems*, 27(3):259–270, 2003. TY - JOUR.

[111] S. N. Ogulata, M. Koyuncu, and E. Karakas. Personnel and patient scheduling in the high demanded hospital services: A case study in the physiotherapy service. *Journal of Medical Systems*, 32(3):221–228, 2008.

[112] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips. Gpu computing. *Proceedings of the IEEE*, 96(5):879–899, 2008.

[113] I. Ozkarahan. Allocation of surgical procedures to operating rooms. *Journal of Medical Systems*, 19(4):333–352, 1995. 10.1007/BF02257264.

[114] I. Ozkarahan. Allocation of surgeries to operating rooms by goal programing. *Journal of Medical Systems*, 24(6):339–378, 2000. TY - JOUR.

[115] J. Patrick, M. L. Puterman, and M. Queyranne. Dynamic multi-priority patient scheduling for a diagnostic resource. *Operations Research*, 56(6):1507–1525, 2008.

[116] E. Pérez, L. Ntaimo, C. O. Malavé, C. Bailey, and P. McCormack. Stochastic online appointment scheduling of multi-step sequential procedures in nuclear medicine. *Health care management science*, 16(4):281–299, 2013.

[117] S. Petrovic and E. Castro. A genetic algorithm for radiotherapy pretreatment scheduling. In *Applications of Evolutionary Computation*, pages 454–463. Springer, 2011.

[118] S. Petrovic and P. Leite-Rocha. Constructive and grasp approaches to radiotherapy treatment scheduling. In *World Congress on Engineering and Computer Science 2008, WCECS'08. Advances in Electrical and Electronics Engineering-IAENG Special Edition of the*, pages 192–200. IEEE, 2008.

[119] S. Petrovic, W. Leung, X. Song, and S. Sundar. Algorithms for radiotherapy treatment booking. In *Proceedings of the 25th Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG'2006), Nottingham, UK*, 2006.

[120] D.-N. Pham and A. Klinkert. Surgical case scheduling as a generalized job shop scheduling problem. *European Journal of Operational Research*, 185(3):1011–1025, 2008. 0377-2217.

[121] V. Podgorelec and P. Kokol. Genetic algorithm based system for patient scheduling in highly constrained situations. *Journal of medical systems*, 21(6):417–427, 1997.

[122] A. A. B. Pritsker, L. J. Watters, and P. M. Wolfe. Multiproject scheduling with limited resources: A zero-one programming approach. *Management Science*, 16(1):93–108, 1969.

[123] R. Pulido, A. M. Aguirre, N. Ibáñez-Herrero, M. Ortega-Mier, l. García-Sánchez, and C. A. Méndez. Optimization methods for the operating room management under uncertainty: stochastic programming vs. decomposition approach. *Journal of Applied Operational Research*, 6(3):145–157, 2014.

[124] A. Rais and A. Viana. Operations research in healthcare: a survey. *International Transactions in Operational Research*, 18(1):1–31, 2011.

[125] M. G. Resende and C. C. Ribeiro. Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In *Handbook of metaheuristics*, pages 283–319. Springer, 2010.

[126] A. Riise and C. Mannino. The surgery scheduling problem - a general model. Report, SINTEF, 2012. ISBN 978-82-14-05281-7.

[127] B. Roland, C. Di Martinelly, and F. Riane. Operating theatre optimization : A resource-constrained based solving approach. In *Service Systems and Service Management, 2006 International Conference on*, volume 1, pages 443–448, 2006.

[128] B. Roland, C. Di Martinelly, F. Riane, and Y. Pochet. Scheduling an operating theatre under human resource constraints. *Computers & Industrial Engineering*, 58(2):212–220, 2010.

[129] H. Saadouli, B. Jerbi, A. Dammak, L. Masmoudi, and A. Bouaziz. A stochastic optimization and simulation approach for scheduling operating rooms and recovery beds in an orthopedic surgery department. *Computers & Industrial Engineering*, 80(0):72–79, 2015.

[130] H. Saadouli, M. Masmoudi, B. Jerbi, and A. Dammak. An optimization and simulation approach for operating room scheduling under stochastic durations. In *Control, Decision and Information Technologies (CoDIT), 2014 International Conference on*, pages 257–262, 2014.

[131] M. Sabzehparvar and S. M. Seyed-Hosseini. A mathematical model for the multi-mode resource-constrained project scheduling problem with mode dependent time lags. *The Journal of Supercomputing*, 44(3):257–273, 2008.

[132] A. Sadki, X. Xie, and F. Chauvin. Bedload balancing for an oncology ambulatory care unit. In *8th International Conference of Modeling and Simulation-MOSIM'10*. Citeseer, 2010.

[133] A. Sadki, X. Xie, and F. Chauvin. Patients assignment for an oncology outpatient unit. In *Automation Science and Engineering (CASE), 2010 IEEE Conference on*, pages 891–896. IEEE, 2010.

[134] P. Santibanez, M. Begen, and D. Atkins. Surgical block scheduling in a system of hospitals: an application to resource and wait list management in a british columbia health authority. *Health Care Management Science*, 10(3):269–282, 2007. TY - JOUR.

[135] A. Saremi, P. Jula, T. ElMekkawy, and G. G. Wang. Appointment scheduling of outpatient surgical services in a multistage operating room department. *International Journal of Production Economics*, 141(2):646–658, 2013.

[136] M. Schatz and C. Trapnell. Fast exact string matching on the gpu. Report, Center for Bioinformatics and Computational Biology, 2007.

[137] K. Schimmelpfeng, S. Helber, and S. Kasper. Decision support for rehabilitation hospital scheduling. *OR spectrum*, 34(2):461–489, 2012.

[138] C. Schulz, G. Hasle, A. R. Brodtkorb, and T. R. Hagen. Gpu computing in discrete optimization. part ii: Survey focused on routing problems. *EURO Journal on Transportation and Logistics*, 2(1-2):159–186, 2013.

[139] C. Serrano, A. Jiménez, C. Amaya, and N. Velasco. Optimization model to minimize the makespan in a hospital's gastroenterology service. Report, Universidad de los Andes, 2009.

[140] D. Sier, P. Tobin, and C. McGurk. Scheduling surgical procedures. *Journal of the Operational Research Society*, 48(9):884–891, 1997. ISI Document Delivery No.: XU899 Times Cited: 5 Cited Reference Count: 11.

[141] A. Turkcan, B. Zeng, and M. Lawley. Chemotherapy operations planning and scheduling. *IIE Transactions on Healthcare Systems Engineering*, 2(1):31–49, 2012.

[142] C. Van Huele and M. Vanhoucke. Analysis of the integration of the physician rostering problem and the surgery scheduling problem. *Journal of medical systems*, 38(6):1–16, 2014.

[143] C. Van Huele and M. Vanhoucke. Decomposition-based heuristics for the integrated physician rostering and surgery scheduling problem. *Health Systems*, 2014.

[144] P. J. Van Laarhoven and E. H. Aarts. *Simulated annealing*. Springer, 1987.

[145] B. Vijayakumar, P. J. Parikh, R. Scott, A. Barnes, and J. Gallimore. A dual bin-packing approach to scheduling surgical cases at a publicly-funded hospital. *European Journal of Operational Research*, 224(3):583–591, 2013.

[146] C. Voudouris and E. P. Tsang. *Guided local search*. Springer, 2003.

[147] Y. Wang, J. Tang, and R. Y. Fung. A column-generation-based heuristic algorithm for solving operating theater planning problem under stochastic demand and surgery cancellation risk. *International Journal of Production Economics*, 158:28–36, 2014.

[148] L. A. Wolsey. Mixed integer programming. In *Wiley Encyclopedia of Computer Science and Engineering*. John Wiley & Sons, Inc., 2007.

[149] W. Xiang, J. Yin, and G. Lim. A short-term operating room surgery scheduling. *surgery*, 2:6, 2014.

[150] E. Yellig and G. Mackulak. Robust deterministic scheduling in stochastic environments: The method of capacity hedge points. *International Journal of Production Research*, 35(2):369–379, 1997.

[151] Q. Yu, C. Chen, and Z. Pan. Parallel genetic algorithms on programmable graphics hardware. In *Advances in Natural Computation*, pages 1051–1059. Springer Berlin Heidelberg, 2005.

[152] Z. Zhao and X. Li. Scheduling elective surgeries with sequence-dependent setup times to multiple operating rooms using constraint programming. *Operations Research for Health Care*, 3(3):160–167, 2014.

[153] L. Zhongwen and L. Hongzhi. Cellular genetic algorithms and local search for 3-sat problem on graphic hardware. In *IEEE Congress on Evolutionary Computation, 2006. CEC 2006*, pages 2988–2992, 2006.