FACULTEIT ECONOMISCHE EN
TOEGEPASTE ECONOMISCHE
WETENSCHAPPEN

KATHOLIEKE
UNIVERSITEIT
LEUVEN

Exact and Heuristic Methodologies for Scheduling in Hospitals: Problems, Formulations and Algorithms

Proefschrift voorgedragen
tot het behalen van de graad
van Doctor in de Toegepaste
Economische Wetenschappen

door

Jeroen BELIEN

Nummer 214 2006

Committee

Prof. dr. E. Demeulemeester (advisor)	*Katholieke Universiteit Leuven*
Prof. dr. W. Herroelen	*Katholieke Universiteit Leuven*
Prof. dr. M. Lambrecht	*Katholieke Universiteit Leuven*
Prof. dr. W. Sermeus	*Katholieke Universiteit Leuven*
Prof. dr. M. Vanhoucke	*Universiteit Gent*
	Vlerick Leuven Gent Management School
Prof. dr. E. Burke	*University of Nottingham*

Daar de proefschriften in de reeks van de Faculteit Economische en Toegepaste Economische Wetenschappen het persoonlijk werk zijn van hun auteurs, zijn alleen deze laatsten daarvoor verantwoordelijk.

Dankwoord

Het schrijven van dit dankwoord geeft me de kans om eventjes stil te staan bij de afgelopen vier jaar. Het zijn vier mooie jaren geweest, waarin ik mezelf enorm heb kunnen ontplooien op allerlei vlakken. Gaandeweg ben ik echt gaan houden van onderzoek doen. Ik ben dan ook fier op dit resultaat. Maar bovenal voel ik me dankbaar aan al diegenen die me, rechtstreeks of onrechtstreeks, in grote of kleine mate, geholpen hebben.

Op de eerste plaats denk ik natuurlijk aan mijn promotor Professor Erik Demeulemeester. Die formele titel voelt een beetje vreemd aan, want al op de eerste dag vroeg hij me "Professor" maar achterwege te laten en hem gewoon "Erik" te noemen. Het typeert hem. Ik vergelijk onze relatie graag met de relatie tussen een ridder en een schildknaap. Zoals een ridder de schildknaap leert omgaan met wapens en paarden, bracht Erik me de kneepjes van het vak van de onderzoeker bij. Hij introduceerde me op de grote tornooien (congressen) en zorgde ervoor dat ik steeds met een indrukwekkend wapenarsenaal op het strijdperk verscheen. Maar hij liet me ook voldoende vrijheid en creativiteit zodat ik mijn onderzoek steeds als een boeiende uitdaging ervaarde. Ik denk over voldoende levenservaring te beschikken om te stellen dat hetgeen Erik voor me gedaan heeft, niet iets vanzelfsprekends is.

Niet alleen had ik het geluk een goede promotor te hebben, ook kwam ik in een bijzonder aangename onderzoeksgroep terecht. Dit is in niet geringe mate te danken aan Professor Willy Herroelen en Professor Marc Lambrecht. Beiden zetelen bovendien in mijn commissie en ik dank hen voor de interessante vragen en discussies die mee hebben bijgedragen tot de totstandkoming van dit werk. Het formele respect dat ik aanvankelijk voor hen had, heeft geleidelijk aan plaats gemaakt voor echte waardering en zelfs bewondering. Ik herinner me nog levendig hoe tijdens mijn

eerste congres een Spaanse ober bij Willy informeerde of hij op stap was met al zijn kinderen. We hebben er toen hartelijk om gelachen, maar in zekere zin zat de brave man er niet zo ver naast. Willy is de wortel van een imposante wetenschappelijke stamboom die reeds uit vier generaties bestaat. Marc was de afgelopen vier jaar de voorzitter van ons departement. Een taak die hij met hart en ziel op zich nam. Hoe beter ik hem leerde kennen, hoe meer raakvlakken ik met hem vond. Zo zijn we beiden fysiek verhinderd om te voetballen, maar supporteren we wel voor dezelfde club.

Professor Walter Sermeus heeft in vele opzichten een belangrijke bijdrage geleverd. Al van meet af aan was hij één van de bezielers van dit project. Als tussenschakel naar de ziekenhuizen zorgde hij herhaaldelijk voor nieuwe inspiratie, contactpersonen en praktijkdata. Als commissielid waakte hij tenslotte mee over de kwaliteit van dit proefschrift. Toen ik laatst iets over hem opzocht op het Internet, kreeg ik pas notie van zijn buitengewone staat van dienst. Toch heb ik hem altijd gekend als een zeer vriendelijk en bescheiden man.

Professor Mario Vanhoucke is in een vorig leven ook assistent geweest van Erik en kan aldus een beetje als mijn voorganger beschouwd worden. Ik heb altijd enorm opgekeken naar mijn voorganger. Ik zal nooit vergeten hoe ik hem heb leren kennen in Valencia. Als jonge snaak kwam ik toen in een nieuwe wereld terecht waarin hij fungeerde als de gedroomde gids. Hij is één van die weinige mensen waarmee het meteen goed klikte op allerlei vlakken (Nirvana, ...). Ook Erik is dit niet ontgaan. Ik heb het altijd als een enorme eer ervaren wanneer hij opmerkte dat hij blijkbaar steeds hetzelfde type van assistenten aantrekt.

I thank Professor Edmund Burke from the University of Nottingham for the thorough reading of my manuscript. It was a real honor for us to have such an expert in automated scheduling and timetabling in my committee. His useful comments and suggestions have undoubtedly improved both the quality and the readability of this thesis.

Tijdens de congressen genoot ik steeds van de gemoedelijke en collegiale sfeer onder de assistenten (en proffen). Stijn, Robert, Roel, Kristof, Olivier, Dries, Jade, Brecht, de mannen van Gent (Dieter en Broos): het zou niet hetzelfde geweest zijn zonder jullie. En Stefan, hopelijk zal ik ooit met jou eens op congres gaan, want je was het laatste jaar de motor achter onze sociale activiteiten. Ook de andere leden

van de vakgroepen Productie & Logistiek en ORSTAT wens ik te bedanken voor de toffe werksfeer. Vooral tijdens de externe activiteiten zoals de departementskwissen, spelletjesavonden, etentjes en bedrijfsbezoeken werd duidelijk dat we een hechte groep vormden. Anneleen, onze dagelijkse babbel tijdens het woonwerkverkeer was voor mij een heel fijne manier om de werkdag te beginnen en af te sluiten.

Een speciale vermelding gaat uit naar mijn bureau-genoten: Peter en Dries. Wat betreft Dries wilde het toeval dat ik ooit een Deense professor als volgt over zijn coauteur hoorde spreken: "My coauthor is a good guy, however it seems I will never get rid of him. I first met him during my undergraduate studies and from then on he kept following me. After graduation he also started a Ph.D. and thereafter we became colleagues at the same university. In between, he married my sister. But above all, he is a real friend of mine and I'm really proud to present you this coauthored paper." Ik hoop dat ik ooit precies dezelfde inleiding kan geven over Dries. In ieder geval zijn we al goed op weg. Peter heeft me de eerste drie jaar op vele vlakken wegwijs gemaakt, zoals bv. de organisatie van Megabike en de tekstverwerking in LaTeX. Voor dat laatste wens ik ook Jan Adem te bedanken die me een electronische versie van zijn proefschrift bezorgde en me alzo veel tijd deed besparen. Tenslotte dank ik Professor Spieksma voor het meedenken over mijn werk, getuige daarvan zijn talloze suggesties, en voor zijn bereidheid om als voorzitter op te treden tijdens mijn publieke verdediging.

Verder wens ik ook Kris Vanhaecht (Centrum voor Ziekenhuis- en Verplegingswetenschap, Leuven), Jenny Cristael (Oogziekenhuis Gasthuisberg, Leuven), Pierre Luysmans en Professor Guy Bogaert (Chirurgisch Dagcentrum Gasthuisberg, Leuven) en Jurgen Huygh en Geert Moechars (Virga Jesse Ziekenhuis, Hasselt) te bedanken voor hun steun en enthousiasme voor dit project enerzijds en het bezorgen van praktijkdata anderzijds.

Wat het financiële aspect betreft, wens ik een woordje van dank te richten aan elke lotto-speler en belastingbetaler wiens bijdrage op de één of andere manier bij het Fonds voor Wetenschappelijk Onderzoek (FWO) Vlaanderen terechtgekomen is. Uiteraard waardeer ik dan ook dat het FWO heeft besloten geld vrij te maken voor dit project (onder contract nummer G.0463.04). Daarnaast ben ik ons wijlen departement TEW dankbaar voor de financiële steun.

Ik had enorm veel respect voor de manier waarop Chris Massie de doctoraatsstudenten van ons departement als een moederkloek onder haar hoede nam. Samen met Elke en Isabelle zorgde ze ervoor dat alles altijd tot in de puntjes geregeld was.

Mama en papa, ik weet dat jullie nog veel trotser zijn dan ikzelf met dit doctoraat. Misschien is dat ergens wel logisch, want jullie hebben er ongetwijfeld de grootste verdienste aan. Dankzij jullie heb ik me nog nooit zorgen hoeven te maken op financieel vlak. Maar nog veel belangrijker zijn de waarden die jullie aan me doorgegeven hebben via een opvoeding waarin vrijheid steeds centraal stond. Het is bepalend geweest voor de persoon die ik geworden ben. Zowel voor mijn studies als voor dit doctoraat toonden jullie vaak oprechte interesse zonder ook maar één moment bemoeizuchtig te worden.

Tim en Elke, het was me een genoegen om met jullie mijn jeugdjaren te delen. Het ziet er naar uit dat ik moeilijk nog van jullie af ga komen. Elke zie ik opnieuw steeds vaker via Dries. En Tim, met jou woon ik nog altijd samen onder één dak. Ook jij zal binnenkort afdoctoreren zodat onze tweelingsband een nieuwe dimensie krijgt. Ons kot van doctoraatsstudenten begint stilaan een kot van doctors te worden. Bart en Leen hebben het ons voorgedaan; Tim, Kristien en An zullen nog volgen. Het was fijn om mensen rondom me te hebben die weten hoe moeilijk de bevalling van een doctoraat soms is. Onze regelmatige onderlinge aanmoedigingen betekenden een enorme steun voor me.

Voor ontspanning waren er de wekelijkse trainingen en wedstrijden met de rolstoelbasket. Samen winnen en verliezen, dat creëert een band. Ik dank mijn basketvrienden voor al die plezante onzin die ze de afgelopen jaren uitgekraamd hebben, waardoor ze onbewust mijn geest verlichtten.

Tijdens de weekends kon ik steeds terecht bij mijn schoonfamilie in Gooik. Kan een mens zich een zaliger ontspanningsoord indenken? Chef kokkin Martine bereidde telkens opnieuw een maaltijd waarvoor je bij de Comme Chez Soi minstens 500 Euro neertelt. Martin zorgde af en toe voor wat afwisseling in mijn zittend bestaan en deed me beseffen wat echt werken was. Koen, Kristien en Lieve waren soms echte kwelduivels, maar zorgden steeds voor de welgewaardeerde animo en ontspanning. Dankzij jullie werd mijn hoofd volledig vrij gemaakt voor een nieuwe week. En dit

alles op één van de mooiste plekjes in het rustieke, glooiende Pajottenland.

Tot slot richt ik mij tot jou, Annemie, al bijna acht jaar mijn engel, steun en toeverlaat. De beëindiging van dit doctoraat is maar een kantlijn in vergelijking met die andere belangrijke afspraak in 2006. Je beseft maar half hoe gelukkig je me maakt. Trouwen met jou is voor mij een droom die werkelijkheid wordt.

Jeroen Beliën
November 2005

Abstract

The scheduling of people and resources is a key issue in modern hospital management. Well-thought-out scheduling practices entail several financial and social benefits. However, due to the increased pressure on scarce resources and the proliferation in rules and regulations, scheduling is often a difficult and time consuming task. Fortunately, the continuously growing computation power of PC's and the advances in database technology have opened up a treasure of opportunities to improve today's scheduling practices. This thesis deals with a number of methods that better exploit these opportunities. More specifically, we detect a number of challenging scheduling problems in hospitals, formulate these problems mathematically and develop algorithms that can efficiently solve them.

This thesis can be divided into three parts. The first part deals with staff scheduling. We propose a new formulation and decomposition approach for a problem that concerns building long term trainee schedules. The approach decomposes the problem on the activities and uses column generation to find an optimal solution. The resulting branch-and-price algorithm was embedded in a software application, some heuristic search procedures were added and it was tested on some real life instances of trainee scheduling problems. Our experimental results show an important increase in efficiency compared to the traditional approaches that decompose the problem on the staff members.

The second part of this thesis copes with operating room scheduling. First, we present a model and software tool for visualizing the usage of various resources as a function of the cyclic master surgery schedule. Next, we propose a number of models and algorithms to build surgery schedules with leveled resultant bed occu-

pancy. Our ideas have been tested on real life data by means of two case studies.

In the third part of this thesis, we combine the knowledge gained from the preceding two parts and present an integrated model for staff and operating room scheduling. The model is solved using a branch-and-price algorithm that repeatedly solves two different pricing problems. The first one involves the generation of the individual roster lines which is done using dynamic programming. In the second pricing problem, we search for a surgery schedule with a corresponding workload pattern that appropriately fits the generated set of roster lines. To this aim, a mixed integer programming model is solved. We have obtained some nice computational results for this difficult problem. To end, we show how our approach can be employed for benchmarking hospital units. Specifically, we illustrate how the results can be interpreted in order to identify the sources of waste in the hospital's human resource management.

Samenvatting

Het plannen van mensen en hulpmiddelen is een zeer belangrijk onderdeel in het operationele beleid van een ziekenhuis. Een weldoordachte planning kan verschillende financiële en sociale voordelen met zich meebrengen. Door de toegenomen druk op schaarse hulpmiddelen en de wildgroei in wetten en regelgevingen is het opstellen van een planning echter vaak een complexe en tijdrovende aangelegenheid. Gelukkig leiden de voortdurende toename in rekenkracht van computers en de technologische vorderingen op het vlak van gegevensbeheer tot een schat van mogelijkheden om het hedendaags plannen te verbeteren. Deze thesis handelt over een aantal methoden om deze mogelijkheden beter te exploiteren. Meer concreet detecteren we een aantal interessante planningsproblemen, we formuleren deze problemen wiskundig en ontwikkelen algoritmes om ze efficiënt op te lossen.

Deze thesis kan opgedeeld worden in drie delen. Het eerste deel richt zich op personeelsplanning. We introduceren een nieuwe formulering en ontledingsbenadering voor een probleem dat het opstellen van een assistentenplanning op lange termijn behelst. De benadering ontleedt het probleem op basis van de activiteiten en maakt gebruik van kolomgeneratie om een optimale oplossing te bekomen. Het resulterende vertak-en-prijs algoritme werd geïmplementeerd in een applicatie, aangevuld met heuristische zoekprocedures en getest op een aantal assistentplanningsproblemen uit de praktijk. Onze experimentele resultaten tonen een belangrijke toename in efficiëntie aan, vergeleken met de traditionele benaderingen die het probleem ontleden op basis van de stafleden.

Het tweede deel van deze thesis handelt over het plannen van het operatiekwartier. Eerst stellen we een model en computerprogramma voor om het gebruik van diverse hulpmiddelen in functie van de cyclische hoofdplanning van het operatie-

kwartier te visualizeren. Vervolgens stellen we een aantal modellen en algoritmes voor om planningen te genereren met een afgevlakte, resulterende bedbezetting. Onze ideeën werden getest op praktijkdata in twee gevalstudies.

In het derde deel combineren we de kennis verkregen uit de eerste twee delen in een geïntegreerd model voor de planning van het personeel en het operatiekwartier. Het model wordt opgelost door een vertak-en-prijs algoritme dat herhaaldelijk twee verschillende subproblemen oplost. Het eerste behelst het genereren van een individuele planning van een personeelslid d.m.v. dynamische programmering. In het tweede subprobleem zoeken we naar een planning van het operatiekwartier met een bijhorende, benodigde personeelsbezetting die goed past bij de gegenereerde set van individuele planningen. Dit gebeurt via het oplossen van een geheeltallig programmeringsprobleem. We hebben mooie rekenresultaten bekomen voor dit moeilijke probleem. Tenslotte tonen we aan hoe onze benadering gebruikt kan worden om verschillende ziekenhuizen te vergelijken. Concreet illustreren we hoe de resultaten geïnterpreteerd kunnen worden om de bronnen van verspilling in het personeelsbeleid van een ziekenhuis te detecteren.

Contents

Chapter 1

Introduction

This first chapter provides a general introduction to the material presented in this dissertation. This chapter is organized as follows. Section 1.1 gives an outline of the recent evolution in the health care expenses, in which we focus on the Belgian situation. As a motivation for this study, we discuss the importance of operations research techniques, in particular with respect to scheduling, as a useful tool to improve both the effectiveness and efficiency with which health care services are provided within hospitals. In Section 1.2, we justify our working method, explaining why we opted for certain approaches and why we left others out of consideration. Section 1.3 gives an outline of the most important mathematical programming techniques used in the algorithms developed for this study. Section 1.4 situates the chapters of this dissertation in a broader context, providing, in addition, two literature overviews: the first one on hospital staff scheduling and the second one on operating room scheduling. Finally, Section 1.5 summarizes the material that is presented in this chapter.

1.1 Motivation

Health care becomes very expensive. According to the 2005 report of RIZIV, the Belgian national expenses for health care amounted to 15.38 billion Euro in 2003. Five years earlier, in 1998, we spent no more than 11.29 billion Euro. In other words, the total health care expenses have increased by 36% in only five years. The annual figures indicate an average growth of 6.3% per year with a strong accelera-

tion of 8.2% in 2003 (RIZIV, 2005).

The significance of this rise becomes even more pronounced if one compares it with the growth of the Gross Domestic Product (GDP). Over the same period the annual growth percentages of the GDP fluctuated between 2.5 and 5.2%. Hence, the growth in health care expenses dramatically outpaces the GDP increase. Accordingly, the part of the GDP spent on health care has risen from 8.3% in 1998 to 9.6% in 2003 (OECD, 2005). Moreover, the differences between both growth figures continue to widen (Assuralia, 2005).

Equally significant is the fact that the health care expenses are invariably underestimated. In the period 1998-2003 an average annual deficit of 128 million Euro has been recorded (RIZIV, 2005). Also, it is a widespread belief that drugs make up the main cost, blaming the pharmaceutical industry for this trend. However, according to Assuralia (2005), only 16.5% of the total budget was spent on drugs, compared to 31% on hospital care.

Obviously, the main cause of this rise lies in the ageing of the society. Ageing populations are putting disproportionately heavy demands on health systems in high-income countries (Brandeau et al., 2004, p. 5). The continuous technological progression leads to new treatments that are often expensive and hence increase the pressure on the hospitals' budget. A possible way to keep the expenses at an acceptable level is to introduce more responsibility into the system. Principally, the Belgian health care system is free at the point of delivery and therefore neither the patients nor the care providers directly feel the real cost-price of health care. The problem of health care finance is not that the incomes are too low, but mainly that the expenses grow too fast.

Fortunately, there is also some good news. Compared to international statistics, the Belgian health care system performs actually quite well for its levels of cost and quality. For instance, in the US, health care spending amounts to 15% of the GDP in 2003, which is far above the 9.6% recorded for our country (OECD, 2005). Kumar and Ozdamar (2004) present an international comparison of health care systems involving a data envelope analysis on 19 industrialized countries. Data envelope analysis (DEA) is a mathematical evaluation method based on the concept of Pareto-optimal organization. The comparison is conducted under five classes:

health care expenditure, hospital care, physician services, pharmaceutical services, and life expectancy and infant mortality. The measurements selected for evaluating health care expenditure are percent of GDP, percent public spending, and per capita health spending. For hospital care, the measurements chosen are beds per one thousand population, percentage occupancy, expenditures per day and expenditures per admission. Similar measurements are selected for evaluating physician and pharmaceutical services. The Belgian health care system performs excellent for almost all these measurements and is hence top ranked in the DEA analysis under three of the five categories (health care expenditure, hospital services and pharmaceutical services).

In the near future, public resources for health care will become inadequate to meet the demand. Policy makers and health care providers must determine how to provide the most effective health care to citizens using the limited resources that are available. Therefore, they need effective methods for planning, prioritization and decision making. To this purpose, inspiration could be found in the field of Decision Support (DS), Artificial Intelligence (AI) and Operations Research (OR). OR techniques, tools and theories have long been applied to a wide range of issues and problems in traditional business, industrial and manufacturing environments. Fries (1976) present an early bibliographic overview of OR applications in health care systems that mainly deals with staff scheduling. More recently, Wiers (1997) gives a review on the applicability of AI and OR scheduling techniques in practice including health care applications.

In the April 2002 issue of ORMS Today Michael Carter (professor at the university of Toronto and CEO of the Health Care Productivity Research Laboratory) started the introduction of his article, in which he describes many possible applications for operations research in hospitals, as follows (Carter, 2002, p. 26):

> "Health care is the no. 1 domestic industry in the United States and one of the largest industries in the developed world. Health care systems present many complex problems that could benefit from operations research-type analysis and applications. OR professionals, however, have generally neglected the field."

Operations research techniques that have been shown to be successful in business environments could be applied on a variety of problems in health care environ-

ments. Alternatively, new operations research techniques could be developed for dealing with specific health care management problems. These include strategic planning problems such as design of services (e.g., inclusion of neonatal intensive care units in some hospitals), design of the health care supply chain (e.g., design of networks of hospitals, ambulance covering, outpatient clinics, drugs supply and laboratory services), facility planning and design (e.g., location and layout of hospitals), equipment evaluation and selection, process selection and capacity planning. Other planning problems include demand and capacity forecasting, job design, inventory management (e.g., drugs, supplies and blood) and scheduling and workforce planning (Brandeau et al., 2004, p. 8). This research addresses a variety of scheduling problems that occur inside hospitals.

Scheduling is a key issue for successful health care management. Scheduling involves the development of base plans for all types of resources within hospitals. Such base plans specify which resources to use at which time instances to serve which purposes. This research presents a number of exact and heuristic algorithms which provide practical solutions for a number of scheduling problems encountered at hospitals. The gain is manifold. First of all, since the algorithms assist in building the schedules, hospitals can save on (human) resources to do the job. Second, schedules generated by well-thought-out algorithms should be qualitatively 'better' than manually generated schedules. With 'better' we mean that those schedules result either in more output with the same input or in the same output with less input of resources or in a combination of both. Third, better scheduling practices might result in social benefits too like shorter waiting lists (see, e.g., Vissers et al., 2001; Buhaug, 2002; Mullen, 2003). Also work in Psychology has shown that better personnel schedules can have an impact on nurses' well being and job satisfaction (Mueller and McCloskey, 1990; Oldenkamp, 1992) and lead to safer working environments (Wilkinson and Allison, 1989; Folkard and Tucker, 2003).

1.2 Approach

This dissertation contains both fundamental and applied research as well as both exact and heuristic solution approaches. This section explains what we mean by this.

1.2.1 Fundamental and applied research

The research presented in this work is both fundamental and applied. The study is technical in nature: we will focus on problem formulations and algorithms. Each scheduling problem will be defined formally, making abstraction of some case specific issues. This involves the formulation of the problem in a mathematical way. As such, we try to generalize the problem as much as possible in order to ensure that the developed solution approach can be applied in as many practical situations as possible. Furthermore, a formal, general problem formulation can help us to derive some important theoretical properties such as computational complexity and enables us to reduce (parts of) the problem to already solved problems in the literature. Throughout this work, the efficiency of the algorithms is a main issue. Accordingly, all the algorithms are extensively tested. Tests generally include a comparison with other solution approaches, an indication on how problem dimensions influence computation times and an overview of the (computational) contributions of the different algorithmic features. This technically oriented approach has, however, not prevented us from being inspired by and tackling real-life scheduling problems in hospitals.

The concern about the applicability of the algorithms has been at least equal to the attention given to their computational efficiency. Therefore, all the proposed algorithms, with the exception of the integrated scheduling approach of Chapter 6, have been motivated by and tested on real-life data. Again with the exception of Chapter 6, all our algorithms have been implemented in a self written software application with a graphical user interface that could easily be used by both experienced and non-experienced schedulers.

1.2.2 Exact and heuristic algorithms

The difference between exact and heuristic algorithms is guaranteeing optimality. Exact algorithms, also called optimal procedures, can guarantee the optimality of a solution for a given problem, which requires some built-in mechanism for optimality proving. In contrast, heuristic algorithms, also called suboptimal procedures or, simply, heuristics, are designed to find the best possible solution with small computational efforts. Heuristics often lack a mechanism of optimality proving and, therefore, fail to provide any information on the quality of the solution other than

that it is the best solution found.

When it comes to solving practical problems in real-life situations, heuristics are undoubtedly the best choice. If the problem space is very large, which is often the case for real-life problems, heuristics are the only option, because exact algorithms would require endless computation times. Moreover, practitioners are rarely bothered by the optimality of a solution: in real-life, typically all what matters is to have a reasonable solution as fast as possible.

Exact algorithms, on the other hand, are more appropriate to develop fundamental insights into the mathematical complexity of the problem as well as into the managerial aspects of the problem. The last is probably best illustrated in Chapter 6, in which we present a managerial insight gained from developing an exact algorithm for integrating the nurse and surgery scheduling process. Another example is the search to lower bounds in minimization problems (or upper bounds in maximization problems) to cut off large parts of the solution space in order to prove the optimality of a solution. Not rarely such a bound or bounding rule is inspired by the real-life problem, but it also happens that a theoretically found bound or bounding rule, obtained through careful analysis of the problem structure to develop an exact procedure, results in a new managerial insight or at least a confirmation of existing insights. A last example of how managerial insights can be gained as side results from the application of optimal procedures is the use of the dual prices to do a sensitivity analysis in Linear Programming (LP).

This study copes with both exact and heuristic algorithms. It will, however, soon become clear that the focus lies on exact algorithms. All the problems studied are first approached using an exact solution procedure. If the exact algorithm turns out to be too time consuming for solving real-life problems - and it usually does - our attention shifts towards heuristic procedures. In many cases, the exact algorithm can easily be turned into a heuristic by modifying certain subprocedures of the algorithm. An alternative way of viewing this, is that we tolerate an optimality gap in these subprocedures. The main benefit of this approach is that the resulting exact/heuristic hybrid algorithm still provides some information on how far the solution is from the theoretical optimum.

The reason why, in this dissertation, exact algorithms come more into the spotlights than heuristics, is twofold. First of all, it is my conviction that a researcher is obliged, at least at first instance, to investigate whether or not an exact algorithm can be developed for a given problem rather than immediately starting experimenting with different kind of heuristics, which often involves a lot of parameter setting and testing. It seems to me more relevant to first study the structure of the problem, find out what complicates it, discover some important properties and use this knowledge to develop an exact procedure that could solve the problem to optimality, if necessary using smaller problem dimensions. This way of approaching a problem is in my view the best guarantee to gather fundamental insights into the basics of the problem. One will never fully understand the complexity of a problem unless one has explored the borders of optimality. This research is not only motivated by problem solving, but also by the development of new techniques. At one side of the spectrum you have the practitioner who's only concern is to solve the problem at hand in a reasonable way as soon as possible. At the other side there is the researcher who studies the problem, tries to discover certain properties and develops solution procedures that are not only useful for the problem at hand, but also can inspire other researchers studying similar problems and as such, make a contribution to the global knowledge of mankind. Without underestimating the important contribution of the basic heuristic methodologies like tabu search, simulated annealing and genetic algorithms, in my personal view, exact procedures are better suited to serve the last purpose. The second reason for the extensive attention given to exact procedures is a subjective one. Being capable of detecting the optimal solution out of millions or billions of alternatives and, above all, proving its optimality, is really fascinating to me. The resulting challenge makes the study, development and coding of exact algorithms most exciting.

1.3 Mathematical programming foundations

This section provides a short introduction to the most important mathematical programming techniques involved in the algorithms presented later in this work. For an in-depth analysis of these techniques, the interested reader is each time referred to a number of basic works and/or survey papers.

1.3.1 Linear programming

Linear programming (LP) is an optimization technique in which a problem is formulated as a linear function that has to be optimized, i.e., minimized or maximized, subject to a set of linear constraints. To build such a formulation, one first translates the real-life decisions into a set of decision variables. Next, the real-life objective is stated as a linear function of these decision variables, called the objective function. Finally, the real-life constraints are also expressed linearly in the decision variables. Let us illustrate this with a simple example:

> Pharma Inc. wants to produce a new drug in the cheapest way possible. The drug has a specified weight w expressed in grams and consists of three ingredients. Each ingredient has a certain cost per gram, say c_1 for ingredient 1, c_2 for ingredient 2 and c_3 for ingredient 3. In order to make the drug effective, research has indicated that the share of ingredient 1 should not exceed twice the share of ingredient 2. Furthermore, the summed shares of ingredient 2 and 3 should exceed a target level of l_1 grams. Finally, the summed shares of ingredient 1 and 3 may not surpass a target level of l_2 grams.

To formulate this problem as a linear program, we first identify the decision variables. Since Pharma Inc. has to decide upon the share of the three ingredients, we define x_1, x_2 and x_3 as the respective shares of each ingredient in the final drug. The problem can then be stated as follows:

$$\text{Minimize } c_1x_1 + c_2x_2 + c_3x_3 \tag{1.1}$$

subject to:

$$x_1 + x_2 + x_3 - w = 0 \tag{1.2}$$
$$x_1 - 2x_2 \leq 0 \tag{1.3}$$
$$x_2 + x_3 - l_1 \geq 0 \tag{1.4}$$
$$x_1 + x_3 - l_2 \leq 0 \tag{1.5}$$
$$x_1, x_2, x_3 \geq 0 \tag{1.6}$$

The objective function (1.1) is simply the minimization of total costs. Constraint (1.2) makes sure the drug weights w grams. Constraint (1.3) ensures that the share

of ingredient 1 does not exceed twice the share of ingredient 2. Constraints (1.4) and (1.5) imply the target level share restrictions. Finally, restrictions (1.6) are the nonnegativity constraints.

To solve LP problems, several highly efficient procedures have been developed, of which the most well-known are the primal and dual simplex algorithm and the interior-point method. The efficiency combined with the generality of the approach has made linear programming a real success story. Since the late fifties, linear programming has become a well-established tool for solving a wide range of optimization problems in various fields production, network design, manufacturing, routing, engineering and scheduling. This trend continues with the developments in modeling, algorithms, the growing computational power of personal computers and the increasing performance of commercial linear programming solvers (Johnson et al., 2000). Unfortunately, many real-life decisions, particularly in the field of scheduling, are modeled using variables that are required to be integral; for instance, you can assign 1, 2, 3,. . . , nurses to work a particular shift, but not 0.5, 1.5, 2.5 nurses. Solving these so-called mixed integer problems is generally much harder (see Section 1.3.3) and the efficient LP procedures mentioned earlier can only be used to solve a relaxation of the problem.

Chvátal (1983) provides an in-depth analysis on linear programming. Winston (1993) gives an excellent introduction to the general field of operations research, explaining linear programming in detail. The textbook of Williams (1999) includes many examples of how to design mathematical models, including linear programming formulations.

1.3.2 Quadratic programming

A quadratic program (QP) is a variant of a linear program in which the objective function contains quadratic terms. Quadratic programming problems are frequently encountered in finance, more specifically in portfolio selection applications, in which one would like to spread the risk as much as possible (for a basic reference, see Markowitz, 1959). Resource leveling is another typical application of quadratic programming. For further reading we refer to Gill et al. (1982) and Winston (1993) who cover, among other topics, quadratic programming. In this study, quadratic programming problems only show up in Chapter 4 and Chapter 5 where

we respectively develop and apply an operating room scheduling model that aims at leveling the resultant bed occupancy.

1.3.3 Integer programming and branch-and-bound

If some of the decision variables are required to be integral, the problem becomes a mixed integer program (MIP). This is the case for many practical applications, including most scheduling problems. Unfortunately, the introduction of integral variables often seriously complicates the problem, making it extremely hard to solve. When an integer variable is restricted to the values 0 or 1, it is called a binary variable. Mixed integer programming problems are usually solved using a branch-and-bound approach. Branch-and-bound is an implicit enumeration approach, which, as the name suggests, succeeds in detecting the optimal solution without having to explicitly consider (enumerate) all solutions.

The enumeration scheme involves a stepwise partitioning of the solution space. In the scheme each partition is further analyzed and partitioned until a (better) feasible solution is found or it is determined that the partition does not contain a better solution. The enumeration scheme can often nicely be represented by a branching tree (see, e.g., Figure 1.1).

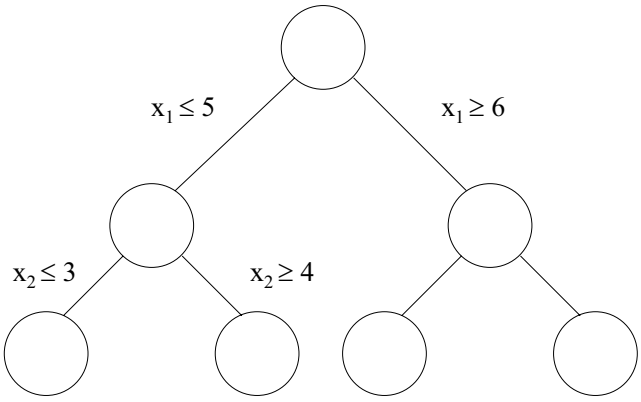

Figure 1.1: Branch-and-bound tree example

The root node of the tree represents the original problem, and has a solution space that holds all feasible solutions to this problem. The first step entails the partitioning of this solution space into two or more subsets represented by two or more child nodes. This process of partitioning the solution space, which is called branching, is continued in the child nodes, which become parent nodes to new child nodes in a subsequent partitioning of the solution space. Consider now a particular node in the tree. Suppose we are able to detect the best solution in each child of this node. The best one amongst these solutions is obviously also the best solution for the parent node. If we continue in this way bottom up until the root node is reached, the problem has been solved to optimality. It may even not be required to detect the best solution in each child node. Indeed, whenever we know that the best solution in a particular node will be worse than a feasible solution of an (already explored) node anyway, this node does not have to be considered or further partitioned. Not explicitly considering a node refers to the bounding aspect of the branch-and-bound methodology.

The question remains of course how to efficiently partition the solution space in order to take full advantage of this divide-and-conqueror strategy. This depends on the problem at hand. In any case, two conflicting criteria are important in designing an efficient branching scheme. On the one hand, the number of generated nodes is best kept as small as possible. On the other hand, the nodes that are not further partitioned should be easily solvable.

In integer programming applications the branch-and-bound algorithm involves a standard branching scheme. In the root node a relaxation of the original problem is being solved. A relaxation is a simplification of the problem, for instance by leaving out one or more of the constraints. In linear integer programming problems, the root node typically corresponds to the linear relaxation, i.e., the problem without the integrality constraints. As the algorithm progresses, the solution space is partitioned by adding constraints to the problem in the root node, forming two or more child nodes. Such a constraint typically cuts away a current fractional solution, by implying a fractional variable either to be larger than or equal to the first upper integer or to be smaller than or equal to the first integer smaller than the current fractional value (see, e.g., Figure 1.1).

Each node in the branch-and-bound tree is associated with a lower and an upper bound, which are used to fathom certain nodes from further consideration. In minimization (maximization) problems the lower (upper) bound represents the theoretically best possible solution value that could be found by further exploring the node, while the upper (lower) bound is the best solution value which has so far been shown to exist. In minimization problems a node can be fathomed, if the lower bound in that node is larger than or equal to the current upper bound. Obviously, the reverse applies in maximization problems. Branch-and-bound methods have been successfully applied in a wide range of optimization problems including knapsack problems (e.g., Kellerer et al., 2004) and resource constrained project scheduling problems (e.g., Demeulemeester and Herroelen, 1992; Vanhoucke, 2001).

The recently appeared tutorial survey of search methodologies edited by Burke and Kendall (2005) contains a comprehensive chapter on integer programming written by Bob Bosch and Michael Trick. Widely cited references on integer programming include Winston (1993), Wolsey (1998) and Nemhauser and Wolsey (1999). These books explain the branch-and-bound algorithm in detail. In the description of the branch-and-bound methodology outlined above we basically followed the exposition of Hans (2001). Agin (1966) and Johnson (2000) also provide an excellent discussion on the fundamentals of branch-and-bound.

Almost all problems dealt with in this study will be formulated as integer programs. However, only the operating room scheduling problems (extensively studied in Chapter 4 and Chapter 5 and appearing as a subproblem in Chapter 6) are actually being solved using standard integer programming techniques. The staff scheduling problems, on the other hand, are formulated using a huge number of decision variables. Such formulations involving many variables entail several benefits, most importantly a tighter bound relaxation and elimination of symmetry (for more details, see Barnhart et al., 1998), but cannot be efficiently solved with a standard, commercial MIP solver. Therefore, column generation is used to solve the LP relaxations of these mixed integer programming problems.

1.3.4 Column generation and branch-and-price

If the number of decision variables in an LP problem is (exponentially) large, say, in the order of one million variables or more, column generation can significantly

speed up the optimization process. Hans (2001) gives a short, although most understandable exposition of the column generation technique. In this technique the LP is solved to optimality by first considering a restricted LP (RLP), where many variables are left out. The RLP is also called the restricted master problem. The solution of the RLP is optimal for the LP if all variables of the LP have non-negative reduced cost. Since only a subset of the variables of the LP is explicitly available, this cannot be checked explicitly. A so-called pricing algorithm is used to verify optimality. This algorithm solves a pricing problem so as to find a new variable that could improve the current LP solution. Such an improving variable is called to price out. If the current solution is not optimal, the pricing algorithm identifies at least one variable with non-negative reduced cost. This variable is added to the RLP and the procedure continues. The column generation scheme terminates when no variables with negative reduced cost exist anymore. At that point, the LP is solved to optimality. The decision variables are often referred to as columns, which explains the name column generation.

Column generation is especially appropriate for solving LP problems involving a huge set of variables of which most of them will have their associated variable equal to zero in an optimal solution anyway. Dantzig and Wolfe (1960) were the first to propose the column generation technique. Since then it has been applied to a wide variety of problems. The most well-known is the application to the stock cutting problem (Gilmore and Gomory, 1961; Peeters, 2002). Other applications in which column generation turned out to be particularly fruitful, include, e.g., vehicle routing problems with time windows (Desrosiers et al., 1984; Desrochers et al., 1992) crew scheduling and rostering (Vance et al., 1997; Gamache et al., 1999) and capacitated lot sizing (Jans, 2002).

Column generation is referred to as branch-and-price when it is used to solve the LP relaxation in every node of a branch-and-bound tree originating from solving a mixed integer linear programming problem. Hence, branch-and-price is basically a combination of branch-and-bound and column generation. There are, however, fundamental difficulties in applying column generation techniques for linear programming in integer programming solution methods (Johnson, 1989). First of all, conventional integer programming branching on variables (as illustrated in Figure 1.1) may not be effective because of symmetry problems and because fixing variables can destroy the structure of the pricing problem. Second, the LPs are often

not needed to be solved to optimality in each node of the branch-and-bound tree. A careful inclusion of rules for managing the branch-and-price tree can significantly speed up the branch-and-price algorithm.

Barnhart et al. (1998) present a general methodology for branch-and-price which unifies the existing literature. The book by Wolsey (1998) also contains a chapter on the use of column generation for solving integer programming problems.

Column generation techniques will be extensively applied throughout this thesis for solving staff scheduling problems encountered in hospitals. The pricing problems often involve the solution of a restricted shortest path problem, which will be optimized using dynamic programming.

1.3.5 Dynamic programming

Dynamic programming is another basic mathematical programming technique that is frequently applied in this thesis. The following exposition is largely based on the brief, but excellent description of dynamic programming by Hans (2001). For more extensive expositions we refer to Winston (1993) and Wolsey (1998) who both cover dynamic programming in a separate chapter and to Dreyfus and Law (1977), who discuss the fundamental background of dynamic programming in detail. Finally, when it comes to the implementation details, Sedgewick (1998) provides an excellent guide to the efficient coding of dynamic programming algorithms.

Dynamic programming (DP) is a decomposition technique that first decomposes the problem into a nested family of subproblems. One can distinguish between deterministic and probabilistic dynamic programming problems. In probabilistic or stochastic DP the decisions have a stochastic outcome, and the goal is to determine the decisions that minimize the expected cost (or maximize the expected reward), while in deterministic DP all decision variables yield a deterministic contribution to the objective. Since only deterministic DP is used in this work, we only discuss this category of dynamic programming. For literature concerning probabilistic DP we refer to Ross (1983) and Sennott (1999).

A typical dynamic programming application includes the following five characteristics:

1. The problem can be divided into a number of stages t, with a decision x_t required at each stage.

2. Each stage t has a set of states $\{i_t\}$ associated with it. At any stage a state holds all the information that is needed to make a decision.

3. The decision taken at any stage determines how the state at the current stage is transformed into the state at the next stage, as well as the immediately earned reward or cost.

4. Given the current state, the optimal decision for each of the remaining stages must not depend on previously reached states or previously taken decisions. This is the so-called principle of optimality for dynamic programming (Bellman, 1957).

5. If the states for the problem have been classified into one of T stages, there must be a recursion that relates the cost or reward earned during stages $t, t+1, \ldots, T$ to the cost or reward earned from stages $t+1, t+2, \ldots, T$.

In a forward DP algorithm, the recursion mentioned in the fifth characteristic can often be written as:

$$F_t(i_t) = \underset{x_t \in S^t}{\text{MIN}} \Big\{ c_t(i_t, x_t) + F_{t-1}(i_{t-1}(i_t, x_t)) \Big\}, \tag{1.7}$$

where S^t is the set of possible decisions x_t in stage t, where $c_t(i_t, x_t)$ is the cost (or reward in a maximization problem) function that returns the cost for moving from state $i_{t-1}(i_t, x_t)$ to state i_t according to decision x_t, where $i_{t-1}(i_t, x_t)$ is the state from which i_t is reached, given decision x_t, and where $F_t(i_t)$ is the total minimum cost (or maximum reward in a maximization problem) incurred from stage 1 to stage t given state i_t in stage t.

In forward DP it is assumed that the desired state we want the system to be in, in stage T (call it i_T), can be specified. An optimal solution requires the identification of an optimal set of decisions, one for each stage t. The algorithm to find these decisions first applies a forward calculation pass, followed by a backward decision determination pass. In the forward calculation pass, we first compute the $F_1(i_1)$ for all possible states in stage 1. We then apply (1.7) to calculate the $F_2(i_2)$'s in terms of the $F_1(i_1)$'s, and continue in this fashion until $F_T(i_T)$ has been reached.

At this moment the forward calculation pass has been terminated. The backward pass then starts with the determination of the optimal decision in stage T that attains $F_T(i_T)$. This decision in turn determines a state i_{T-1} in stage $T-1$ from which we arrive to state i_T during the last stage. We then determine the optimal stage $T-1$ decision, which in turn determines a state i_{T-2} in stage $T-2$, and continue until the optimal stage 1 decision is found.

1.4 Situating the chapters in a broader context

For a good understanding of the material presented hereafter, it is necessary to place it in a larger context. As an additional benefit, the reader obtains a better view on the link between the different chapters. Basically, the considered problems fall into either the field of staff scheduling or into the field of operating room scheduling or into a combination of both.

1.4.1 Staff scheduling

Humans undoubtedly make up the most important resource employed in hospitals. Many operational scheduling problems encountered at hospitals can be classified as staff scheduling problems. The fact that personnel scheduling in hospitals is subject to specific constraints compared to other service organizations (like for instance round the clock scheduling) makes the development of good staff schedules a challenging issue. The first and most famous problem concerns the scheduling of nursing personnel. In the nurse scheduling problem one has to determine when nurses have to be present, taking into account a variety of hard/soft constraints such as legal regulations, nurses' preferences, minimal coverage requirements, personnel policies and many other restrictions that may be hospital specific.

The classic nurse scheduling problem (NSP) consists of generating a configuration of individual schedules over a given time horizon in order to meet hospital staffing demand. An individual's *roster line* can be viewed as a sequence of *days on* and *days off*, where each day on contains a single *shift* identified by a label such as 'day', 'evening' or 'night'. Each such label coincides with a start and a finish time of the corresponding shift. Furthermore, a day is subdivided into several *demand periods*, indicating how many nurses are required to cover the work. These demand

periods often (but not necessarily) coincide with the shifts. The individual roster lines are subject to a large number of constraints, further referred to as collective agreement requirements. Examples of such constraints include shift transition constraints (e.g., a night shift cannot be followed by a morning shift), total workload limitations, restrictions on the number of weekend shifts, holidays, etc.

A second important staff scheduling problem concerns the scheduling of so-called trainees. Trainees are graduated students that wish to specialize further in a specific field of health care. The scheduling of trainees can be seen as a specific case of nurse scheduling since it involves many similar constraints such as coverage requirements and staff preferences. Trainees differ however from other nursing staff in that they still have to complete an education. This education requires that they have to perform a number of activities in a given time horizon. Such activities include amongst others assisting during surgery, performing consultation or being standby for emergency cases. **Chapter 2** discusses in detail the solution approach for solving a trainee scheduling problem. Although motivated by and tested on a specific real-life case, the proposed solution method can easily be generalized and is hence applicable in many situations.

Next to nurses and trainees, hospitals encounter several other types of staff scheduling problems such as the scheduling of physiotherapists (e.g., Carter and Lapierre, 1999), anaesthetists (e.g., Dexter and Traub, 2000) or scheduling administrative personnel.

1.4.2 Operating room scheduling

A critical resource in each hospital is the operating room. As pointed out by Litvak and Long (2000), the operating room can be seen as the engine of the hospital. Indeed, the activities inside the operating room have a dramatic impact on many other activities within hospitals. For instance, patients undergoing an operation are expected to recover during a number of days. Consequently, bed capacity and nursing staff requirements are dependent on the operating room schedule. By well-thought-out scheduling of the operating room, the expected variability in resource demand can be minimized.

Variability has a very negative impact on productivity and reducing it is one of the major concerns of health care management. One can distinguish between two types of variability: natural variability and artificial variability. Natural variability is inherent to the uncertain world of health care. This variability arises from uncertainty in patient show-ups (e.g., emergency cases), uncertainty in recovery time, uncertainty in the successfulness of therapies etc. Artificial variability originates from poor scheduling policies. A poor operating room schedule could for instance directly be responsible for a shortage in beds each Wednesday, whereas there is overcapacity on all other days of the week. Exact and/or heuristic algorithms can assist in minimizing artificial variability. Although natural variability is by definition uncontrollable, its negative consequences can be minimized by developing algorithms which aim at producing schedules with leveled resource uses.

Summarizing, what happens inside the operating room determines the demand for various resources throughout the rest of the hospital. Altering the number of hours preserved for certain ailments leads to important changes in the absolute demand for several resources. The demand for services nearly always exceeds the available capacity. As a result, an important question encountered at each hospital is which services it should provide. In other words, what is the optimal case mix? In the long term, this case mix determines for which ailments capacity will be preserved. Obviously, case mix decisions depend largely on the funding of the hospital, as indicated by Blake and Carter (2002), and these decisions have a large impact on the quality of service. For instance, if the share of a certain ailment in the total case mix decreases, patients suffering from this ailment will be confronted with longer waiting times. In this dissertation, case mix decision problems are left out of consideration. In other words, the absolute number of hours of operating room time, allocated to each surgical group, is assumed to be fixed.

The absolute number of hours, allocated to each surgical group, is only one side of the story though. At least of equal importance is the timing of the resource needs. It is the operating room schedule that largely determines the time dimension of the demand for resources. **Chapter 3** presents a computer program that visualizes the load of various resources as a function of the operating room schedule. The underlying model, on which the graphical user interface has been built, is very basic for the resource consumption patterns are assumed to be deterministic. Moreover, the software merely serves as a visualization module. It does not include automation

procedures, neither for the operating room scheduling, nor for those resources for which scheduling by itself is already fairly complex (like nurses). Nonetheless, this chapter is a good introduction for the following two chapters. By means of a case study, the chapter provides a non-exhaustive overview of the key resources used in hospitals. The most important amongst those are elaborated in the later chapters. In **Chapter 4** we develop a number of surgery scheduling algorithms that aim at leveling the resultant bed occupancy. Subsequently, **Chapter 5** describes a real-life application of the models and algorithms developed in the preceding chapter.

If one mentions beds as an important resource to be considered when scheduling the operating room, one implicitly refers to staffed beds. Indeed, a bed considered as a physical unit by itself, is, although sometimes very sophisticated, relatively cheap. On the contrary, a staffed bed is expensive and consequently much more limited in capacity. We define a staffed bed as a bed for which the required care for the occupying patient is guaranteed at each time instance by sufficient availability of correctly skilled staff. Staff scheduling problems have already been introduced in Section 1.4.1. As already mentioned above, Chapter 2 treats a specific staff scheduling problem. **Chapter 6** presents a model as well as a solution procedure to integrate the operating room and the nurse scheduling process. The algorithmic procedure was developed using knowledge gained from the staff scheduling field as well as from the operating room scheduling field, combining several building blocks into a beautiful integrated approach. As such, this chapter links the preceding chapters.

1.4.3 Nurse scheduling literature review

In general nurse scheduling different approaches exist for various time horizons. In many literature overviews the nurse scheduling problem is therefore decomposed into different phases (3 phases in Warner, 1976a; Bradley and Martin, 1990; Sitompul and Randhawa, 1990 and 5 phases in Tien and Kamiyama, 1982). Often a distinction is made between staffing or planning and scheduling or rostering. Staffing refers to longer term personnel planning and copes with decisions on the number of hired nurses of the required skills in order to meet predicted demand, defining work agreements for part time workers, deciding whether substitution of skill categories is allowed, etc. Scheduling or rostering, on the other hand, refers

to the short term timetabling of staff (with a typical time horizon of a few weeks). Since staffing and scheduling takes place on different levels and for different time horizons, it would be unworkable in practice to handle them simultaneously all the time. Nevertheless, as staffing determines the input for scheduling, interaction between the levels is certainly necessary. Venkataraman and Brusco (1996) even present a completely integrated nurse staffing and scheduling system.

Recently, a brief but well-structured review on nurse scheduling is provided by Cheang et al. (2003). Very recently, Burke et al. (2004) present a more extensive and excellent survey that mainly copes with nurse rostering rather than staffing. Ernst et al. (2004) and Blöchliger (2004) present a very comprehensive overview of the literature on staff scheduling and rostering that they describe in general rather than concentrating on nurse scheduling in particular. Both reviews, however, contain a category that specifically discusses health care systems and is mainly concerned with nurse scheduling.

The paper by Warner et al. (1991) on patient-oriented and employee-oriented issues in nurse management contains a description of the history of computerized nurse scheduling in the United States. Also, Siferd and Benton (1992) provide an overview in which they specifically deal with the factors influencing hospital staffing and scheduling in the US. Ikegami and Niwa (2003) consider nurse scheduling problems in Japan for which rapid shift transitions are very common, i.e., nurses work different shifts per week. Silvestro and Silvestro (2000) discuss the results of a survey of nurse rostering practices in the UK National Health Service. For which concerns the Belgian situation, excellent work has been done by Burke et al. (1998, 1999, 2001, 2003 and 2004) and De Causmaecker and Vanden Berghe (2003). They developed a general model for the nurse rostering problems and refer to it as Advanced Nurse Rostering Model (ANROM). The problem dealt with in their model is situated at the short-term timetabling level. Its main objective is to understand and automatically generate comfortable shift schedules for personnel members in order to meet the staff coverage while an extensive set of realistic constraints is captured and integrated, together with explicit and implicit objectives, in a general, flexible model. A more detailed analysis of the model and solution framework can also be found in Burke et al. (2001a, 2001b, 2002 and 2006). A software package based on the model and the solution framework was first implemented in 1995 but the system is still evolving to cope with the new and more complex real-world

problems that keep appearing. So far, over 40 hospitals in Belgium have replaced their time consuming manual scheduling by this system.

Several studies in the literature have utilized mathematical programming techniques to assist in finding efficient staff schedules (see, e.g., Warner and Prawda, 1972; Abernathy et al., 1973; Warner, 1976b; Miller et al., 1976; Bailey and Field, 1985; Rosenbloom and Goertzen, 1987; Beaumont, 1997; Millar and Kiragu, 1998; Bard et al., 2003; Isken, 2004). Mathematical programming approaches have often been used when multiple objectives are considered (see, e.g., Arthur and Ravindran, 1981; Ozkarahan and Bailey, 1988; Franz et al., 1989). These so-called goal programming models allow for more flexibility to relative ranking assigned to various objectives by defining target levels for different criterions and relative priorities to achieve these goals (e.g., Azaiez and Sharif, 2005). Ozkarahan (1989) and Chen and Yeung (1993) combine goal programming with expert systems (see further). More recently, also heuristic procedures have been proposed for multiple objective nurse scheduling (Berrada et al., 1996; Jaszkiewicz, 1997; Burke et al., 2002).

The main problem of integer programs lies in the large computation times needed for many practical instances, even to obtain just a feasible solution. To overcome this problem, heuristic approaches and techniques originating from the artificial intelligence field have been developed and successfully applied on various nurse scheduling problems. For specific problems, however, exact approaches that exploit specific features of the problem structure can suffice to obtain reasonably small computation times and as such form an alternative for standard integer programming techniques. We distinguish between branch-and-bound and branch-and-price approaches. Bosi and Milano (2001) combine constraint logic programming with branch-and-bound techniques for scheduling problems. Trivedi and Warner (1976) provide another example of a branch-and-bound algorithm that applies on the short-term assignment of so-called float nurses (nurses from other units) whenever there is a shortage of nurses in a particular unit. Ikegami and Niwa (2003) present a branch-and-bound algorithm extended with heuristic search that generates very promising results for 2-shift and 3-shift problems.

Examples of branch-and-price approaches for solving general staff scheduling problems can be found in Mehrotra et al. (2000) and Caprara et al. (2003). Papers involving branch-and-price techniques that specifically deal with nurse scheduling

problems include Jaumard et al. (1998), Mason and Smith (1998) and Bard and Purnomo (2005a) and (2005b).

Artificial intelligence techniques focus on finding feasible solutions rather than optimizing a particular objective function. We distinguish between declarative and constraint programming on the one hand and knowledge-based and expert systems on the other hand. Constraint satisfaction models have been proposed by Okada (1988), Okada and Okada (1992), Darmoni et al. (1995), Weil et al. (1995), Cheng et al. (1997), Abdennadher and Schlenker (1999) and Musliu et al. (2000). As it is often not possible in real-life applications to satisfy all the constraints anyway, Meyer auf'm Hofe (1997) present a hierarchical constraint satisfaction model built on a library of search algorithms and constraint propagation techniques. In an advanced model of the same author (Meyer auf'm Hofe, 2001) fuzzy constraints are introduced that can be partially violated and partially satisfied in a constraint optimization framework. Constraint programming approaches are often combined with other procedures. For instance, Meisels et al. (1996) and (1998) combine constraint networks and knowledge-based rules to solve employee timetabling problems and Li et al. (2003) apply a hybrid constraint satisfaction/local search technique for building personal schedules.

Expert system approaches provide the possibility for developing user-interactive, integrated (staffing, rostering) decision support methodologies for nurse scheduling problems (Nutt, 1984). Smith et al. (1979) developed an interactive 'what-if' decision support system that allows users to assign and modify weights to different objectives and to take personal preferences into account. Bell et al. (1986) present a rather basic visual interactive decision support system for workforce scheduling. As already mentioned earlier, Ozkarahan (1989) and Chen and Yeung (1993) combine expert systems with goal programming. Case-based reasoning models have been developed by Scott and Simpson (1998) and Petrovic et al. (2003). They attempt to generate good quality solutions by mimicking the human style of reasoning in existing manual rostering practices.

Heuristic procedures form another alternative to cope with large solution spaces and hence long computation times. We distinguish between simple heuristic procedures and metaheuristics. Simple heuristics mimic the trial-and-error manner that the planner employs to construct the schedule by hand. Early contributions on

interactive heuristic procedures are presented by Smith (1976) to construct cyclical schedules and by Smith and Wiggins (1977) for non-cyclical schedules. The work of Blau (1985) is one of the earlier attempts to evenly treat personnel with respect to workload and shift preferences. Anzai and Miura (1987) present a cyclic descent algorithm for rostering problems with identical staff members. Blau and Sear (1983) and Kostreva and Jennings (1991) solve the nurse scheduling problem in two phases. Blau and Sear generate feasible shift patterns in a first step and use a cyclic descent algorithm to assign a shift pattern to each nurse in order to obtain an overall optimal schedule. The first step in Kostreva and Jennings (1991) involves the generation of groups of feasible schedules that respect the minimum staffing requirements. In the second step, the best possible solution, which is based on the individual preferences, is calculated. Schaerf and Meisels (2000) present hill climbing algorithms for general local search that allows partial assignments and thus provides more flexibility towards satisfying coverage constraints.

Metaheuristic procedures are more general solution approaches since these techniques are less problem dependent and hence can be applied to almost any kind of combinatorial optimization problem. The most important metaheuristic approaches include simulated annealing, tabu search and genetic algorithms.
Examples of simulated annealing approaches can be found in Isken and Hancock (1991) and Brusco and Jacobs (1993). The model presented by Isken and Hancock provides an original contribution as it does not assume fixed shifts but incorporates flexible hours which is very common in many modern hospitals and allows for more flexibility in staffing coverage. Brusco and Jacobs combine simulated annealing with a simple local search heuristic to construct cyclical schedules for continuously operating organizations (24 hours per day, 7 days per week) like hospitals, public safety and telecommunication companies. For a short exposition on the basics of the simulated annealing metaheuristic we refer the reader to Chapter 4, more specifically to Section 4.4.4, in which we outline a simulated annealing procedure for building a cyclical surgery schedule with leveled resultant bed occupancy.

Many modern attempts to solve complex scheduling problems involve tabu search techniques. The originality of Dowsland's (1998) contribution lies in the fact that the search oscillates between feasible and non-feasible regions whereas other approaches tend to avoid infeasibility. Often tabu search is combined with another search technique which results in a hybrid approach. Berrada et al. (1993), for in-

stance, apply tabu search instead of mathematical programming in a multi-objective framework. Burke et al. (1998) and (1999) hybridize a tabu search approach with improvement techniques that simulate human reasoning. More details on the variable neighborhood search approach can be found in Burke et al. (2003) and De Causmacker and Vanden Berghe (2003). A general overview of this work is presented in Burke et al. (2004). Valouxis and Housos (2000) and Dowsland and Thompson (2000) integrate tabu search in an integer programming model. Ikegami and Niwa (2003) first decompose the problem into subproblems in which all but one of the nurses' schedules are fixed. Then, a tabu search algorithm tries to repeatedly satisfy the constraints on different subproblems. For this algorithm, they only report results for 2-shift problems. However, this work also contains a branch-and-bound method (see further) that has also been applied on 3-shift problems. Bellanti et al. (2004) developed a tabu search method as well as an iterated local search approach for solving a particular rostering problem in an Italian hospital.

Genetic algorithms make up the last important metaheuristic that we discuss in this survey. Easton and Mansour (1993) and Tanomaru (1995) were among the first to propose a genetic algorithm for employee staffing and scheduling, however their models are possibly too simple for real-life applications. Aickelin and Dowsland (2000) use problem specific knowledge to guide the crossover operator and extend their genetic algorithm with a hill-climbing operator. They decompose the problem into easier to solve subproblems by taking advantage of the fact that night and day shifts are preferably not combined in a one-week nurse schedule and higher skilled nurses can replace lower qualified nurses but not vice versa. The work by Aickelin and Dowsland (2004) deals with the same nurse rostering problem as in Aickelin and Dowsland (2000), but presents an indirect genetic algorithm with a separate heuristic decoder instead of working with a direct representation of the schedules. Also Ackelin and White (2004) tackle the same problem. Their main contribution includes a statistical method for comparing algorithms that was used to build a heuristic that performed better than the earlier heuristics. The so-called population-less co-operative genetic algorithm by Jan et al. (2000) and (2002) applies a 2-point crossover on the worst and a randomly selected schedule. The genetic algorithm searches solutions in the feasible region only. Nevertheless, applying crossover to nurse rostering nearly always causes problems of infeasibility. Kawanaka et al. (2001) overcome this problem by exchanging shifts while attempting to maintain certain characteristics of the parents. Burke et al. (2001) present

a set of genetic as well as memetic algorithms (genetic algorithms extended with local improvement procedures) to address some of the shortcomings of the approach described in Burke et al. (1999). The authors show that these memetic algorithms can handle initialization parameters and a range of instances more robustly than a single-population approach (tabu search, see Burke et al., 1999 and 2003), at the expense of longer computation times.

1.4.4 Operating room scheduling literature review

The literature on operating room scheduling can be structured using the three stages that can be distinguished in developing operating room schedules. In the first stage, often called case mix planning, it is decided how the available operating room time is divided over the different surgeons (or surgical groups). Case mix planning problems have been studied by amongst others Hughes and Soliman (1985), Rifai and Pecenka (1989), Robbins and Tuntiwongbiboon (1989) and Blake and Carter (2002) and (2003).

Once the operating room time allocated to each surgical group has been chosen, the second stage involves the development of a master surgery schedule. In the hierarchical framework for hospital production and control by Vissers et al. (2001) this second stage of operating room scheduling could be positioned somewhere between the Resource Planning & Control level and the Patient Group Planning & Control level. The master surgery schedule is a cyclic timetable that defines the number and type of operating rooms available, the hours that rooms will be open, and the surgical groups or surgeons who are to be given priority for the operating room time (Blake et al., 2002). A new master schedule is created whenever the total amount of operating room time changes. Blake et al. (2002) propose an integer programming model that minimizes the weighted average undersupply of operating room hours, that is allocating to each surgical group a number of operating room hours as close as possible to its target operating room hours (see also Blake and Donald, 2002). Santibanez et al. (2005) present a system-wide optimization model for block scheduling that enables managers to explore trade-offs between operating room availability, booking privileges by surgeons, bed capacity and waitlists for patients.

After the development of the master surgery schedule, elective cases can be scheduled. This third stage involves the detailed planning of each elective case. This stage has more of an operational focus as it occurs on a daily basis and includes operational scheduling decisions like assigning the cases to operating rooms, determining the order and the start and end times of the cases (e.g., Weiss, 1990; Ozkarahan, 1995 and 2000), the reservation of specialized equipment etc. Sometimes, this third stage is integrated with longer term scheduling. For instance, Guinet and Chaabane (2003) and Jebali et al. (2006) propose a two-step approach for operating theatre planning. The first step involves assigning patient interventions to operating rooms on a medium term horizon. The second step entails the daily rescheduling of the patient interventions in order to integrate more characteristics for human and material resource synchronization. Other interesting work that applies on the third stage has been done by Lapierre et al. (1999), Dexter et al. (1999) and (2001), Dexter and Traub (2002) and Marcon et al. (2003).

When building surgery schedules, several objectives could be taken into account. Much research has focussed on the maximization of operating room utilization for which many algorithms have been studied ranging from simple heuristics (e.g., earliest start time first, largest duration first, etc.) to more complex bin packing algorithms (see, e.g., Dexter et al., 1999; Dexter and Traub, 2002). A strongly related objective is to minimize the operating room staffing costs (e.g., Dexter et al., 2000). An objective that receives more and more attention nowadays is the management of uncertainty. Many studies have focussed on increasing the punctuality of the schedule realized (e.g., Lapierre et al., 1999; Dexter et al., 2001; Marcon et al., 2003). Obviously, managing uncertainty requires insight into a number of aspects of the interaction of the planned (elective) and the emergency (non-elective) cases. Gerchak et al. (1996) provide a stochastic dynamic programming model for the advance scheduling of elective surgery under uncertain demand for emergency surgery. Kim et al. (2000) describe a flexible bed allocation scheme that reserves one or more beds for the exclusive use of elective-surgery patients to enhance the operations of the intensive care unit. Kim and Horowitz (2002) elaborate on this work and use a simulation model to show that the combination of this flexible bed allocation scheme and a quota system for elective surgery greatly reduces the number of canceled surgeries. Bowers and Mould (2004) propose a policy of including planned, elective patients within the trauma (non-elective) session and show by

means of simulation how substantially greater throughputs can be achieved.

The management of resources is often considered a crucial issue in operating room scheduling. Ozkarahan (1995) proposes an expert hospital decision support system for resource scheduling that combines mathematical programming, knowledge base, and database technologies. Five years later, the same author (Ozkarahan, 2000) describes a goal programming model which can produce schedules that best serve the needs of the hospital, i.e., by minimizing idle time and overtime, and increasing satisfaction of surgeons, patients, and staff. The approach involves sorting the requests for a particular day on the basis of block restrictions, room utilization, surgeon preferences and intensive care capabilities.

When planning a surgical procedure, one of the first things checked is whether or not a bed is available for the patient. A good assignment of the current bed capacity to incoming patients may result in the reservation of buffer capacity to absorb peeks in emergency cases. Once again, a variety of constraints have to be taken into account when assigning beds to patients. It is for instance not desirable that male and female patients or people with large age differences share a room. Clerkin et al. (1995) propose an automated expert system that assigns beds to incoming patients. The relation between bed occupancy and the surgery schedule has been subject to many studies (e.g., Dumas, 1984 and 1985; Harris, 1985; Wright, 1987; Gorunescu et al., 2002; McManus et al., 2004; Santibanez et al., 2005). Also, the importance of the relation to staff decision problems has been recognized. Dexter and Traub (2000) determine staffing requirements for a second shift of anaesthetists by graphical analysis of data from operating room information systems. Griffiths et al. (2005) model the requirement for supplementary nurses in an intensive care unit. Hamilton and Breslawski (1994) argue that the factors considered by operating room administrators to be critical to operating room scheduling are dependent on the nature of the scheduling system. The results of their large scale survey indicated that in block systems, which is the system used throughout this dissertation, the number of operating rooms, the equipment limitations, the block times assigned and the hospital scheduling policy are considered to be important criteria. In first come, first served systems the number of operating rooms, the estimated room set up duration, the estimated case duration and the equipment restrictions are considered to be essential.

1.5 Summary

Due to the ageing of society and the continuously growing demands, health care is becoming increasingly expensive. Techniques originating from operations research, decision support and artificial intelligence might provide an answer to cope with the growing pressure on scarce resources. A key issue for successful health care management is scheduling. The quality of the produced schedules as well as the efficiency with which the schedules are developed have a large contribution to the overall hospital performance. Well-thought-out scheduling procedures can lead to many benefits, financial ones as well as social ones. Good scheduling practices result in a more efficient use of recourses and hence reduce costs. Social benefits might include the increase of the personnel's satisfaction and well being, safer working environments and a higher quality of care for the patients.

This thesis deals with algorithmic procedures for solving scheduling problems encountered in hospitals. Our research concerns both staff and operating room scheduling problems. The research is fundamental as well as applied. Although heuristic approaches are included, the main focus lies on exact solution procedures.

Almost all problems will be stated using an integer programming formulation. While developing the different algorithms, several mathematical programming techniques are considered. The techniques that are most often used in this thesis are column generation, branch-and-price and deterministic dynamic programming.

The remainder of this thesis is organized as follows. Chapter 2 discusses in detail the solution approach for solving a trainee scheduling problem. Chapter 3 presents a computer program that visualizes the load of various resources as a function of the operating room schedule. Chapter 4 then focuses on one of these resources, namely beds, and proposes a number of operating room scheduling models, formulations and algorithms that aim at leveling the resultant bed occupancy. Subsequently, Chapter 5 describes a real-life application of the models and algorithms developed in the preceding chapter. Chapter 6 provides a conjunction of the staff scheduling techniques with the operating room scheduling procedures, as it proposes an integrated approach for building surgery and nurse schedules. To end, Chapter 7 draws some general conclusions and sketches some directions for future research.

Chapter 2

Scheduling trainees at a hospital department using a branch-and-price approach

Scheduling trainees is a complicated problem that has to be solved frequently in many hospital departments. We will describe a trainee scheduling problem encountered in practice at the ophthalmology department of the university hospital Gasthuisberg, Leuven. In this problem a department has a number of trainees at its disposal, which assist specialists in their activities (surgery, consultation, etc.). For each trainee one has to schedule the activities in which (s)he will assist during a certain time horizon, usually one year. Typically, this kind of scheduling problem is characterized by four types of constraints: work covering constraints, formation requirements, non-availability constraints and setup restrictions. This chapter describes a number of exact branch-and-price methods to solve the problem to optimality and a number of heuristic extensions to find good solutions for a generalized version of the problem.

2.1 Introduction

In this chapter the problem of scheduling medical trainees to perform a number of activities over a given time horizon is addressed. Although frequently encountered in practice, to the best of our knowledge, no papers in the literature deal with this problem. In a broader view the trainee scheduling problem can however be classified as a medical staff scheduling problem. It distinguishes from the classic Nurse Scheduling Problem (NSP) in the fact that the NSP usually deals with detailed shift scheduling, e.g., determining the exact hours nurses have to work during the next month, whereas trainees are scheduled over a much longer time horizon (usually one year). Moreover, in contrast to qualified nurses, trainees still have to complete an education. This education requires that they have to perform a number of widely divergent activities and the capacity of the trainee posts is often limited. Consequently, the set covering constraints in the NSP are replaced by set partitioning constraints. A second important difference is the undesirability of a situation where a trainee alternates too much between the activities. Each activity (re)start represents a discontinuity in his/her education and involves a considerable mastering time for the trainee. Hence, when a trainee has a week-off, (s)he cannot simply be replaced by another trainee, due to the difference in qualification and due to the mastering time. Nurses, however, can usually exchange weeks-off rather easily by mutual agreement.

To solve the trainee scheduling problem, we will develop two branch-and-price algorithms. Branch-and-price has gained considerable attention during the last decade. Most of the encountered scheduling problems studied in the literature are short-term shift scheduling problems involving some kind of set covering or set partitioning formulation (e.g., Caprara et al., 2003; Jaumard et al., 1998; Bard and Purnomo, 2005a and 2005b; Mehrotra et al., 2000; Mason and Smith, 1998). Alternatively, 0-1 multi-commodity flow formulations are proposed (e.g., Cappanera and Gallo, 2001; Moz and Pato, 2003). To the best of our knowledge all branch-and-price approaches for staff scheduling problems decompose on staff members, i.e., generate columns per staff member. In contrast, we study a slightly simplified version of a long-term scheduling problem for which we propose a decomposition scheme on the tasks, further referred to as activities, instead of decomposing on the employees. This approach enables us to find optimal solutions for real-life data sets. The problem will be written as a 0-1 multi-commodity flow problem with

side constraints, where each activity corresponds to a commodity. Afterwards, this original approach is compared to the more common approach in which the problem is decomposed on the staff members resulting in a set partitioning problem.

The remainder of this chapter is organized as follows. In Section 2.2 the problem will be stated and written as an integer program. Next, Section 2.3 describes an exact branch-and-price method in which a slightly simplified version of the problem is solved, using a decomposition on the activities approach. Section 2.4 discusses the computational results that are obtained. Section 2.5 draws some conclusions with respect to the decomposition on the activities approach and introduces the remaining part of this chapter. In Section 2.6 the developed approach is compared with the more common solution procedure for staff scheduling problems in which the problem is decomposed on the trainees. Subsequently, Section 2.7 describes a generalization of the problem and proposes some heuristic extensions. Section 2.8 presents the graphical user interface that was developed on top of the algorithmic procedures. Finally, Section 2.9 draws overall conclusions and lists some topics for future research.

2.2 Problem Statement

Consider a hospital department in which trainees have to perform a number of activities over a certain time horizon. The task is to schedule these trainees to perform the activities so that a number of constraints is satisfied. First, during each period, each activity has to be performed by exactly one trainee out of a given set. These trainee sets are mainly determined by the experience levels of the trainees. Second, for each trainee the minimum and maximum number of periods (s)he has to perform each activity is given in order to meet formation objectives. Third, for each trainee it is known for each time period whether or not (s)he is available to be scheduled. Finally, in order to maximize both the efficiency and the quality of the service provided, we cannot split activities up per trainee. The efficiency decreases with each new activity start of a trainee, because it takes (again) some time to master the skills required for the activity. Moreover, patients have a smaller chance to be treated by one and the same trainee, resulting in less efficient care. In the ideal case each trainee starts each activity only once and performs it for a minimum

number of consecutive periods.

The last two constraints are soft constraints meaning that they can be violated at a certain cost. Since a split-up in activities is considered to be worse than the violation of a non-availability constraint, we will concentrate on the problem solution in which we only relax the non-availability constraints. Therefore, the trainees have to quantify their preferences for having weeks off. This happens by dividing a number of points per trainee over the scheduling horizon. The higher the number of points a certain period receives, the stronger the trainee feels about not being scheduled during that period.

Let us illustrate this problem with a simple example. Suppose we have a problem with three activities, four trainees and ten periods. Furthermore, suppose that each assistant has to perform each activity between a minimum of two and a maximum of three consecutive periods. This example is graphically represented in Table 2.1. The columns represent the trainees and the rows represent the periods. The numbers indicate the non-availability costs for each trainee. Note that each trainee has divided in total five points over the ten periods. It is realistic that these points are concentrated in a small number of periods.

A possible solution for this problem is represented in Table 2.2. In this solution, trainees 1 and 3 are both scheduled during a period in which they actually prefer not to be scheduled, respectively period 7 and period 9 (indicated in bold), making up for a total cost of 1+1 = 2. In practice, this means that either the trainee has to give up his/her preference for having a period off or the trainee has to be replaced by someone else in this period, resulting in a decrease of the quality of care. As a final remark, observe that in this solution during four time periods no activity is scheduled for a particular trainee although (s)he is available. During these periods, the trainees will perform activities for which no specific skills are required and for which consequently both the experience level and the minimal formation requirements are less important. An example of such an activity is consultation. Such activities are called easy activities, in contrast with the difficult activities that were discussed above. A two-phase approach is thus being used to solve this problem. In the first phase, the difficult activities are scheduled. In the second phase, the partial schedule is completed with the easy activities. The scheduling of the easy activities is straightforward and can easily be done manually. Therefore, we will

only concentrate on the scheduling of the difficult activities in the first part of this chapter.

Table 2.1: A problem instance

	Non-availability costs			
Period	Trainee 1	Trainee 2	Trainee 3	Trainee 4
1				
2	4			
3			4	
4		1		
5				
6				2
7	1			
8		4		
9			1	
10				3

In order to provide more insight into the problem, we will shortly describe how this task was carried out up till now. In a first step, the responsible scheduler collects the required data. Coverage constraints and formation requirements are provided by the head of the department. Non-availability constraints are collected in a hierarchical way. A list circulates in which the trainees successively indicate during which weeks they will be absent and during which weeks they would like to take vacation. To ensure that vacation periods are sufficiently spread, the number of trainees having vacation at the same time is limited. Next, using pencil and paper, the scheduler tries to find a schedule that satisfies as many constraints as possible. She mainly concentrates on the satisfaction of the coverage constraints. At a certain moment, typically when about 75% of the schedule is completed, she fails to satisfy the next coverage constraints without violating one or more of the formation, non-availability or setup constraints. At that moment, she tries to solve the schedule conflict by making a number of assignments undone or performing a number of switches. If she fails to solve the conflict in a limited number of tries, she accepts the violation of one (or more) constraints and continues the construction

Table 2.2: A solution for the problem instance

	Activity schedule			
Period	Trainee 1	Trainee 2	Trainee 3	Trainee 4
1		act 1	act 2	act 3
2		act 1	act 2	act 3
3	act 3	act 1		act 2
4	act 3		act 1	act 2
5		act 3	act 1	act 2
6	act 2	act 3	act 1	
7	**act 2**	act 3		act 1
8	act 2		act 3	act 1
9	act 1	act 2	**act 3**	
10	act 1	act 2	act 3	

of the schedule. Upon completion, the schedule is communicated to all the people involved (trainees and surgeons). Since this task essentially involves the solution of a complex combinatorial puzzle and PC's are typically more suitable to solve such problems than humans, we believe that a well-thought-out algorithm could save construction time as well as generate qualitatively better schedules. First, we will state the problem mathematically as an integer program.

If we are to state an integer programming formulation for this problem, we first have to define a set of decision variables. Let i be the period index, j the trainee index and k the activity index. Since we have to decide for each trainee which activity (s)he will perform during each week, a natural choice of decision variables would be:

$$x_{ijk} = \begin{cases} 1, & \text{if during period } i \text{ trainee } j \text{ is scheduled to perform activity } k; \\ 0, & \text{otherwise.} \end{cases}$$

$$y_{ijk} = \begin{cases} 1, & \text{if trainee } j \text{ starts activity } k \text{ during period } i; \\ 0, & \text{otherwise.} \end{cases}$$

Let p_{ij} be the penalty cost charged for assigning trainee j to period i. It must be clear that p_{ij} equals 0 if trainee j is available during period i. Let l_{jk} and u_{jk} be the respective minimum and maximum number of periods assistant j has to perform activity k. We assume that $u_{jk} > 0$ implies $l_{jk} > 0$ although this assumption is not necessary for our algorithm. Finally, let S^k represent the set of trainees that will perform activity k in the given time horizon (i.e., all trainees j for which $u_{jk} > 0$). The integer linear programming model (ILP) is given below.

$$\text{Minimize} \sum_{i=1}^{n} \sum_{k=1}^{p} \sum_{j \in S^k} p_{ij} x_{ijk} \tag{2.1}$$

subject to:

$$\sum_{k=1}^{p} x_{ijk} \leq 1 \qquad \forall i = 1, ..., n \text{ and } \forall j = 1, ..., m \tag{2.2}$$

$$\sum_{j \in S^k} x_{ijk} = 1 \qquad \forall i = 1, ..., n \text{ and } \forall k = 1, ..., p \tag{2.3}$$

$$\sum_{i=1}^{n} x_{ijk} \geq l_{jk} \qquad \forall k = 1, ..., p \text{ and } \forall j \in S^k \tag{2.4}$$

$$\sum_{i=1}^{n} x_{ijk} \leq u_{jk} \qquad \forall k = 1, ..., p \text{ and } \forall j \in S^k \tag{2.5}$$

$$y_{1jk} = x_{1jk} \qquad \forall k = 1, ..., p \text{ and } \forall j \in S^k \tag{2.6}$$

$$y_{ijk} \geq x_{ijk} - x_{(i-1)jk} \qquad \forall i = 2, ..., n \text{ and } \forall k = 1, ..., p \text{ and } \forall j \in S^k \tag{2.7}$$

$$\sum_{i=1}^{n} y_{ijk} = 1 \qquad \forall k = 1, ..., p \text{ and } \forall j \in S^k \tag{2.8}$$

$$y_{ijk}, x_{ijk} \in \{0, 1\} \qquad \forall i = 1, ..., n \text{ and } \forall k = 1, ..., p \text{ and } \forall j \in S^k \tag{2.9}$$

The objective function (2.1) minimizes the total schedule cost. Constraint set (2.2) ensures that each trainee can perform no more than one activity in each period. Constraint set (2.3) makes sure that in each period every activity is performed by exactly one trainee. Constraint sets (2.4) and (2.5) state that each trainee performs each activity between a minimum and a maximum number of periods. The fact that each trainee starts each activity only once is reflected in (2.6), (2.7) and (2.8).

Finally, (2.9) defines x and y as binary variables.

A last remark concerns activities that require more than one trainee during each period. We replace each of these activities by two or more artificial activities, dividing the trainees in the original set S^k over these new activities. The total number of resulting activities equals the number of required trainees for that activity. As each resulting activity requires one trainee, the scheduling of this new set of activities satisfies the original coverage constraint. The trainees having to perform the original activity are divided over the new set of activities. This division takes place before the start of our algorithm. Consider, for instance, an activity where two trainees are required each time period and twelve trainees have to perform the activity. Then, this activity will be replaced by two new activities each of which has to be performed by six trainees. This assumption facilitates the construction of an enumeration scheme at the cost of possibly excluding an optimal solution, since we only consider one particular division of trainees over the newly introduced activities.

For a problem with n time periods, m trainees and p activities, this notation requires at most $2nmp$ binary decision variables. For instance, a problem with 35 periods, 8 trainees and 6 activities involves at most 3360 binary decision variables in the formulation of (1)-(9). In Section 2.4.1 we show that problems of these dimensions require long solution times using this formulation and a commercial integer linear programming solver. A dedicated branch-and-price procedure will be shown to be more suitable to solve this problem.

2.3 A branch-and-price approach

2.3.1 Decomposition of the problem

Decomposition involves the division of the problem into several subproblems. Another way of seeing this is to introduce new decision variables, each one representing a subset of the old decision variables, that implicitly satisfy a number of constraints. Solving a subproblem is then analogous to finding a new decision variable or column for the ILP. The constraints that remain in the ILP can be seen as the *linking* constraints. The advantage of such an approach is that the LP bound of the new formulation is usually much stronger than that of the original formulation and consequently the branch-and-bound tree remains smaller.

2.3.2 An alternative formulation

The integer program of (2.1)-(2.9) could be formulated in a totally different way. Observe that the problem can be seen as the scheduling of p activity patterns. An activity pattern includes the scheduling of all trainees having to perform the activity. For reasons that will become clear in a moment, an activity pattern will be called a column in the rest of this chapter. Now, we can introduce new binary decision variables that explicitly incorporate these columns. Let binary decision variable z_{kt} be defined as follows:

$$z_{kt} = \begin{cases} 1, & \text{if column } t \text{ was chosen for activity } k; \\ 0, & \text{otherwise.} \end{cases}$$

Let c_{kt} be the total cost of column t for activity k and NC_k the total number of different columns for activity k. Let a_{ijkt} equal 1 if in column t trainee j is scheduled during period i for activity k. The model can then be formulated as follows:

$$\text{Minimize} \sum_{k=1}^{p} \sum_{t=1}^{NC_k} c_{kt} z_{kt} \tag{2.10}$$

subject to:

$$\sum_{k=1}^{p} \sum_{t=1}^{NC_k} a_{ijkt} z_{kt} \leq 1 \qquad \forall i = 1, ..., n \text{ and } \forall j = 1, ..., m \tag{2.11}$$

$$\sum_{t=1}^{NC_k} z_{kt} = 1 \qquad \forall k = 1, ..., p \tag{2.12}$$

$$z_{kt} \in \{0, 1\} \qquad \forall k = 1, ..., p \text{ and } \forall t = 1, ..., NC_k \tag{2.13}$$

The objective function is again the minimization of costs, but now expressed in terms of the new z_{kt} variables. Constraint set (2.11) states that each trainee can

perform no more than one activity at the same time. Constraint set (2.12) implies that exactly one column has to be selected for each activity. The main drawback, however, is that this new model can have far more variables than can be reasonably attacked directly. Fortunately, column generation can help to overcome this difficulty. This technique is well known for this type of problems (see, e.g., Jaumard et al., 1998; Bard and Purnomo, 2005b). Column generation is based on the observation that it is not necessary to enumerate all possible columns in order to solve the LP to optimality. The LP can be solved by using only a subset of the columns and can generate more columns as needed.

The LP is solved to optimality when no more columns price out, i.e., no more columns with negative reduced cost can be found. Let λ_{ij} represent the dual prices of restrictions (2.11) and let γ_k represent the dual prices of restrictions (2.12). The reduced cost of a new column t for activity k is given by:

$$\begin{aligned} & c_{kt} - \gamma_k - \sum_{i=1}^{n}\sum_{j=1}^{m} \lambda_{ij} a_{ijkt} \\ & = -\gamma_k + \sum_{i=1}^{n}\sum_{j=1}^{m} (p_{ij} - \lambda_{ij}) a_{ijkt} \end{aligned} \tag{2.14}$$

The master problem (2.10)-(2.13) is in fact a 0-1 capacitated multicommodity flow problem in which each activity corresponds to a commodity and all arc capacities and commodity requirements are equal to 1. Tests revealed that the LP relaxation of this formulation provides a much stronger lower bound than that from the original formulation of (2.1)-(2.9) (see Section 2.4.1). The reason is that in the new formulation (2.10)-(2.12) the optimization is performed over the convex hull of feasible points of the subproblems, and not just over the relaxed feasible region of (2.1)-(2.8). It can easily be shown that each feasible solution of the new formulation (2.10)-(2.12) is also feasible in the original formulation (2.1)-(2.8). To see this, observe that each z_{kt} solution can be reformed to an x_{ijk} solution, which is feasible to (2.1)-(2.8). Therefore, simply set each x_{ijk} equal to $\sum_{t}^{NC_k} a_{ijkt} z_{kt}$. The reverse is not true when the convex hull of the subproblem constraints is not integral. The

feasible solution space of (2.1)-(2.8) is much larger than that of (2.10)-(2.12). Although a smaller feasible region is no guarantee to obtain a better bound, it is quite likely that the optimal solution of (2.1)-(2.8) is not feasible in the new formulation (2.10)-(2.12).

In the next sections the branch-and-price algorithm will be expanded upon. First, an overview of the algorithm is given. Second, the pricing problem is discussed. Third, three possible branching strategies are elaborated. Finally, some improvements to speed up the computation time will be explored.

2.3.3 Branch-and-price algorithm overview

In Algorithm 1, an overview of the branch-and-price algorithm is given. The algorithm starts with a heuristic in order to find an initial solution. This heuristic successively generates activity patterns (columns) without violating the no-overlap constraint. If the algorithm succeeds in finding a feasible solution, the schedule is saved and an initial upper bound is registered. The master is initialized with both the p columns making up the best solution found and p supercolumns (one per activity), which are needed to ensure feasibility of the master at each level of the branch-and-bound tree. Each iteration of the main while loop consists of two parts: the LP optimization loop (which upon termination provides a lower bound) and a move in the branch-and-bound search tree.

Algorithm 1 BRANCH-AND-PRICE

```
apply heuristic to find initial solution;
if (solution found) then
  register schedule;
  upper_bound ← best solution found;
  initiate master with p columns from initial solution and p supercolumns;
else
  upper_bound ← +∞;
  initiate master with p supercolumns;
end if
l ← 0;
while (l ≥ 0) do
  LP_opt_found ← FALSE;
  while (LP_opt_found=FALSE) do
    LP_opt_found ← TRUE;
    upper_bound ← SOLVE-MASTER-LP();
    for (k = 1 to p) do
      RC_k ← FIND-NEW-COLUMN(k);
      if (RC_k < 0) then
        add new column to master;
        LP_opt_found ← FALSE;
      end if
    end for
  end while
  continue ← TRUE;
  while (continue=TRUE) do
    if (LP_opt ≥ upper_bound) then
      while (all branches on level l explored) do
        l ← l − 1; {backtrack}
      end while
      explore next branch on level l;
      add corresponding branching restriction;
      continue ← FALSE;
    else if (fractional solution) then
      l ← l + 1; {branch one level further}
      add new branching restriction;
      continue ← FALSE;
    else if (integral solution) then
      register schedule;
      upper_bound ← LP_opt;
    end if
  end while
end while
```

2.3.4 The pricing problem

The pricing problem for activity k can be stated as follows. Let x_{ij} equal 1 if trainee j is scheduled to perform activity k during period i and let y_{ij} be 1 if trainee j starts activity k at period i.

$$\text{Minimize} \sum_{i=1}^{n} \sum_{j \in S^k} (p_{ij} - \lambda_{ij}) x_{ij} \tag{2.15}$$

subject to:

$$\sum_{j \in S^k} x_{ij} = 1 \qquad \forall i = 1, ..., n \tag{2.16}$$

$$\sum_{i=1}^{n} x_{ij} \geq l_{jk} \qquad \forall j \in S^k \tag{2.17}$$

$$\sum_{i=1}^{n} x_{ij} \leq u_{jk} \qquad \forall j \in S^k \tag{2.18}$$

$$y_{1j} = x_{1j} \qquad \forall j \in S^k \tag{2.19}$$

$$y_{ij} \geq x_{ij} - x_{(i-1)j} \qquad \forall i = 2, ..., n \text{ and } \forall j \in S^k \tag{2.20}$$

$$\sum_{i=1}^{n} y_{ij} \leq 1 \qquad \forall j \in S^k \tag{2.21}$$

$$x_{ij}, y_{ij} \in \{0, 1\} \qquad \forall i = 1, ..., n \text{ and } \forall j \in S^k \tag{2.22}$$

Objective (2.15) simply entails the minimization of the (variable part of) the reduced cost (2.14). Constraints (2.16)-(2.22) are just a repetition of the constraints (2.3)-(2.9) for activity k.

The pricing problem (2.15)-(2.22) is a restricted shortest path problem. Suppose we are searching a new column for activity k. This problem can be visualized by a matrix. The columns in this matrix represent the trainees $j \in S^k$ and the rows represent the time horizon. Each cell of the matrix has a cost g_{ij} which equals the corresponding non-availability cost p_{ij} minus the corresponding dual price λ_{ij}. This matrix has to be traversed from top to bottom in the cheapest way possible, while visiting each column exactly once between a minimum and a maximum number of rows. Table 2.3 represents the best solution and Table 2.4 represents one of the several alternative second best solutions for an instance of a pricing problem for an activity with four trainees that all have to be scheduled between one and two

periods in a time horizon of six periods. The found solutions are indicated in bold. Hence, the optimal solution first schedules trainee 1 for two periods, followed by trainee 3 for two periods, then trainee 2 for one period and finally trainee 4 for one period.

Table 2.3: Pricing problem[a]: optimal solution indicated in bold

	g_{ij} = non-availability cost p_{ij}- dual price λ_{ij}			
Period i	Trainee $j=1$	Trainee $j=2$	Trainee $j=3$	Trainee $j=4$
1	**0**	1	1	2
2	**0**	1	1	1
3	4	2	**1**	4
4	2	1	**0**	0
5	0	**0**	4	0
6	1	5	3	**1**

[a] For ease of explanation all cost values are integer. Note however that during column generation these cost values are usually fractional due to the dual prices.

The pricing problem is solved with a dynamic programming approach (Bellman, 1957; Dreyfus and Law, 1977). Let T denote a set of trainees. The subproblem can be described as finding the cheapest way to reach period i with all trainees in T scheduled. Let $cost(i,T)$ represent this cost. If g_{ij} is the cost to assign period i to trainee j and we search a column for activity k, then $cost(i,T)$ can be formulated recursively as follows:

$$cost(i,T) = \underset{j\in T}{\text{MIN}}\left\{ \underset{l_{jk}\le d\le u_{jk}}{\text{MIN}}\left\{ cost(i-d, T\backslash\{j\}) + \sum_{b=1}^{d} g_{i-d+b,j} \right\} \right\} \tag{2.23}$$

Table 2.4: Pricing problem[a]: 2^{nd} best solution indicated in bold

	g_{ij} = non-availability cost p_{ij}- dual price λ_{ij}			
Period i	Trainee $j=1$	Trainee $j=2$	Trainee $j=3$	Trainee $j=4$
1	**0**	1	1	2
2	**0**	1	1	1
3	4	2	**1**	4
4	2	**1**	0	0
5	0	**0**	4	0
6	1	5	3	**1**

[a] For ease of explanation all cost values are integer. Note however that during column generation these cost values are usually fractional due to the dual prices.

To solve the pricing problem, we must calculate $cost(n, S^k)$ using (2.23) and make sure we know which schedule it represents. The different values for $cost(i|T)$ can be calculated working backwards using a recursive procedure. After application of the backward dynamic recursion to the example above, the values for $cost(i, T)$ are known for all possible values of i and T. These values are represented in the left part of Table 2.5.

Obviously, values for state spaces that cannot lead to a feasible path are not calculated. Nevertheless, the number of cost calculations grows exponentially with the number of trainees that have to be scheduled. For realistic data (number of trainees) this number is not too large which results in acceptable solution times ($\ll$ 1s).

Once all calculations are done, the cheapest way to reach period n can be found easily. Observe that $cost(n|S^k) = c^*$ contains the cheapest cost. The schedule is constructed backward step-by-step by searching for which trainee $j \in S^k$ and for which $d = l_{jk}$ to u_{jk} the following expression holds:

$$c^* = cost(n - d|S^k \backslash \{j\}) + \sum_{b=1}^{d} g_{n-d+b,j} \tag{2.24}$$

If a match is encountered, trainee j is scheduled for d periods starting from the current period to the front and both the current period and the set of already scheduled trainees are updated. This process continues until the last trainee is scheduled at the beginning of the scheduling horizon.

An important advantage of the pure dynamic programming approach (i.e., without lower bound pruning) is that it can be easily extended to find the b^{th} best solution instead of only the optimal solution. This property is very useful in a branch-and-price environment with branching on the column variables z_{kt} (see 2.3.6.1). Our algorithm to find the b^{th} best column reflects the same idea as the algorithm

Table 2.5: State spaces for example 1

Period i	T	$cost(i,T)$	2^{nd} best value
1	{1}	0	
	{2}	1	
	{3}	1	
	{4}	2	
2	{1}	**0**	$+\infty$
	{2}	2	
	{3}	2	
	{4}	3	
	{1,2}	1	
	{1,3}	1	
	{1,4}	1	
	{2,3}	2	
	{2,4}	2	
	{3,4}	2	
3	{1,2}	2	
	{1,3}	1	
	{1,4}	4	
	{2,3}	3	
	{2,4}	5	
	{3,4}	4	
4	{1,2}	3	
	{1,3}	**1**	8
	{1,4}	4	
	{2,3}	3	
	{2,4}	6	
	{3,4}	4	
	{1,2,3}	2	
	{1,2,4}	2	
	{1,3,4}	1	
	{2,3,4}	3	
5	{1,2,3}	**1**	2
	{1,2,4}	2	
	{1,3,4}	1	
	{2,3,4}	3	
6	{1,2,3,4}	**2**	3

proposed by Jiménez and Marzal (1999) for computing the K shortest paths in a network. To find the second best solution one first searches for the optimal solution as described above. Then, starting at the beginning of the time horizon all cost values being part of the found column are adapted to represent the second cheapest cost values. During this cost recalculation phase $|S^k|$ cost recalculations are made. We start with the first state of the optimal path (let j_1 be the first trainee scheduled and i_1 the number of periods this trainee is scheduled):

$$cost(i_1, \{j_1\}) = \infty \tag{2.25}$$

Working forward all cost values of the states (i, T) on the optimal path can now be recalculated using (2.23). After these recalculations the cost values represent the second best value.

The second best values for these states are indicated in the right part of Table 2.5. The recalculations are as follows. First, $cost(2|\{1\})$ with initial value zero is updated to ∞, since there is no other possibility to reach period 2 with only trainee 1 scheduled. Next, $cost(4|\{1,3\})$ with initial value one is updated to 8 (schedule trainee 3 during the first two periods and trainee 1 during the following two periods). The value of $cost(5|\{1,2,3\})$ changes from 1 to 2, since the cheapest way to reach period 5 with trainees 1, 2 and 3 scheduled is now by scheduling trainee 1 during periods 1 and 2, trainee 3 during period 3 and trainee 2 during periods 4 and 5. Finally, $cost(6|\{1,2,3,4\})$ is updated from 2 to 3 (recall that the second best path is represented in Table 2.4). Note that the same cost can also be obtained when scheduling trainee 2 only during period 4 and trainee 4 during periods 5 and 6.

When the $|S^k|$ cost values are updated, the backward construction phase described above will now generate the second best column.

The old cost values (and the partial paths represented by these values), however, have to be saved in a list, because they could be part of the second best column and thus could be necessary during backward construction. This will be the case if the second best column has the same *head* as the optimal column but a different *tail*, which is the case for the example in Table 2.4. When during backward construction no trainee assignment can be found that matches (2.24), the list has to be scanned. In this case the list always contains a matching cost value. Note that once a cost value is retrieved from the list, the remaining trainee assignments can

also be found in the list; they occupy the immediately preceding positions. After the recalculation phase the list contains the values of the old states, together with the paths they represent (Table 2.6).

For the example, the list is needed when state $(3, \{1, 3\})$ is reached. From this state, no matching cost value (state) can be found, since the needed value $cost(2, \{1\}) = 0$ now equals ∞. A scan through the list resolves the problem and completes the construction of the second best path. In the recalculation phase one also has to take into account the state spaces in the list as possible starting states for calculating the next best state values. These states can only be taken into consideration if the list does not already contain the same partial path as the one that would be constructed now, i.e., if the path represented by the start state (in the list) completed with the last arc is not already present in the list. This condition is needed to prevent construction of the same path twice. It is now clear why also the path has to be saved in the list. Since the list is empty during the first recalculation phase and this condition will never be satisfied during the second recalculation phase, the states in the list have only to be taken into consideration from the 3^{rd} recalculation phase (for finding the 4^{th} best column) on. Obviously, after each recalculation phase the list grows with $|S^k|$ (=nr. trainees) items.

Space complexity. For each time instance we need to store at most all possible subsets of m trainees. Hence, the space complexity is given by $O(n \cdot 2^m)$. To find the b^{th} best path we also need to store a list containing $(b-1) * m$ items, so the overall space complexity is $O(n \cdot 2^m + (b-1) \cdot m)$.

Run time complexity. Without loss of generality we assume the difference between the minimum and maximum number of periods equal for each trainee having to

Table 2.6: List after first recalculation phase for example 1

Item	i	T	$cost(i, T)$	path
1	2	{1}	0	1-1
2	4	{1,3}	1	1-1-3-3
3	5	{1,2,3}	1	1-1-3-3-2
4	6	{1,2,3,4}	2	1-1-3-3-2-4

perform activity k, i.e., $u_{jk} - l_{jk} = R$, $\forall j \in S^k$. The run time complexity for finding the b^{th} best path is analyzed for the three phases separately: backward calculation pass, recalculation pass and backward construction pass. In the backward calculation pass each state (i, S) leads to at most $R * m$ other states (as a matter of fact, this maximum only holds for state (n, S^k)). The complexity of the backward calculation pass is thus $O(n \cdot 2^m \cdot R \cdot m)$. In the recalculation pass at most m states have to be recalculated. For each of these states at most $R * m$ calculations have to be made, which gives complexity $O(m^2 \cdot R)$. For finding the b^{th} best path, one also needs to consider the $(b-1) * m$ states in the list as starting states for each updated state. For each of these starting states one has to verify if the new path (start state + new trainee) is not already present in the list in order to avoid generation of the same path twice. Hence, the complexity of the recalculation pass is $O(m^2 \cdot R + m \cdot ((b-1)m(b-1)m)) = O(m^2 \cdot R + m^3 \cdot (b-1)^2)$. In the backward construction pass each next trainee is found after visiting at most $R * m$ states. In the worst case, the last trainee cannot be found in the usual data structure, but is located in the list, which requires one scan through the list. The complexity of the backward pass for finding the b^{th} best path is thus $O(m^2 \cdot R + (b-1) \cdot m)$. Hence, the total complexity is given by $O(n \cdot 2^m \cdot R \cdot m + m^2 \cdot R + (b-1)^2 \cdot m^3)$. Since b is often very small, $(n \cdot 2^m \cdot R \cdot m)$ is the dominant term in this expression. This has been confirmed by our computational tests. The backward dynamic programming pass, which only has to be done to search the best column, is computationally much more expensive than the recalculation pass and the backward construction pass, which have to be done upon detection of an already found column.

2.3.5 Column addition

An important characteristic of the branch-and-price algorithm is the number of columns that are added after each master LP optimization. Three strategies can be distinguished. First, we could add the most negative reduced cost column for each activity. Second, we could add only one column for all activities, i.e., the column with the overall most negative reduced cost. Finally, we could add the most negative reduced cost column for activity k, re-optimize the master and search for a new column for activity $k+1$. Note that in this last strategy it is no longer possible to prune nodes based on Lagrange relaxation (see 2.3.7.2), since the reduced costs of all activities, needed to calculate the lower bound, are no longer available. Obviously, for all three strategies, columns with non-negative reduced cost are never added.

2.3.6 Branching

The LP relaxation of the master problem may not have an integral optimal solution. Branching refers to the process of partitioning the solution space to eliminate the current fractional solution. After branching, it may be the case that there exists a column that would price out favorably, but is not present in the column pool. Applying standard branch-and-bound procedures to the master problem over the existing columns is unlikely to find an optimal, or good, or even feasible solution. To illustrate this point, a branch-and-bound algorithm was written to find the best possible integral solution given the column pool after LP optimization. When the algorithm was run on the problem set, it never succeeded in finding a feasible solution, because the columns could not be combined into an integer solution. Three binary branching schemes were implemented and extensively tested: branching on the column variables, branching on timetable cells and branching on precedence relations.

2.3.6.1 Branching on column variables

In this branching scheme branching happens by fixing the largest fractional variable z_{kt} either to one (left branch) or to zero (right branch). It is however well known that *direct* partitioning of the solution space, i.e., by fixing (or bounding) individual column variables, is not appropriate because of two reasons. First, it could require significant alterations to the pricing problem and second, it yields an unbalanced branch-and-bound tree (Vanderbeck, 2000). The first problem is encountered along a branch, where a variable has been set to zero. Recall that z_{kt} represents a particular schedule for activity k. Hence, z_{kt} equal to 0 means that this schedule is excluded. However, it is possible (and quite likely) that the next time the pricing problem is solved for the k^{th} activity the optimal solution is precisely the one represented by z_{kt}. In that case it would be necessary to find the second best column. At depth l in the branch-and-bound tree we may need to find the l^{th} best column. We already showed that the dynamic programming approach for the pricing problem (see Section 2.3.4) can be easily extended to handle this need and this at a negligible computational effort. The unbalanced branch-and-bound tree remains a problem, but also involves an advantage as faster detection of (sub)optimal integral solutions may be expected.

2.3.6.2 Branching on timetable cells

Since timetable cells are represented by the original variables, this branching scheme will also be referred to as branching on the original variables. When columns can be associated with paths in a network, a possible branching scheme consists of fixing single components of the arc incidence vector (Vanderbeck, 2000). If this branching principle is applied to our problem, it results in branching on the original x_{ijk} variables. The next x_{ijk} to branch on is found by selecting the largest fractional column. Suppose this is a column for activity k. Then, we search for this column the first timetable cell (i, j) for which there exists another fractional column which schedules a different activity at timetable cell (i, j). In the left branch x_{ijk} is set to 1, in the right branch it is set to 0. It can easily be shown that this branching scheme is complete. In other words, upon detection of a fractional solution, it is always possible to find a pair of fractional columns to initiate a new branch. Alternatively, this branching rule can be seen as Ryan-Foster branching (Ryan and Foster, 1981) which was been developed for set partitioning problems. This branching scheme first identifies two rows, say r and s, covered by different fractional columns. In the left branch rows r and s have to be covered by the same column and in the right branch by different columns. Observe that in our application row r corresponds to one of the capacity constraints (2.11) and row s corresponds to one of the convexity constraints (2.12). The main advantage of this branching scheme is that it does not destroy the structure of the pricing problem, because the resulting modifications simply entail amending the cost of the corresponding arc in the underlying network. If x_{ijk} is set to 1, $g_{ij'}$ is set to $+\infty$ for all $j' \neq j$ in the pricing problem of activity k. For the pricing problems for activities $k' \neq k$ only g_{ij} is set to $+\infty$. Else if x_{ijk} is set to 0, g_{ij} is set to $+\infty$ in the pricing problem of activity k. A second advantage is the fact that this branching scheme yields a balanced branch-and-bound tree. The main drawbacks of this branching scheme are the large number of arcs (x_{ijk}'s) to choose from and the fact that a branching constraint that involves a single arc might not be very restrictive.

2.3.6.3 Branching on precedence relations

In this branching scheme, branching happens on precedence relations between the trainees performing an activity. A precedence relation for an activity k between two trainees, say j and j', simply states that trainee j has to perform activity k either before or after trainee j'. Upon detection of a fractional solution, the algorithm

searches for two fractional columns for the same activity with different orderings in trainee assignments over the scheduling horizon. Then, a precedence relation, which is satisfied by only one of both fractional columns, is implied. If, e.g., trainee j is succeeded by trainee j' in one fractional column and preceded by trainee j' in the other fractional column, the implied branching constraint could be: "trainee j before trainee j' for activity k". The main drawback of this branching scheme is that it is not guaranteed that it drives the solution completely to integrality. Theoretically, it is possible that an optimal fractional solution is found in which all fractional columns for each activity have the same trainee ordering. If this would be the case, the algorithm rounds all fractional values to 1 and verifies whether or not the resulting solution contains an overlap. In the case of an overlap, we have no feasible solution. The LP objective value (before rounding) provides, however, a lower bound (for that node) which is in general much better than the LP value of the root node, since it incorporates the constraint that fractional columns cannot have different orderings in trainees performing the activity. This is a relaxed constraint compared to the real constraint which states that only one column must be selected for each activity. However, it excludes the majority of fractional solutions and hence results in a better lower bound. Preliminary tests indicated that non-detection of an integer solution occurs rarely. Similar to the case in which branching occurs on the timetable cells, application of this branching scheme preserves the structure of the pricing problem. A precedence relation can be implied easily by simply amending the costs of certain arcs in the dynamic programming network. Compared to the other two branching schemes, this scheme clearly yields the most balanced branch-and-bound trees. The restrictions implied in the left branches are similar to the ones implied in the right branches, i.e., if trainee j cannot perform activity k before trainee j', it means that trainee j' has to perform activity k before trainee j.

2.3.7 Speed-up techniques

Since we have a method to generate columns and a branching scheme to cut away fractional solutions, our branch-and-price algorithm is complete. The performance of the algorithm is, however, strongly dependent on a number of speed-up techniques, which are described below.

2.3.7.1 Initial heuristic

The column pool is initialized with the columns making up an initial solution. The heuristic works as follows. First, the activities are sorted so that the most constrained activities are considered first. The less trainees have to perform an activity, the more constrained the activity is. In the case of a tie, the activity with the lowest average gap between maximum and minimum number of periods for which the trainees have to perform the activity, is the most constrained activity. Then, the first activity is scheduled optimally using the dynamic program of the pricing algorithm. All occupied timetable cells receive large costs (+1000) in order to exclude overlaps when scheduling the next activities. In addition, the timetable cell costs just before and after the already scheduled activity are slightly decreased (-0.05), making them more attractive for scheduling the following activities. This prevents as much as possible the appearance of *holes*, i.e., blocks of timetable cells that are too small to fit an activity. Then, the next activity is considered, taking into account the new timetable costs. This process continues until either all activities are scheduled or an activity cannot be scheduled any more due to an overlap. If all activities are scheduled, the total cost is compared with a solution that was found earlier and if lower, the upper bound is decreased. The changes to the timetable cell costs are made undone and the process restarts with the first activity. In order to avoid generation of the same columns as much as possible, the costs of the occupied timetable cells in the previous iteration are increased slightly (+0.01) before starting a new iteration. The heuristic ends if it fails to improve the current best solution during a predetermined number of iterations. For the tested problems, this number was set to 100, resulting in relatively small computation times, ranging from 0.05 to 2 seconds.

2.3.7.2 Lower bound calculation

Our column generation scheme exhibits the tailing-off effect, i.e., requiring a large number of iterations to prove LP optimality. Instead of solving the linear program to optimality, i.e., generating columns as long as profitable columns exist, we could end the column generation phase based on bound comparisons. It is well known that Lagrangian relaxation can complement column generation in that it can be used in every iteration of the column generation scheme to compute a lower bound to the original problem with little additional computational effort (see, e.g., Van den Akker et al., 2002; Vanderbeck and Wolsey, 1996). If this lower bound exceeds an

already found upper bound, the column generation phase can end without any risk of missing the optimum. Using the information from solving the reduced master and the information provided by solving a pricing problem for each activity k, it can be shown (see, e.g., Hans, 2001) that a lower bound is given by:

$$\delta + \sum_{k=1}^{p} RC_k \theta_k \tag{2.26}$$

where δ is the objective value of the reduced master, RC_k is the reduced cost of a newly found column for activity k and θ_k is a binary variable equal to 1 when RC_k is non-negative and set to zero, otherwise. This lower bound is referred to as the Lagrangian lower bound, since it can be shown that it equals the bound obtained by Lagrange relaxation. In addition with an upper bound it can also be used to fix variables. When the reduced cost of a variable z_{kt} is larger than $UB - LB$, we know from linear programming theory that $z_{kt} = 0$ in any solution with a value less than UB. Hence, that variable can be fixed in the current node and in all nodes below that node. Analogously, when the reduced cost is smaller than $LB - UB$ then $z_{kt} = 1$ in any solution with a value less than UB.

2.3.7.3 Initial network restriction

Recall that, to price out a new column, a shortest path network problem is solved by applying a forward dynamic program approach. For problems in which these networks are very large, the pricing problems are the bottleneck of the algorithm. We distinguish two ways of decreasing the required solution times of the pricing problems. First, one could initially restrict these networks. Specifically, arcs with positive non-availability costs are excluded during the early phase of each LP optimization loop. When no more columns can be found with negative reduced cost, these arcs are restored. The benefits are twofold. First, the required time for the pricing algorithm dramatically decreases during the early phase of column generation. Second, from the start on, the algorithm is forced to price out qualitatively good columns.

2.3.7.4 Master LP optimization

An important computational issue relates to the optimization of the master linear program. When new columns are added and the master is re-optimized, the (dual) simplex algorithm could be started either from an empty base or from the optimal

base of the previous iteration. Tests revealed that the LP is optimized fastest when started from an advanced base.

2.3.7.5 Cost varying horizon

To limit the solution space as much as possible, we implemented the idea of a cost varying horizon. This idea is equivalent with a time varying horizon in exact algorithms for the *Resource Constrained Project Scheduling Problem* (Demeulemeester and Herroelen, 2002). When implementing a cost varying horizon, one could distinguish between a maximum and a minimum bounding search strategy. Both strategies are different with respect to the value of the upper bound. In a minimum bounding search strategy the upper bound reflects the best found solution. When it is important to prove the optimality of a solution, a maximum bounding approach can be more effective than a minimum one. In a maximum bounding search strategy the upper bound is set to the first integer equal to or higher than the LP lower bound. If the algorithm succeeds in finding a solution with a total cost equal to this upper bound, we have found an optimal solution. Otherwise, both the upper and the lower bound are increased by one, the column pool is re-initiated with the columns making up the LP optimum and the algorithm tries to find a solution equal to this new upper bound. This approach corresponds to best-first search in branch-and-bound, in the sense that the first solution obtained is also the optimal solution. Tests indicated that the maximum bounding search slightly decreases computation times at the expense of not providing (sub)optimal solutions during the search process.

2.3.7.6 Column elimination

The idea of column elimination is inherent in all branching schemes except for the column-based scheme. To fully exploit the column-based branching strategy, the branching scheme was extended so that it also inherits the idea of column elimination. The solution time of the master LP grows strongly with the number of columns in the master, even when their associate column variables z_{kt} cannot be positive in a feasible solution. After branching, an important number of already generated columns could be excluded from the master. If a particular column, say $z_{k't'}$, is set to 1, all the other columns $z_{k't}$ with $t \neq t'$ are excluded implicitly because of the convexity constraint (2.12) in the master. To speed up the computation time of the master, these columns can be excluded explicitly from the master (by eliminating

them). Similarly, all columns having an overlap with column $z_{k't'}$ can be excluded as well, due to the no-overlap constraints (2.11). Observe that eliminated columns have to be saved, since they have to be reentered upon backtracking. Obviously, if column $z_{k't'}$ is set to 0, no columns but $z_{k't'}$ can be left out.

Column elimination is inherent when branching occurs on the original variables. Consider the situation in which $x_{i'j'k'}$ is set to 1. All columns $z_{k't}$ not including timetable cell (i', j') (i.e., having $a_{i'j'k't} = 0$) will be left out. Similarly, all columns z_{kt} with $k \neq k'$ including timetable cell (i', j') (i.e., having $a_{i'j'kt} = 1$) will be removed as well. If $x_{i'j'k'}$ is set to 0, the reverse applies. Column elimination is also inherent in the precedence relation branching scheme. Columns that do not satisfy the introduced precedence relations will be eliminated explicitly from the master. The same reasoning leads to the artificial adaptation of dual prices when branching occurs on the column variables. During preliminary tests of the algorithm, columns were generated that share timetable cells with already branched-to-one columns. Obviously, these columns can never enter the basis. The algorithm was adapted in that the dual prices of all timetable cells making up branched-to-one columns are increased with an artificially high value. Observe, again, that these artificial cost adaptations are inherent when branching is done on timetable cells and on precedence relations.

2.4 Computational results

2.4.1 Real-life data sets

First, the branch-and-price algorithm as well as the original ILP formulation of (2.1)-(2.9) have been applied on two real-life problems. For the branch-and-price algorithm, all above discussed speed-up techniques turned out to be useful to reduce computation times. The experiments were performed on a 2.4 GHz Pentium 4 PC with the Windows XP operating system. The algorithm was written in MS Visual C++.NET and linked with the CPLEX 8.1 optimization library. Also for the original ILP formulation of (2.1)-(2.9) we used the CPLEX 8.1 MIP solver with standard settings. All the speed-up techniques described above were incorporated in the branch-and-price algorithm. We apply maximum bounding search and distinguish between the three branching strategies. The real-life problems were rather small. Both problems involve 35 periods and 8 trainees. All trainees have

to perform 6 activities in the first and 7 activities in the second problem. The gaps between the maximum and minimum number of periods each trainee has to perform each activity vary between 1 and 4 for both problems. Table 2.7 contains the computational results for both problems.

Table 2.7: Results on two real-life problems

	Problem 1		Problem 2	
	ILP (2.1)-(2.9)	Branch-and-price	ILP (2.1)-(2.9)	Branch-and-price
Best solution found	-	16	-	10
LP relaxation	5.00	15.09	6.00	9.37
Explored nodes	453	7-100-21	390	34-24-27
Nodes left	416	0-0-0	328	0-0-0
Comp. time (s)	1800	9.59-78.73-16.53	1800	24.55-20.89-22.32

For the number of explored nodes, the number of nodes left and the computation time we distinguished in the branch-and-price algorithm between the three branching strategies, which explains the three numbers. For ILP (2.1)-(2.9) the optimizer was stopped after 1800 seconds of computation time. At that moment, in both problems the number of nodes left was still growing. As one can see from these results, our branch-and-price algorithm outperforms the original ILP formulation (2.1)-(2.9) substantially. Whereas through ILP (2.1)-(2.9) even though no feasible solution could be found within half an hour of computation time, the branch-and-price algorithm could solve both problems to optimality in times ranging from 10 seconds to 80 seconds, depending on the branching scheme. The main cause of the weak performance of ILP (2.1)-(2.9) is also indicated in the table. As one can see, there is a huge difference between the LP relaxations of both formulations: 5 versus 15.09 for the first problem and 6 versus 9.37 for the second problem. If we compare these bounds with the optimal solutions of 16 and 10 respectively, we conclude that the LP relaxation of ILP (2.1)-(2.9) is dramatically weak, whereas the LP relaxation of the branch-and-price formulation (2.10)-(2.13) is extremely strong.

2.4.2 Test set

At this point we were curious about how the algorithm would perform on larger problems, how the different problem dimensions influence the performance of the algorithm and how the different speed-up techniques contribute. Therefore, a test set was generated. First, six factors that have an influence on the complexity of the problem were identified. These are the number of periods, the number of trainees, the number of activities, for each activity the number of trainees performing the activity, the difference between the maximum and minimum number of consecutive weeks (further referred to as the range) and finally the magnitude of the costs. Table 2.8 contains a number of settings for these six factors.

Observe that the number of activities and the number of trainees having to perform an activity is expressed as a percentage of the number of trainees. For instance, a test problem with 10 trainees and 90% activities includes 9 activities. Note also that the number of activities cannot exceed the number of trainees, because otherwise not all activities can be performed. The ratio number activities over number trainees represents the total schedule occupation percentage. Recall that the remaining part of the schedule has to be filled up with activities for which the consecutiveness is not important. random(x) indicates that the factor setting is random with an average of x. For instance, the range setting random(2) means that the ranges are generated randomly in such a way that the average amounts to

Table 2.8: Design of the experiment

Factor setting	Nr. of periods	Nr. of trainees	Nr. of activities	Nr. of trainees per activity	Range	Magnitude of costs
1	18	6	60%	60%	1	1
2	35	8	75%	75%	2	U(1,5)
3	52	10	90%	90%	3	
4		12		random(75%)	4	
5					random(2)	
6					random(3)	

2. If the magnitude of the costs is 1, it means that all non-available time periods, which are generated randomly for each trainee, have a cost of 1. Alternatively, these cost values are drawn from a uniform distribution between 1 and 5. According to these factor settings, problem instances were generated with randomness on both the activity-trainee assignments and the non-available periods. In order to exclude non-feasible and trivial problems as much as possible, the trainee occupations were kept more or less at the same level. Without loss of generality, all non-availability costs are assumed to be integral. The total number of periods containing positive costs equals 3, 4 or 5 for problems with respectively 18, 35 and 52 periods. If we generate three problem instances per factor setting, we obtain 3*(3*4*3*4*6*2) = 5184 problem instances. In order to decrease this number, we decided to subsequently fix the first three factors and the next two factors (the fourth and fifth factor) at an intermediate level, making us end up with 3*(4*6*2)+ 3*(3*4*3*2) = 360 problem instances.

2.4.3 Discussion of results

In this section, we summarize the most important findings from our computational experiments. In Table 2.9, a subset of these results is represented in order to give the reader an idea of the running times, the number of generated columns, the number of nodes in the search tree, the lower bound of the root node, the heuristic solution value, etc. The problem name in the second column refers to the different factor settings for generating the problem (see Table 2.8). The first number stands for the first factor, the second for the second factor etc. The last number (after '_') indicates the replication number. Observe that for the last problems in our test set the algorithm fails to find an optimal solution. These are problems with 52 periods, 10 activities and 12 trainees. For these problems, the time limit of 600 seconds does not suffice to obtain an optimal integral solution. The root node could, however, be solved and hence a lower bound is obtained. The reason why the computation time was limited to 600 seconds is that the algorithms could not solve the problem to optimality within a reasonable time limit for the larger problem instances in our test set. A large number of the problems could however be solved within the limit of 600 seconds. This provides sufficient data to make a detailed analysis on how the problem dimensions influence the difficulty of the problem, on how the different branching schemes perform and on the contributions of the different speed-up

techniques.

In Table 2.10, the number of problems that could be solved to optimality within 10 minutes is given for each branching scheme together with the average computation times. The second row (*) contains the average times for only those problems for which all three branching schemes succeeded in finding (and proving) the optimal solution within 600 seconds. In the third row(**), average times are calculated based on all problems. For these calculations, 600 seconds were accounted for those problems for which no optimal solution was found within 600 seconds.

A first important observation is that the branching scheme that is based on precedence relations is clearly outperformed by the first two branching schemes. A second observation is that, although branching on timetable cells yields more problems solved to optimality, the required computation times are generally higher than those for the column-based branching scheme. This is a first indication of the appearance of unbalanced branch-and-bound trees when branching occurs on the column variables. In order to verify whether or not these results are statistically significant, the running times of the different branching strategies have been compared using a number of paired Student T-tests (two-tailed). In Table 2.11 the results are given. The large p-values for the comparison between the first and second branching strategy indicate that there is statistically no difference between branching on the column variables and branching on the timetable cells. All other differences are significant at the 5% level.
Next, the impact of the different factors in Table 2.8 on the running times has been statistically tested. The p-values for the different factors are indicated in Table 2.12. Again, a distinction is made between the three branching strategies. The small p-values for the first four factors indicate that each of them has a statistically significant influence on the running times. For the last two factors, range and magnitude of costs, the impact could not be confirmed by our test set.

Table 2.9: Subset of the results obtained

					branch on columns			branch on timetable cells			branch on prec. relations		
Nr.	Problem	root	heur	sol	time	cols	nodes	time	cols	nodes	time	cols	nodes
1	222111_1	13	14	13	0.22	69	1	0.2	63	1	0.2	63	1
2	222111_2	16	20	16	0.36	95	3	0.28	80	2	0.27	79	1
3	222111_3	16	17	16	0.2	60	1	0.25	69	4	0.31	78	8
4	222112_1	36	41	36	0.2	60	0	0.25	73	2	0.38	107	3
5	222112_2	50	53	50	0.22	69	0	0.28	84	2	0.44	116	5
6	222112_3	45	45	45	0.13	43	0	0.11	39	0	0.11	39	0
...													
64	222252_1	10.81	27	13	600	13252	1163	453.45	10680	684	561.53	10154	592
65	222252_2	22.58	38	24	63.9	3410	146	9.57	955	14	39.15	2275	58
66	222252_3	19.49	29	22	512.87	9910	589	174.91	7359	211	171.37	4916	192
67	222261_1	6	11	6	18.61	1635	27	19.77	1728	20	11.47	731	16
68	222261_2	7	10	8	121.47	3804	127	36.12	2485	44	35.17	1759	42
69	222261_3	4.01	10	5	35.36	2033	108	30.58	2425	44	10.52	693	23
70	222262_1	14.14	38	15	9.73	860	7	10.97	997	12	15.11	1006	15
71	222262_2	20.29	27	21	14.25	1254	15	25.14	2123	29	133.01	3504	74
72	222262_3	13.98	28	16	344.36	7687	1198	113.02	5798	243	193.96	6041	312
...													
112	222412_1	26.43	-	29	53.36	2955	328	24.56	2436	94	37.77	2641	110
113	222412_2	24.5	-	28	11.86	1447	70	10.02	1266	65	83.03	3911	561
114	222412_3	32.25	-	-	600	12525	3968	600	17638	3646	600	13329	2583
115	222421_1	6.57	12	7	9.97	1216	27	21.72	1928	33	7.27	649	12
116	222421_2	7.41	13	8	4.5	460	3	6.69	687	13	13.23	842	18
117	222421_3	8	10	8	13.59	1119	37	5.89	622	8	12.84	626	10
...													
241	222231_1	6	11	6	17.31	1456	18	12.19	1089	10	27.47	1445	23
242	222231_2	5.82	12	7	147.08	3888	200	69.8	3325	86	121.39	3829	155
243	222231_3	6.42	10	7	36.25	2252	58	8.25	840	11	449.3	6863	233
244	222232_1	12.39	22	13	57.16	2824	74	25.98	1742	21	71.73	2569	49
245	222232_2	17	32	17	6.31	614	6	14.45	1273	13	16.42	1082	13
246	222232_3	16	34	16	12.06	1007	7	7.22	700	4	8.55	642	9
247	223231_1	13.09	-	14	41.61	3230	28	600	9976	191	600	8596	156
248	223231_2	13	-	-	600	12231	286	600	11827	239	600	9782	148
249	223231_3	14.17	21	15	24.56	1826	14	435.92	8091	119	41.08	1735	28
...													
355	343231_1	13.78	30	-	600	10411	15	600	8012	7	600	8059	10
356	343231_2	11	34	-	600	8546	10	600	6485	8	600	7496	9
357	343231_3	13.17	-	-	600	9412	5	600	7145	6	600	7321	8
358	343232_1	26.45	-	-	600	7856	7	600	6512	6	600	6999	8
359	343232_2	26.37	-	-	600	10111	10	600	8065	8	600	8692	10
360	343232_3	31.62	97	-	600	9641	7	600	7423	6	600	7671	11

Table 2.10: A first comparison between branching schemes

Branch on:	column variables (1)	timetable cells (2)	prec. relations (4)
Nr. solved to optimality	311	319	281
Avg. comp. time*(s)	13.5	18.2	41.5
Avg. comp. time**(s)	110.5	112.1	170.7

Table 2.11: Statistical comparison between branching schemes

p-values	(1) vs (2)	(1) vs (3)	(2) vs (3)
Total set*(s)	0.3491	< 0.0001	< 0.0001
Limited set**(s)	0.3242	< 0.0001	< 0.0001

Table 2.12: Statistical analysis of factors

Branch on:	column variables (1)	timetable cells (2)	prec. relations (3)
Nr. of periods	0.0001	< 0.0001	0.0006
Nr. of trainees	< 0.0001	< 0.0001	< 0.0001
Nr. of activities	< 0.0001	< 0.0001	< 0.0001
Nr. of trainees per activity	< 0.0001	0.0035	< 0.0001
Range	0.0821	0.3315	0.0732
Magnitude of costs	0.9483	0.7669	0.2617

2.4.4 Contributions of speed-up techniques

In order to gain some insight into the contributions of the different speed-up techniques, an experiment was performed including all 307 problems for which an optimal solution was found within 600 seconds for both the first and the second branching scheme. Besides all speed-up techniques (see 2.3.7) the influence of the two alternative ways of column addition (see 2.3.5) was investigated. The results are presented in Table 2.13 and visualized in Figure 2.1.

Table 2.13 contains the average computation times and p-values of the paired T-tests (one-tailed) between the basic setting (including all speed-up techniques) and a specific setting (all speed-up techniques but one). The first row of Table 2.13 contains the results for the basic algorithm. Rows 2 to 7 contain the average computation times when the respective speed-up technique was omitted. Note that the effects are not cumulative, i.e., the algorithm always included all but one speed-up technique. Row 8 gives the computation times when only one column, i.e., the overall best (most negative reduced column), was added after each master optimization. Finally, row 9 contains the computation times when the master was re-optimized after the generation of only one column, instead of one column for each activity. Recall that column elimination is inherent in the second branching scheme.

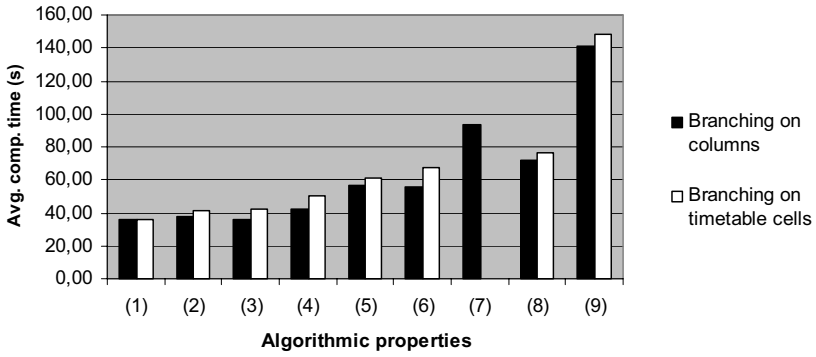

Figure 2.1: Contributions of algorithmic improvements

Table 2.13: Contributions of speed-up techniques

		Branching on columns		Branching on timetable cells	
		Avg. time (s)	p-value	Avg. time (s)	p-value
Basic algorithm	(1)	35.07		35.26	
Without initial heuristic (2.3.7.1)	(2)	37.14	0.2269	40.90	0.0144
Without initial network restriction (2.3.7.3)	(3)	35.88	0.4793	41.52	0.0785
Without Lagrange dual pruning (2.3.7.2)	(4)	41.67	< 0.0001	49.11	< 0.0001
Minimum bounding search strategy (2.3.7.5)	(5)	55.34	< 0.0001	60.17	< 0.0001
Solving master LP starting from empty basis (2.3.7.4)	(6)	54.98	< 0.0001	66.40	< 0.0001
Without column elimination first branching strategy (2.3.7.6)	(7)	92.16	< 0.0001	-	-
Search k columns, add 1 column after each master optim. (2.3.5)	(8)	71.53	< 0.0001	75.33	< 0.0001
Search 1 column, add 1 column after each master optim. (2.3.5)	(9)	139.42	< 0.0001	146.32	< 0.0001

We can draw two conclusions with respect to this experiment. First, the method of column addition plays a major role in fast convergence of column generation. Adding only one (optimal) column after each master optimization seems to be outperformed by adding k (suboptimal) columns after each master optimization. The main reason for the large difference between (8) and (9) is probably the impossibility in (9) to prune nodes based on Lagrange relaxation. Second, the small p-values clearly indicate the positive impact of almost all speed-up techniques. The effect of the initial heuristic and the initial network restrictions could not be confirmed by our test set if one is branching on the column variables. If one is branching on the timetable cells, no significant effect could be detected for the initial network restrictions.

2.5 Conclusions for the decomposition on the activities approach

In the first part, a branch-and-price approach has been proposed in which we decomposed on the activities. The pricing problem could be formulated as a constrained shortest path problem and can be solved efficiently using a forward dynamic programming approach. An important feature of this dynamic program is the ability to find also the 2^{nd}, 3^{rd}, ..., k^{th} shortest path at a very low computational extra cost. This property enabled us to develop a branching scheme based on the column variables. Alternatively, a branching scheme based on timetable cells and a precedence relation based branching scheme have been elaborated. Finally, several speed-up techniques were discussed. In the next part, extensive computational results were presented. An experiment was set up in which the influence of six factors on the complexity of the problem was investigated and the three branching schemes were compared.

Concerning theoretical issues, there are four main conclusions. The first one is that the branch-and-price algorithm for the new formulation clearly outperforms the ILP optimizer applied on the old formulation. The second is that, within the branch-and-price algorithm, branching on the timetable cells and branching on the column variables outperform the branching scheme based on precedence relations. Third, the number of periods, trainees, activities and trainees per activity have an important impact on the computation times, whereas the impact of the magnitude

of the range and the non-availability costs could not be confirmed by our results. Finally, different speed-up techniques are useful in order to improve the performance of the branch-and-price algorithm.

Concerning practical issues, the application makes it possible to find better solutions in less time compared to previous ways of scheduling. To illustrate this, earlier schedules were built for 18 periods. These 18 periods represent 52 weeks (16 3-week periods and 2 2-week periods). If a trainee was not available during a certain week, the full period was made unavailable (for scheduling the difficult activities). The developed application is able to deal with scheduling problems for 52 periods. Also, the formerly seniority based division of weeks off can now be replaced by an approach that takes as much as possible all preferences of all trainees into account. Of course, senior trainees may still be given more priority by assigning to them a larger total amount of non-availability costs.

The presented approach is different from the common column generation approaches to solve staff scheduling problems. Indeed, in early work problems have always been decomposed on the staff members instead of on the tasks. If the considered problem is decomposed on the staff members, we would obtain a set partitioning problem instead of a 0-1 multicommodity flow problem. It would be interesting, of course, to compare our approach with the more traditional decomposition on the trainees approach, and this both for the computational and the formulation issues. This will be the subject of Section 2.6.

Despite all the improvements, the borders of optimality searching within reasonable time were reached when considering problems starting from twelve trainees and ten activities. For the real-life problems solved in this work, this was no serious drawback. There were no trainees that have to perform more than ten different activities within one year. We have encountered real-life problems involving twenty to thirty trainees, however, as these trainees could easily be divided into several subsets each having less than twelve trainees, the exact solution procedure could still be applied on each of these subsets. Nevertheless, for larger problem dimensions, a number of heuristic extensions have to be added to keep the computation time within reasonable limits. We present a number of these heuristic extensions in Section 2.7.3. Another interesting research direction would be to consider a more general problem formulation, e.g., a formulation that includes more general coverage

and formation requirements and handles setup costs explicitly. Setup costs would occur each time a trainee (re)starts a certain activity. In this way, one could search for the optimal trade-off between assigning preferred weeks-off and splitting up activities within trainees. The more general problem formulation as well as the heuristic extensions are the subject of Section 2.7.

2.6 Decomposition on the trainees

In the literature, staff scheduling problems are usually decomposed on the staff members, in this case the trainees, instead of on the activities (see, e.g., Caprara et al., 2003; Mason and Smith, 1998; Jaumard et al., 1998; Bard and Purnomo, 2005b; Mehrotra et al., 2000). An example of such a column for trainee 2 in the problem instance described in Section 2.2 can be found in Table 2.14.

Table 2.14: A column for trainee 2

	Activity schedule			
Period	Trainee 1	Trainee 2	Trainee 3	Trainee 4
1		act 1		
2		act 1		
3		act 1		
4				
5		act 3		
6		act 3		
7		act 3		
8				
9		act 2		
10		act 2		

To decompose (2.1)-(2.9) on the trainees, we introduce decision variables that represent individual trainee schedules. Let binary decision variable z_{jt} be defined as follows:

$$z_{jt} = \begin{cases} 1, & \text{if column } t \text{ was chosen for trainee } j; \\ 0, & \text{otherwise.} \end{cases}$$

Let a_{ijkt} equal 1 if trainee j is scheduled during period i in column t to perform activity k. Let A^j denote the set of all activities that have to be performed by trainee j, i.e., all activities k for which $l_{jk} > 0$. Let c_{jt} be the total cost of column t for trainee j (i.e., $c_{jt} = \sum_{i=1}^{n} \sum_{k \in A^j} a_{ijkt} p_{ij}$) and NC_j the total number of different columns for trainee j. The model can then be formulated as follows:

$$\text{Minimize} \sum_{j=1}^{m} \sum_{t=1}^{NC_j} c_{jt} z_{jt} \tag{2.27}$$

subject to:

$$\sum_{j \in S^k} \sum_{t=1}^{NC_j} a_{ijkt} z_{jt} = 1 \qquad \forall i = 1, ..., n \text{ and } \forall k = 1, ..., p \tag{2.28}$$

$$\sum_{t=1}^{NC_j} z_{jt} = 1 \qquad \forall j = 1, ..., m \tag{2.29}$$

$$z_{jt} \in \{0, 1\} \qquad \forall j = 1, ..., m \text{ and } \forall t = 1, ..., NC_j \tag{2.30}$$

The objective function (2.27) is again the minimization of costs, but now expressed as the sum of the trainee schedules. Constraint set (2.28) states that each activity has to be performed by exactly one trainee at each time period. Constraint set (2.29) implies that exactly one column has to be selected for each trainee. The master problem (2.27)-(2.30) is now a 0-1 set partitioning problem. Let π_{ik} represent the dual prices of restrictions (2.28) and let μ_j represent the dual prices of restrictions (2.29). The reduced cost of a new column t for trainee j is now given by:

$$\begin{aligned} & -\mu_j + c_{jt} + \sum_{i=1}^{n} \sum_{k \in A^j} -\pi_{ik} a_{ijkt} \\ & = -\mu_j + \sum_{i=1}^{n} \sum_{k \in A^j} (p_{ij} - \pi_{ik}) a_{ijkt} \end{aligned} \tag{2.31}$$

2.6.1 Pricing problem

Compared to the pricing problem when decomposing on the activities, which is stated in (2.15)-(2.21), the pricing problem only differs with respect to the first constraint when decomposing on the trainees. Since constraint (2.2) applies at the individual trainee level, this constraint is included in the pricing problem. As the linking constraint is now constraint (2.3), this constraint is left out of the pricing problem. Let x_{ik} equal 1 if trainee j is scheduled to perform activity k during period i and y_{ik} be 1 if trainee j starts activity k at period i. The pricing problem for trainee j can be stated as follows.

$$\text{Minimize} \sum_{i=1}^{n} \sum_{k \in A^j} (p_{ij} - \pi_{ik}) x_{ik} \tag{2.32}$$

subject to:

$$\sum_{k \in A^j} x_{ik} \leq 1 \qquad \forall i = 1, ..., n \tag{2.33}$$

$$\sum_{i=1}^{n} x_{ik} \geq l_{jk} \qquad \forall k \in A^j \tag{2.34}$$

$$\sum_{i=1}^{n} x_{ik} \leq u_{jk} \qquad \forall k \in A^j \tag{2.35}$$

$$y_{1k} = x_{1k} \qquad \forall k \in A^j \tag{2.36}$$

$$y_{ik} \geq x_{ik} - x_{(i-1)k} \qquad \forall i = 2, ..., n \text{ and } \forall k \in A^j \tag{2.37}$$

$$\sum_{i=1}^{n} y_{ik} \leq 1 \qquad \forall k \in A^j \tag{2.38}$$

$$x_{ik}, y_{ik} \in \{0, 1\} \qquad \forall i = 1, ..., n \text{ and } \forall k \in A^j \tag{2.39}$$

Objective (2.32) simply entails the minimization of the (variable part of) the reduced cost (2.31). Constraints (2.33)-(2.39) are just a repetition of the constraint (2.2) and constraints (2.4)-(2.9) for activity k.

The pricing problem can be solved with a dynamic programming approach similar to the one applied when decomposing on the activities. This time the columns of the cost matrix represent the activities. Since it is possible that a trainee performs no activity during certain time periods, an extra column has to be added which represents 'performing no activity' and can be visited more than once. Obviously,

all rows of this column have a cost equal to 0. Table 2.15 visualizes the pricing problem for a particular trainee j which has to perform three activities all between a minimum of one and a maximum of two periods. Each cell of the matrix has a cost h_{ik} which is the difference between the corresponding non-availability cost p_{ij} and the corresponding dual price π_{ik}. Note that the cost values can be negative due to possible positive values for the dual prices π_{ik}. We also applied dynamic programming to solve this pricing problem. The recursive algorithm is very similar as the one outlined above.

Table 2.15: Pricing problem[a] for trainee j: optimal solution in bold

	h_{ik} = non-availability cost p_{ij}- dual price π_{ik}			
Period i	Activity $k=1$	Activity $k=2$	Activity $k=3$	No activity
1	**-2**	1	1	0
2	**-2**	1	1	0
3	4	2	1	**0**
4	2	1	**-1**	0
5	0	**-3**	4	0
6	1	5	3	**0**

[a] For ease of explanation all cost values are integer. Note however that during column generation these cost values are usually fractional due to the dual prices.

Suppose a trainee has to perform p activities. For each time instance we need to store at most all possible subsets of p activities. Hence, the space complexity is given by $O(n \cdot 2^p)$. In the recursive algorithm each state (i, S) leads to at most $R * p * n$ other states. The added factor n, in contrast with the previous subproblem, is due to the fact that the activities do not necessarily immediately follow each other, but instead some periods may be left blanc. The complexity of this recursion is thus $O(n^2 \cdot 2^p \cdot R \cdot p)$.

2.6.2 Branching

Since the computational results presented in Section 2.4.3 indicated that branching on the timetable cells provides the best and most robust results, we will use this

branching scheme. The same branching scheme can be applied if we are decomposing on the trainees. Here, the next x_{ijk} to branch on is also found by selecting the largest fractional column. Suppose this is a column for trainee j. Then, we search for this column the first time period i for which there exists a second fractional column which schedules a different activity than the first column during time period i. Suppose that the first fractional column schedules activity k during the conflicting time period i. Again, x_{ijk} is set to 1 in the left branch and to 0 in the right branch. The timetable costs in the pricing problems are modified as follows. If x_{ijk} is set to 1, $h_{ik'}$ is set to $+\infty$ for all activities $k' \neq k$ in the pricing problem of trainee j. Furthermore, h_{ik} is set to $+\infty$ in the pricing problems of all trainees $j' \neq j$. Else if x_{ijk} is set to 0, h_{ik} is set to $+\infty$ in the pricing problem of trainee j.

Since we have a method to generate columns and a branching scheme to cut away fractional solutions, our branch-and-price algorithm is complete. This algorithm is also extended with the speed-up techniques described in Section 2.3.7. Again, after each master optimization exactly one pricing problem is solved for each trainee. Hence, the dual prices are updated after the addition of at most m columns.

2.6.3 Computational results

2.6.3.1 Two real-life problems

First, the decomposition on the trainees approach has also been applied on the two real-life problems presented in Section 2.4.1. Table 2.16 contains the computational results for both problems. For ease of exposition the results for the original ILP formulation (2.1)-(2.9) are repeated.

Table 2.16: Results on two real-life problems

	Problem 1			Problem 2		
	ILP (2.1)-(2.9)	Decomp. on activities	Decomp. on trainees	ILP (2.1)-(2.9)	Decomp. on activities	Decomp. on trainees
Best solution found	-	16	20	-	10	10
LP relaxation	5.00	15.09	15.00	6.00	9.37	9.30
Explored nodes	453	100	71	390	24	42
Comp. time (s)	1800	78.73	1800	1800	20.89	1036.06

Table 2.16 shows that the best results are obtained when decomposition takes place on the activities. In this approach the first problem could be solved in 78.73 and the second in 20.89 seconds. When one is decomposing on the trainees only the second problem could be solved to optimality within the given time limit.

2.6.3.2 Extensive comparison

The computational performance of both decomposition approaches is also compared using the problem set introduced in Section 2.4.2. In this section, we summarize the most important findings from our computational experiments. The results of the decomposition on the activities approach have already been presented in Section 2.4.3. For ease of exposition, these results are repeated in all Tables stated below.

Table 2.17 and following contain subsets of these results. These tables have the same format as Table 2.9. If the algorithm fails to find an optimal solution within 1800 seconds, 1800 is indicated for the computation time in Table 2.17 and following. Obviously, without this time limit, it would be much easier to make a fair comparison between both decomposition approaches. We could, for instance, look to the total number of nodes or to the total computation time required by each decomposition technique to obtain the optimum for all the instances. Unfortunately, some problems may require days (or even weeks) of computation time, particularly for the decomposition on the trainees. Therefore, we had no option but to set a time limit within which a reasonable amount of the problem instances could be solved by at least one of the decomposition approaches. Fortunately, a time limit of 1800 seconds leads already to important performance differences and hence provides us with a suitable database for comparing both decomposition approaches.

Table 2.17 gives a first indication of how the two decomposition approaches compare to each other. This table contains the summarized results for 36 of the easy instances in our test set. These are problems with 35 periods, 8 trainees and 5 activities in which each activity is performed by 4 trainees. For these dimensions both decompositions manage to find the optimal solution for all problem instances within the time limit. However, the computation times tend to be higher when decomposing on the trainees. Note that one of the explanations of this performance difference can be found in the difference between the LP relaxations. The LP relaxations of the decomposition on the activities approach tend to be higher than

those of the decomposition on the trainees approach.

Table 2.18 contains the same information, but now for 36 problem instances in which the number of trainees having to perform each activity increases from 4 to 6. If we compare this table with the previous one, it becomes clear that the number of trainees having to perform each activity is an important factor for the difficulty of our problem. When decomposing on the trainees, only 13 problems could be solved to optimality within 1800 seconds, compared to 35 problems when decomposing on the activities. When the algorithm failed to solve the problem to optimality, a computation time of 1800 seconds was accounted for the calculation of the average. Even with this underestimation of the computation times for the non-solved problems, there is a clear difference between the average computation times of both decomposition approaches. If we compare the average solution quality, we can conclude that the trainees decomposition, although frequently not capable of detecting the optimal solution, succeeds in finding close to optimal solutions.

When we look at the problem instances with only 18 periods instead of 36 (Table 2.19), we see that all 72 problems could be solved to optimality within the time limit of 1800 seconds when decomposing on the activities. When decomposing on the trainees, the optimum was not found for one problem instance.

If we compare these figures with the results for the problem instances with 52 periods (Table 2.20), we can conclude that also the number of periods is an important factor for the difficulty of the problem. For the largest problems often even no feasible solution could be obtained. In that case the column containing the best found solution (Sol.) reports a "-" and the average solution could not be calculated.

The complexity of the problem also grows with an increasing number of trainees, an increasing number of activities and an increasing magnitude of the range. Compare, for instance, the first six lines with the last six lines in Table 2.19 and Table 2.20.

An overall summary of the computational results is given in Table 2.21. The first row indicates the number of problems that could be solved to optimality within 1800 seconds using each decomposition approach. The second row contains the number of problems for which the decomposition was faster. For the remaining

Table 2.17: Results for problems with 35 periods, 8 trainees and 6 activities in which each activity is performed by 4 trainees

		Decomposition on activities					Decomposition on trainees				
Nr.	Problem	LP relax.	Sol.	Time	Columns	Nodes	LP relax.	Sol.	Time	Columns	Nodes
1	222111_1	13	13	0.20	63	1	13	13	6.84	4095	10
2	222111_2	16	16	0.28	80	2	16	16	1.25	960	1
3	222111_3	16	16	0.25	69	4	16	16	5.23	3126	7
4	222112_1	36	36	0.25	73	2	36	36	3.17	2283	4
5	222112_2	50	50	0.28	84	2	50	50	2.33	1696	4
6	222112_3	45	45	0.11	39	0	45	45	0.56	328	0
...											
30	222152_3	37	37	0.33	94	0	37	37	4.05	1629	1
31	222161_1	12	12	0.67	230	2	12	12	19.73	4184	4
32	222161_2	10.5	11	1.05	358	4	10.5	11	54.33	7331	11
33	222161_3	12.5	13	0.91	338	6	12.4	13	39.44	5006	7
34	222162_1	32	32	0.95	303	1	32	32	37.73	4213	2
35	222162_2	37	37	0.47	134	0	37	37	4.05	992	0
36	222162_3	33.56	35	6.47	1393	67	33.51	35	28.56	5772	9
	Average	24.35	24.86	0.91	279.17	6.86	24.13	24.86	80.60	7837.08	13.83
		Nr. Solved to optimality: 36					Nr. Solved to optimality: 36				

Table 2.18: Results for problems with 35 periods, 8 trainees and 6 activities in which each activity is performed by 6 trainees

		Decomposition on activities					Decomposition on trainees				
Nr.	Problem	LP relax.	Sol.	Time	Columns	Nodes	LP relax.	Sol.	Time	Columns	Nodes
37	222211_1	13	13	1.34	479	10	13	13	333.24	13760	14
38	222211_2	14	14	0.25	78	0	14	14	5.88	800	0
39	222211_3	13	13	1.23	445	6	13	13	1248.62	37182	56
40	222212_1	33	33	1.49	580	10	33	33	223.49	8902	6
41	222212_2	31	31	1.25	458	11	31	31	196.35	12470	14
42	222212_3	26	26	1.39	518	10	26	26	340.72	15642	16
...											
67	222261_1	6	6	18.61	1635	27	6	7	1800	28618	88
68	222261_2	7	8	121.47	3804	127	6.33	9	1800	33093	101
69	222261_3	4.01	5	35.36	2033	108	4	6	1800	32430	75
70	222262_1	14.13	15	9.73	860	7	14	18	1800	30918	70
71	222262_2	20.29	21	14.25	1254	15	20	23	1800	31414	90
72	222262_3	13.98	16	344.36	7687	1198	13.5	20	1800	32367	76
	Average	13.68	14.28	121.20	5083.08	119.61	13.47	15.61	1352.02	27244.61	62.03
		Nr. Solved to optimality: 35					Nr. Solved to optimality: 13				

Table 2.19: Results for problems with 18 periods

		Decomposition on activities					Decomposition on trainees				
Nr.	Problem	LP relax.	Sol.	Time	Columns	Nodes	LP relax.	Sol.	Time	Columns	Nodes
145	111231_1	1	1	0.13	7	0	1	1	0.22	216	0
146	111231_2	6	6	0.01	6	0	6	6	0.09	150	0
147	111231_3	0	0	0	3	0	0	0	0.23	542	2
148	111232_1	1	1	0.03	15	0	1	1	0.14	254	0
149	111232_2	1	2	0.05	55	1	0.8	2	0.39	718	6
150	111232_3	8	8	0	10	0	8	8	0.11	199	0
...											
169	122231_1	3	3	1.84	867	16	3	3	6.36	4510	23
170	122231_2	2	2	1.72	818	14	2	2	83.41	19134	75
171	122231_3	1	1	1.67	798	10	1	1	7.08	4830	20
172	122232_1	9	9	2.47	1191	19	9	9	9.36	5596	25
173	122232_2	8	8	1.81	911	18	8	8	5.94	3622	13
174	122232_3	7	7	1.98	1001	12	7	7	19.42	8956	29
175	123231_1	11	11	8.38	3082	50	11	11	11.63	5646	24
176	123231_2	11	11	4.38	1880	28	11	11	10.48	5362	19
177	123231_3	9	9	5.06	2126	18	9	9	38.38	13570	43
...											
211	143231_1	6	6	51.28	5006	83	6	6	233.32	35572	100
212	143231_2	8	8	51.48	4377	95	8	8	382.77	57272	143
213	143231_3	12	12	38.91	3381	89	12	12	151.41	28600	101
214	143232_1	21	21	65.3	6084	92	21	21	145.31	21854	86
215	143232_2	16	16	44.14	3968	86	16	16	130.83	20002	81
216	143232_3	23	23	55.98	4912	97	23	23	127.49	21238	87
	Average	8.10	8.13	9.60	1434.72	27.00	8.08	8.17	89.50	10686.51	34.29
		Nr. Solved to optimality: 72					Nr. Solved to optimality: 71				

Table 2.20: Results for problems with 52 periods

		Decomposition on activities					Decomposition on trainees				
Nr.	Problem	LP relax.	Sol.	Time	Columns	Nodes	LP relax.	Sol.	Time	Columns	Nodes
289	311231_1	9	9	0.03	9	0	9	9	0.89	374	0
290	311231_2	6	6	0.03	10	0	6	6	0.34	318	0
291	311231_3	4	4	0.03	12	0	4	4	0.5	420	0
292	311232_1	23	23	0.03	8	0	23	23	0.36	310	0
293	311232_2	12	12	0.05	13	0	12	12	0.81	660	2
294	311232_3	14	14	0.05	13	0	14	14	0.56	383	0
...											
313	322231_1	8.15	9	6.53	993	8	8	16	1800	23190	48
314	322231_2	14	14	2.81	407	0	14	18	1800	10879	12
315	322231_3	9.33	10	7.54	1551	19	9.33	13	1800	15588	40
316	322232_1	23	23	3.98	808	4	23	46	1800	8160	7
317	322232_2	20	22	449.42	17250	245	20	46	1800	10262	11
318	322232_3	22.79	25	1800	16469	1102	22	36	1800	10182	14
319	323231_1	22	22	36.31	3953	21	22	100	1800	11799	19
320	323231_2	24	24	14.06	2246	17	24	31	1800	13675	31
321	323231_3	23	23	377.08	15067	89	23	100	1800	7897	11
...											
355	343231_1	13.78	26	1800	21392	206	13.7	-	1800	7326	25
356	343231_2	11	19	1800	20591	193	11	-	1800	7660	25
357	343231_3	13.17	20	1800	15259	82	13	-	1800	7987	21
358	343232_1	26.45	66	1800	21696	242	26.4	-	1800	7108	10
359	343232_2	26.37	-	1800	13151	16	26	-	1800	6958	14
360	343232_3	31.62	53	1800	15114	31	31	-	1800	6632	15
	Average	18.19	-	504.55	6393.38	95.00	18.14	-	1186.15	10000.60	20.01
		Nr. Solved to optimality: 54					Nr. Solved to optimality: 27				

problems, either the computation time was the same or none of both approaches succeeded in solving the problem within the time limit of 1800 seconds. The fourth row indicates the average solution quality for the 340 problems for which both decompositions found at least a feasible solution. For the required computation time, the number of columns and the number of nodes, a distinction is made between the results for all problems and the results for only those problem instances for which both decompositions found an optimal solution within the time limit.

These results clearly indicate that decomposition on the activities outperforms decomposition on the trainees. When decomposing on the activities, more problems could be solved to optimality, average computation times are lower, less columns are needed to prove optimality and more nodes could be evaluated. Moreover, only for two instances no feasible solution was found compared to 20 instances in the trainee-based decomposition approach. If we only look at those instances for which both decompositions found a feasible solution, the average solution quality of the trainee-based decomposition exceeds that of the activity-based decomposition by more than 10%.

If we only look at the problems for which both decompositions found an optimal solution, the number of nodes evaluated in the activity-based decomposition still exceeds those of the trainee-based decomposition. The higher number of nodes in the activity-based decomposition is contradictory with the higher LP relaxations. It turns out that the average is misleading at this point. The last-but-one row in Table 2.21 indicates that the activity-based decomposition could solve more problems in less nodes than the trainee-based decomposition. Hence, there is only a small number of problems for which the number of nodes of the activity-based decomposition dramatically exceeds those of the trainee-based decomposition. As can be expected, this mainly occurs in those few problems for which the LP relaxation is lower. The last row contains the number of problems for which the LP relaxation is lower. For only 12 instances the LP relaxation of the activity-based decomposition is lower compared to 76 for the reverse case. For the other instances, both LP relaxations were equal. 8 out of the 12 instances in which the LP relaxation of the activity-based decomposition is lower have a random number of trainees per activity (setting 4 for factor 4 in Table 2.8) and a small range (setting 1 for factor 5 in Table 2.8).

Table 2.21: Overall summary computational results

	Decomposition on:	
	activities	trainees
Nr. solved to optimality	329	220
Nr. times faster	315	10
Avg. solution value [a]	13.14	14.84
Nr. times feasible solution	358	340
Avg. comp. time (s)[b]	218.32	790.90
Avg. comp. time (s)[c]	19.92	149.24
Avg. nr. columns	4543.21	15591.94
Avg. nr. columns[c]	335.38	607.14
Avg. nr. nodes	120.05	39.82
Avg. nr. nodes[c]	29.16	22.42
Nr. times nr. nodes is lower[c]	126	40
Nr. times LP relaxation is lower	12	76

[a] For only the 340 problems in which both decompositions found at least a feasible solution within 1800 seconds.

[b] A computation time of 1800 seconds was accounted if the algorithm failed to find the optimal solution within the time limit.

[c] For only those 219 problems in which both decompositions found an optimal solution within 1800 seconds.

Decomposition on the trainees resulted in a smaller computation time for only 10 instances. How can we explain this difference? First of all, as already mentioned, the LP relaxation of the root node (thus before branching) tends to be higher if one decomposes on the activities. Since the problem structure does not change after branching, LP relaxations may be expected to exceed those in the trainee-based decomposition approach throughout all nodes of the branch-and-bound tree. Consequently, nodes can be pruned earlier in the activity-based decomposition approach. A second reason why decomposition on the activities is faster than decomposition on the trainees lies in the difference between the master problems. The first master contains m times n 'lower than or equal to' constraints, whereas the second master contains p times n 'equal to' constraints. Remark that all equality

constraints are translated into two inequality constraints. Since $m * n$ is smaller than $2 * p * n$ for all our problems, the master is often solved faster for the activity-based decomposition approach. Third, also the networks in the pricing problems tend to be smaller and thus can be solved faster if one decomposes on the activities.

2.6.4 Modeling power

For the problem we have described in Section 2.2, both decomposition techniques could be applied. Would this still be the case if the problem statement slightly changes? In this section it will be shown that decomposing on the activities can only be applied if the problem has specific characteristics. On the contrary, decomposing on the staff members is a more general approach, since it can be used in a much larger range of staff scheduling problems. This nice property of staff-based decomposition is probably the reason why decomposing on the activities has never been applied in the staff scheduling literature so far.

Most studies in the literature deal with short-term nurse rostering problems (see Section 1.4.3). The demand for nurses generally fluctuates between busy shifts, typically morning shifts, and quiet shifts, typically night and weekend shifts. Also, as there are no formation requirements, the nurses are not required to perform a certain set of different activities within a given time limit. Consequently, for short-term nurse rostering problems, it is not that easy to identify activity patterns and hence decomposition on the activities is less appropriate.

In order to successfully decompose a problem, the question one has to ask is which constraints will be taken care of in the subproblem or similarly, which constraints will remain in the master. The information provided by the dual prices of this last set of constraints should be easily carried over to the subproblem without complicating it too much. To make this point clear, suppose that there are precedence constraints on the order in which the staff members perform their respective activities, i.e., a trainee can only perform a certain activity after (s)he has already performed another activity. This constraint would make it considerably harder to decompose on the activities. Similarly, suppose that the *holes* in the individual trainee schedules (with holes we mean periods in which no activity is scheduled), in one way or another, contribute to the objective function (e.g., one hole of two

periods is preferred to two holes of one period). Whereas this extension could be perfectly addressed in the trainee-based decomposition scheme, it produces serious problems in the activity-based decomposition scheme.

As a third example, observe that in our problem the formation requirement constraints (2.4) and (2.5) for each trainee are automatically satisfied if one selects a schedule (column) for each activity. If it would, however, not be possible to select, for each activity, a set of trainees so that the individual requirement constraints are implicitly satisfied (and thus can be left out of the master), decomposition on the activities would not be suitable. Summarizing, decomposition on the activities is only appropriate if either no constraints (or few) apply at the individual staff member level or, alternatively, if these constraints are automatically satisfied when scheduling activity patterns.

2.7 Generalization of the problem

The problem addressed in this section again involves the construction of 1-year trainee schedules at a hospital department but is a generalization of the problem dealt with in the previous sections. It is shown how a number of additional constraints are easily incorporated in the column generator and how both master and column generator could be made heuristically. Indeed, the generalized problem has a larger solution space which cannot be searched efficiently with an exact approach. Therefore, some heuristic extensions are required to speed up the algorithm at the cost of not guaranteeing optimality. Implementation issues and computational results are given for some real life instances for a trainee scheduling problem encountered at the *Oogziekenhuis Gasthuisberg Leuven*.

2.7.1 General problem statement

First, in the previous section it was assumed that an activity is to be performed by exactly one trainee out of set of trainees having the appropriate skills during each time instance of the time horizon. A more general coverage constraint, however, applies on a specific time horizon. For a particular activity, several coverage constraints could be specified, but each coverage constraint involves only one activity.

Second, in the previous sections it was assumed that the formation requirement constraints (2.4) and (2.5), for each individual trainee, are automatically satisfied if one composes sets of trainees for each activity and schedules the resulting activity patterns. In Section 2.6, this assumption has already been dropped. However, in order to be able to make a fair comparison between both decomposition approaches, all the tested problems still satisfied this assumption.

Third, trainee scheduling problems are often overconstrained, meaning that no feasible solution can be found which satisfies all these constraints. Therefore, we have introduced for each constraint in the ILP stated below a dummy variable except for constraint (2.46), since this constraint can, of course, not be violated. Since we wish to satisfy as many constraints as possible, the objective function of our ILP model is to minimize the total sum of these dummy variables. Because some constraints are more important than others, each dummy could be weighted with a penalty cost.

Let r_{ci}^- and r_{ci}^+ be the number of trainees scheduled too few and the number of trainees scheduled too many in period i for coverage constraint c and p_c^- and p_c^+ their respective associated penalty costs (these costs are assumed to be constant over all periods of the coverage constraint horizon but this is not required for our algorithm). Let L_{jk} and U_{jk} be the strict minimum and maximum number of periods trainee j has to perform activity k as stated in the formation requirements. Let F_{jk} be the target number of periods trainee j has to perform activity k. Make f_{jk}^- denote the positive difference between this target and the actual number of periods in the final schedule. Similarly, let f_{jk}^+ be the difference between the scheduled number of periods and the target number in the reverse case where the number of actual periods exceeds the target number. Make v_{jk}^- and v_{jk}^+ denote the associated penalty costs. Let d_{ij} denote the dummy variable forced to be 1 if trainee j is scheduled to perform an activity at a non-available period i and let w_{ij} denote the respective penalty cost. Finally, b_{jk} equals the number of restarts of activity k by trainee j and q_{jk} is the associated penalty cost. The ILP formulation is given below.

$$\text{Min} \sum_{c \in C} \sum_{i=s_c}^{e_c} (p_{ci}^+ r_{ci}^+ + p_{ci}^- r_{ci}^-) + \sum_{j=1}^{m} \sum_{k=1}^{p} (v_{jk}^- f_{jk}^- + v_{jk}^+ f_{jk}^+ + q_{jk} b_{jk}) + \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} d_{ij} \quad (2.40)$$

subject to:

$$\sum_{j \in T^c} x_{ija_c} - r^+_{ci} + r^-_{ci} = R_c \qquad \forall c \in C \text{ and } \forall i = s_c, ..., e_c \quad (2.41)$$

$$\sum_{i=1}^{n} x_{ijk} - f^+_{jk} + f^-_{jk} = F_{jk} \qquad \forall j = 1, ..., m \text{ and } \forall k = 1, ..., p \quad (2.42)$$

$$\sum_{i=1}^{n} x_{ijk} \geq L_{jk} \qquad \forall j = 1, ..., m \text{ and } \forall k = 1, ..., p \quad (2.43)$$

$$\sum_{i=1}^{n} x_{ijk} \leq U_{jk} \qquad \forall j = 1, ..., m \text{ and } \forall k = 1, ..., p \quad (2.44)$$

$$\sum_{k=1}^{p} x_{ijk} - d_{ij} = 0 \qquad \forall j = 1, ..., m \text{ and } \forall i \in N^j \quad (2.45)$$

$$\sum_{k=1}^{p} x_{ijk} \leq 1 \qquad \forall i = 1, ..., n \text{ and } \forall j = 1, ..., m \quad (2.46)$$

$$y_{1jk} = x_{1jk} \qquad \forall j = 1, ..., m \text{ and } \forall k = 1, ..., p \quad (2.47)$$

$$y_{ijk} \geq x_{ijk} - x_{(i-1)jk} \qquad \forall i = 2, ..., n, \forall j = 1, ..., m \text{ and } \forall k = 1, ..., p \quad (2.48)$$

$$\sum_{i=1}^{n} y_{ijk} - b_{jk} \leq 1 \qquad \forall j = 1, ..., m \text{ and } \forall k = 1, ..., p \quad (2.49)$$

$$x_{ijk}, y_{ijk} \in \{0, 1\} \qquad \forall i = 1, ..., n, \forall j = 1, ..., m \text{ and } \forall k = 1, ..., p \quad (2.50)$$

In this formulation, C represents the set of coverage constraints and a_c, T^c, R_c, s_c and e_c are respectively the activity, the trainee set having the appropriate skills, the number of trainees required and the start and end period of the time horizon of coverage constraint c. Constraint set (2.41) states the coverage constraints. Constraints (2.42)- (2.44) contain the formation restrictions. Constraint set (2.45) implies the non-availability restrictions. Constraint set (2.46) ensures that each trainee performs no more than one activity during each time instance. Constraints (2.47)-(2.49) imply the setup restrictions. The last constraint set (2.50) defines x and y as binary variables. Obviously, the dummy variables cannot be negative.

To solve this problem one could use a decomposition scheme based on the trainees and apply column generation to solve the resulting master LP as has been described in Section 2.6. If the LP is solved to optimality, the solution is driven into integrality using a branching scheme that branches on the timetable cells. This approach

works well for the problem dimensions we have considered (no trainee has to perform more than eight different activities) and if activity split-ups are prohibited. If activity split-ups have to be taken into account, the state space grows exponentially since it requires the introduction of new states which track the number of periods each activity is scheduled instead of merely keeping track whether or not a certain activity is scheduled. Therefore, instead of an exact dynamic programming approach, an approximation algorithm could be applied for pricing out new columns (see Section 2.7.3).

2.7.2 Constraint preprocessing

As has been pointed out in Section 2.6, the column generator takes care of all constraints but the coverage constraints. Accordingly, the master LP only contains the coverage constraints and the added convexity constraints which ensure that exactly one column is chosen for each trainee. Since the number of coverage constraints dramatically exceeds the number of convexity constraints (only one for each trainee), this first set of constraints has a large impact on the computation times of the restricted masters. As a matter of fact, each extra coverage constraint generally tends to complicate the problem, whereas each extra trainee-specific constraint tends to simplify the problem, since it often results in smaller pricing problem networks and/or faster pruning in the column generator. Hence, each coverage constraint that could be transformed into one or more trainee-specific constraints may lead to a significant decrease in required computation time. Therefore, we implemented some simple rules to identify such 'transformable' coverage constraints.

Obviously, coverage constraints that apply on a single trainee can easily be transformed. Second, coverage constraints of capacity type ($=$ or $\leq$) with right hand side value equal to zero can also easily be left out of the master. In the real-life problem we considered, first-year trainees were, for instance, not allowed to perform emergency-related activities during the first semester. Instead of keeping a constraint of the form $x_{11k} + x_{12k} + x_{13k} \cdots + x_{1mk} \leq 0$ for each period $1..\frac{n}{2}$ in the master, it is much more efficient to remove these constraints and incorporate them explicitly in the column generator by removing the corresponding arcs out of the networks. This is called constraint preprocessing. Since it reduces the complexity of

both master and subproblems, constraint preprocessing is an important technique for decreasing overall computation times.

2.7.3 Heuristic extensions

Earlier tests revealed that proven optimal solutions could only be found within reasonable time limits if either the total number of possible columns for each trainee is restricted (i.e., small problem dimensions, see Section 2.6.3.2), or if only a small subset of columns has to be explicitly generated in order to find an optimal integral solution. The latter generally occurs if (a) the master LP relaxation is equal to the optimal IP solution value or (b) if the total set of feasible columns could be divided into two subsets, a 'cheap' set, containing a relatively small number of cheap columns and an 'expensive' set containing all other columns, such that a feasible solution can be found with only columns from the 'cheap' set and the corresponding solution value is smaller than the cost of each column of the 'expensive' set. Consider, for instance, the case in which (part of) the setup costs are higher than the total cost of a feasible schedule. Then, the paths emerging from such an expensive split-up can immediately be pruned in the column generator. If, however, both (a) and (b) are not true, then the algorithm has to be extended with a number of heuristic features to ensure that at least a good (not necessarily optimal) solution will be found. We will successively deal with the following heuristic extensions:

- heuristic algorithm for pricing out new columns;
- premature termination of column generation;
- imbalanced branching;
- combining depth-first and best-first search;
- heuristically fixing x_{ijk} variables.

2.7.3.1 Heuristic algorithm for pricing out new columns

Thanks to the tremendous progress in LP optimization code, the bottleneck of many branch-and-price implementations nowadays mostly lies in the solution of the pricing problems. As has already been mentioned in Section 2.7.1, if activity split-ups have to be taken into account, the state space of the dynamic programming

procedure grows exponentially. Therefore, instead of an exact dynamic programming approach, an approximation algorithm could be applied for pricing out new columns. We make the following assumptions:

- each activity can only be restarted once for each trainee;
- if a trainee j restarts an activity k, or in other words, if the activity is split up into two separate parts, then the first part has to contain at least the minimum requirement L_{jk} of periods.

The first assumption can be justified since the setup penalty costs are usually much higher than the non-availability penalty costs and consequently activities that are started more than twice tend to occur rarely in (sub)optimal schedules. The second assumption can also be justified since it stands for a real-life constraint in many practical situations, namely that only an already experienced trainee can replace another trainee to perform an activity the latter cannot perform for one or two weeks. Moreover, it is not desirable that a trainee, who performs an activity for the first time, already quits it after having performed it for a relatively small number of periods. Analogously to the situation with precedence constraints, whereas this extra constraint can easily be dealt with in the column generator, it would have been very difficult to capture in the pure IP formulation.

Both assumptions entail two interesting properties. First of all, the total state space of feasible schedules (columns) is dramatically reduced for each trainee. Second, the dominance rule as well as the upper bound calculation stated above can still be applied. Nevertheless, generating the best column for certain trainees may still be (too) time consuming (recall that this has to be done many times). Observe, however, that it is not necessary to find optimal columns during the early stages of column generation. Instead, good but not necessarily optimal columns, generated by a heuristic algorithm, can already result into an LP objective value that closely approaches the optimal value. Therefore, the column generator outlined above will be truncated after the exploration of a limited number of feasible paths. At each new pricing iteration, the order in which the activities are considered is determined at random in order to ensure that no large parts of the feasible path state space are completely ignored. If optimality proving were the major concern, the column generator may not be truncated upon convergence of column generation, since only optimal columns can provide the information to determine whether or not the master objective value is solved to optimality. If, however, optimality proving is

of smaller importance, but rather a good feasible IP solution is the major concern, then the information provided by heuristically generated columns can also be used to determine whether or not to stop column generation and to start branching.

2.7.3.2 Premature termination of column generation

Our column generation scheme exhibits the tailing-off effect, i.e., requiring a large number of iterations to prove LP optimality. Instead of solving the linear program to optimality, i.e., generating columns as long as profitable columns exist, we could end the column generation phase prematurely when the master LP value sufficiently approximates the (theoretical) optimum. Therefore, a lower bound on the master LP is required. Therefore we use the Lagrangian lower bound, described in Section 2.3.7.2.

2.7.3.3 Imbalanced branching

As already outlined in Section 2.3.6 branching on the x_{ijk} variables is preferred to branching on the z_{jt} variables when optimality proving is a major concern. If, however, fast detection of a good, feasible solution is the main objective, a more imbalanced (and thus more restrictive in one direction) branching scheme like branching on the z_{jt} variables could be more suitable. Indeed, each left branch (z_{jt} set to 1) fixes a full trainee schedule instead of merely a relatively small subset of arcs in the network. Consequently, feasible integer solutions will be detected much sooner. The counterpart is that it will almost be impossible to prove the optimality of a solution (unless the integral solution objective value equals the LP relaxation). To avoid entering the same column twice, each time the column generator discovers a better column, the new column is first checked against the columns in a forbidden list (i.e., columns set to 0 in the branch-and-bound tree). This can be done quite efficiently since each column can be represented with only four integers using a binary encoding scheme. If this is the case, the second best column can be generated using the algorithm described in Section 2.3.4.

2.7.3.4 Combining depth-first and best-first search

Basically, the branch-and-bound tree is traversed in a depth-first way. The advantage of depth-first is that an integer solution is found early on in the search and hence upper bound pruning can be applied soon. An important disadvantage,

however, is that once an integer solution is found, the algorithm may waste a lot of computation time to improve the solution only slightly. Since we track lower bounds for each node in the search tree, we know the best possible solution value for each node and all nodes below it. If, upon backtracking, the possible improvement, measured by the difference between the nodes lower bound and the current best found solution, is relatively small, we may opt to backtrack one or more levels further until a node is reached for which the possible gain is worth exploring it. In the extreme case this would be a best-first strategy (i.e., the next node to explore is the one with the lowest lower bound). The disadvantages of a pure best-first search are the late detection of an integer solution and the requirement of advanced memory management and sorting capabilities. A mixed approach, combining the advantages of both strategies, turned out to be a good choice for our application.

2.7.3.5 Heuristically fixing x_{ijk} variables

A final heuristic extension involves the fixing of a number of x_{ijk} variables before starting the branch-and-price algorithm. More specifically, a number of 'activity patterns' could already heuristically be scheduled. We refer to an *activity pattern* for activity k from period i_1 until period i_2 as the scheduling of activity k over all time periods between i_1 and i_2 such that exactly one trainee is scheduled at each period. Activity patterns could be identified for all activities for which coverage constraints of type ($=$ or $\geq$) exist. In the previous sections it has been shown how a restricted version of the trainee scheduling problem could be completely decomposed on these activity patterns and solved to optimality with column generation. However, as indicated in Section 2.6.4, the main disadvantage of this approach is that it could only deal with those trainee-specific constraints which are automatically satisfied when scheduling the activity patterns. Moreover, if (part of) the coverage constraints require two or more trainees to be scheduled, the optimality of a solution could not be proven.

The idea is, however, useful to apply in this context. A number of activity patterns could be identified and scheduled heuristically before starting the branch-and-price algorithm. This is done using the greedy heuristic described in Section 2.3.7.1. Prescheduling a number of activity patterns considerably simplifies both the master problem (less coverage constraints) and the pricing problem (smaller networks). The more activity patterns are scheduled (i.e., the more x_{ijk} variables are fixed)

the easier the problem becomes, but also the less likely we are to find an optimal solution. Upon completion of the branch-and-price algorithm, the best solution is saved as an upper bound and the process restarts with either the scheduling of a different set of activity patterns or a different schedule of the same set of activity patterns.

2.7.4 Computational results

In this section we present computational results for a real-life trainee scheduling problem encountered at the department *Oogziekenhuis* of the university hospital *Gasthuisberg*, Leuven, Belgium. The number of trainees of this department varies between 20 and 25. These trainees can roughly be divided into four skill categories (depending on their academic phase). However, exceptions are possible and occur frequently (e.g., a 3^{rd} year trainee having to perform a 2^{nd} year activity). Schedules are built at the start of each academic year and define the workload for each trainee for all periods in the coming year. Since coverage and non-availability constraints apply on a weekly basis and formation requirements are expressed in terms of numbers of weeks, the basic scheduling unit is a week and thus the number of periods equals 52.

In order to simplify the complicated task of the scheduler, current practice includes the aggregation of these 52 weeks in 18 multi-week 'blocks' (16 3-week blocks and 2 2-week blocks). The disadvantage of this approach is that the scheduler is not able to fully exploit all scheduling possibilities. If, for instance, a trainee is not available during a particular week, then the whole block is made unavailable. Once the schedule is built in terms of these blocks, the remaining week (in case of a 2-week block) or remaining two weeks (in case of a 3-week block) of non-available blocks, are filled up with the scheduling of activities with low set-up costs and for which there is sufficient capacity left. Similarly, formation requirements could not be met to the same level of detail as would be the case if schedules are built on a weekly basis. Consequently, the resulting schedules were frequently observed as being unfair and had to go through an extended bargaining process. The total time needed to build the schedule, bargain, rebuild, etc., could easily take about 10 days for an experienced scheduler.

Merely for illustrative purposes, we provide computational results for the 2003-2004 academic year trainee scheduling problem. First, we consider the 18-blocks problem, which could be solved rather easily. Next, we try to solve the 52-weeks problem, which is much more complicated, but allows for the construction of more detailed and qualitatively better schedules for the same problem. For this last problem, we show how the heuristic extensions can help finding a good (better) solution in less time. Table 2.22 summarizes the most important properties of both problems.

Table 2.22: Real-life problem

Problem	Nr. of periods	Nr. of trainees	Avg. nr. of activities for each trainee	Nr. of coverage constraints in the master[a]
1	18	21	6	260
2	52	21	6	720

[a] Exclusive the constraints removed by constraint preprocessing (=+/- 10% of total number of constraints).

All our experiments were performed on a 2.4 GHz Pentium 4 PC with the Windows XP operating system. The algorithm was written in MS Visual C++.NET and linked with the CPLEX 8.1 optimization library. Computational results are given in Table 2.23. The first line of this table indicates that the 18-period problem could be solved to optimality within 2205 seconds. The gap with the optimal LP relaxation is 2%. The gap is defined as $100 * (solution - LP\_relaxation) / (LP\_relaxation)$. The next lines illustrate the impact of the different heuristic extensions on the solution quality. The second column indicates the section numbers of the implemented heuristic extensions. As can be observed, these extensions are implemented in a cumulative way. The computation times were limited to 300 seconds. If we apply the heuristic instead of the exact column generator, the same (optimal) solution was detected. We note that optimality was not proven since (a) the LP relaxations are not proven to be optimal and (b) the branch-and-bound tree still contained

unexplored nodes. If we allow for an LP optimality gap of 2% in each node of the branch-and-bound search tree, the best solution found had a gap of 3.5%.

Next, if the balanced branching scheme is replaced with an imbalanced one (branching on the column variables) the gap increases to 5%. However the time needed to find the first integer solution was decreased significantly with almost 50% (from 40 to 23 seconds). The problem here is that the algorithm wasted a lot of time exploring nodes below a right branch (a particular column variable fixed to 0) only to improve the solution slightly. Strong branching decisions (column variables fixed to 1) made near the root of the search tree could not be undone within the restricted time limit.

When the depth-first search was combined with a best-first search by requiring a minimal possible improvement for exploring a node (as outlined in Section 2.7.3.4), a better solution was detected. Finally, if a number of activity patterns are scheduled heuristically (as outlined in Section 2.7.3.5) an integral solution was already found in less than 10 seconds. However, the algorithm lacked the flexibility to find close to optimal solutions. In the first setting (*) two activity patterns were identified and scheduled heuristically, whereas in the second setting (**) four activity patterns were pre-scheduled (freezing approximately 10% and 20% of the schedule). Afterwards the branch-and-price algorithm was run to solve the remaining problem to optimality.

When the branch-and-price algorithm terminates, the process restarts with a different activity pattern set and/or a different scheduling of the same set. In many cases the branch-and-price algorithm could be terminated as soon as the LP lower bound exceeded the current best found solution. For this relatively small problem the results indicate that the gaps tend to increase with added heuristic extensions. However, the time needed to find a first integral solution decreases.

Let us now turn to the 52-period problem. As indicated in the eighth line of Table 2.23, this problem could not be solved to optimality by the exact branch-and-price algorithm (i.e., without heuristic extensions) within 10 hours of computation time. There was a gap of 18.66% between the best solution found and the optimal solution of the LP relaxation. Since only a relatively small number of nodes in the branch-and-bound tree were explored (254) and we know from experience that the

LP relaxation gap is usually much smaller, there is strong indication that better solutions should be possible. Again, we implemented several heuristic implementations and report on the gaps found. The computation times were limited to 900 seconds.

When we replaced the exact column generator with a heuristic algorithm (Section 2.7.3.1), the gap was reduced by 5.3% (from 18.7% to 13.4%). Taking into account the restricted computation time, many more nodes could be evaluated in the search tree (111 nodes in 900 seconds compared to 254 nodes in 36000 seconds). The main reason for this improvement is the fact that the algorithm suffered less from the so-called 'tailing-off effect' observed in many column generation applications. Tailing-off means that the algorithm keeps finding columns with negative reduced cost, but these columns fail to improve the LP objective. In other words, upon LP convergence, many columns are added merely to *prove* LP optimality, but do not result in a decrease of the LP objective. Recall also that in the exact algorithm the main part of the columns were generated with the heuristic column generator. Only those needed to prove LP optimality had to be generated using the exact column generator. The same reasoning applies if we allow for an LP optimality gap of 2% in each node of the branch-and-bound search tree and the solution could be further improved until 7.3% of the LP relaxation.

Next, if the balanced branching scheme is replaced with an imbalanced one (branching on the column variables), the algorithm was not able to find a better solution. However, the time needed to find the first integer solution was again decreased with almost 50% (from 574 to 324 seconds). When the depth-first search was combined with a best-first search, a better solution could be detected. Finally, if a number of activity patterns are scheduled heuristically an integral solution was already found in respectively 46 and 18 seconds, but the gaps increase again.

Table 2.23: Computational results

Problem	Heuristic extensions	Time LP relaxation (s)	Time first integer solution (s)	Total computation time (s)	LP relaxation gap	Nr. nodes	Nr. columns
1	-	21	56	2205	2.0%	534	22184
1	[2.7.3.1]	17	45	300	2.0%	122	5541
1	[2.7.3.1],[2.7.3.2]	16	40	300	3.5%	143	5126
1	[2.7.3.1],[2.7.3.2],[2.7.3.3]	16	23	300	5%	312	6265
1	[2.7.3.1],[2.7.3.2],[2.7.3.3],[2.7.3.4]	16	23	300	3.1%	278	5963
1	[2.7.3.1],[2.7.3.2],[2.7.3.3],[2.7.3.4],[2.7.3.5]*	3	10	300	7.6%	34	2497
1	[2.7.3.1],[2.7.3.2],[2.7.3.3],[2.7.3.4],[2.7.3.5]**	3	8	300	12.3%	25	1842
2	-	184	922	36000	18.7%	254	28515
2	[2.7.3.1]	111	646	900	13.4%	111	9456
2	[2.7.3.1],[2.7.3.2]	105	574	900	7.3%	154	11414
2	[2.7.3.1],[2.7.3.2],[2.7.3.3]	105	324	900	9.6%	377	12438
2	[2.7.3.1],[2.7.3.2],[2.7.3.3],[2.7.3.4]	105	324	900	5.6%	316	12723
2	[2.7.3.1],[2.7.3.2],[2.7.3.3],[2.7.3.4],[2.7.3.5]*	26	46	900	11.9%	123	8456
2	[2.7.3.1],[2.7.3.2],[2.7.3.3],[2.7.3.4],[2.7.3.5]**	4	18	900	15.3%	88	7458

The section numbers of the implemented heuristic extensions are indicated. In premature termination (2.7.3.2) column generation is ended if the LP optimality gap is smaller than 2%. In imbalanced branching (2.7.3.3) branching occurs on the column variables instead of on the timetable cells. In partial best-first search (2.7.3.4) a minimal possible improvement of 2% is required for exploring a node in the search tree. In heuristic fixing [2.7.3.5]*, two activity patterns are pre-scheduled, freezing approximately 10% of the schedule, whereas in heuristic fixing [2.7.3.5]**, four activities were pre-scheduled, freezing approximately 20% of the schedule.

2.8 Graphical user interface

In this section the graphical user interface (GUI) is presented. The GUI is described using the general trainee scheduling problem studied in Section 2.7. Obviously, as the trainee scheduling problem studied in the preceding sections is a special case of this general problem, it can also be modeled using the same GUI. The language of the GUI is Dutch. The GUI serves three important objectives.

First of all, it allows for easy data input and constraint specification. Non-available periods for instance are specified by simply double clicking on the corresponding timetable cell and entering the associated penalty cost. The non-available period will be marked in red as indicated in Figure 2.2. Figure 2.3 shows how the properties of a trainee are specified. Each activity having to be performed by the trainee is checked and the target, minimum and maximum number of periods as well as deviation penalty costs and penalty costs for activity split-ups can easily be specified. Figure 2.4 shows an example of a coverage constraint. The corresponding activity, time horizon, type ($\leq, =$ *or* $\geq$), required number of trainees, trainee set (skill category) and penalty costs associated with the coverage constraint have to be specified. The importance of easy, intuitive constraint specification is extremely important for the acceptance of the software. If the scheduler were required to state the constraints mathematically (like we did in this chapter), it is very likely that (s)he would prefer the old manual way of scheduling.

A second objective is the visualization of the search process and of course of the found solution(s). A found solution is represented in Figure 2.5. The visualization of the search process greatly helps to understand how the algorithm works and enables to identify certain problems at an early stage during the search. Figure 2.6 indicates how the algorithm is visualized during the run. First, each newly found column is drawn. Second, each branching restriction is indicated by coloring the corresponding timetable cell with the associated activity color.

A third objective of the GUI is to enable the user to fix activity assignments before the start of the algorithm and to modify the found schedule afterwards. This can be done very easily by clicking, dragging and dropping. In the extreme case, the scheduler can try to build the whole schedule in this (manual) way. If at a certain moment the scheduler encounters a schedule conflict, (s)he can make

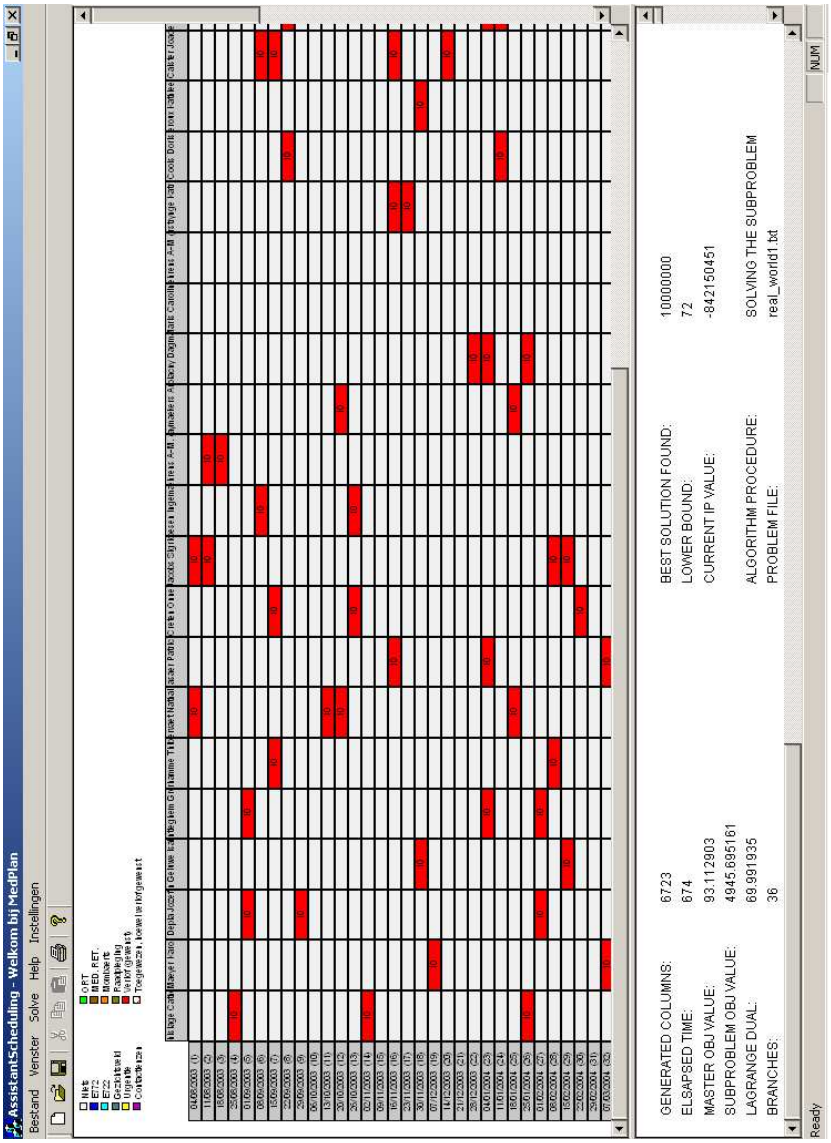

Figure 2.2: GUI: non-available periods indicated in red

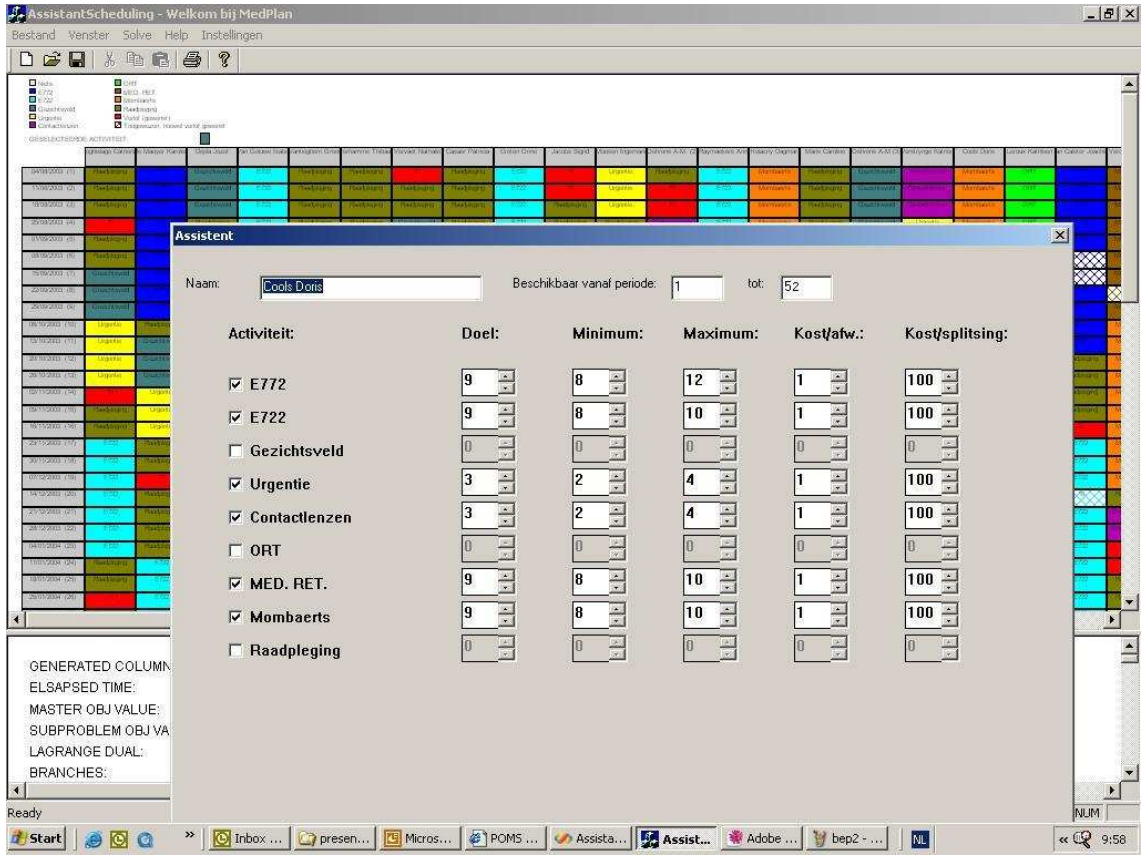

Figure 2.3: GUI: Specifying the properties of a trainee

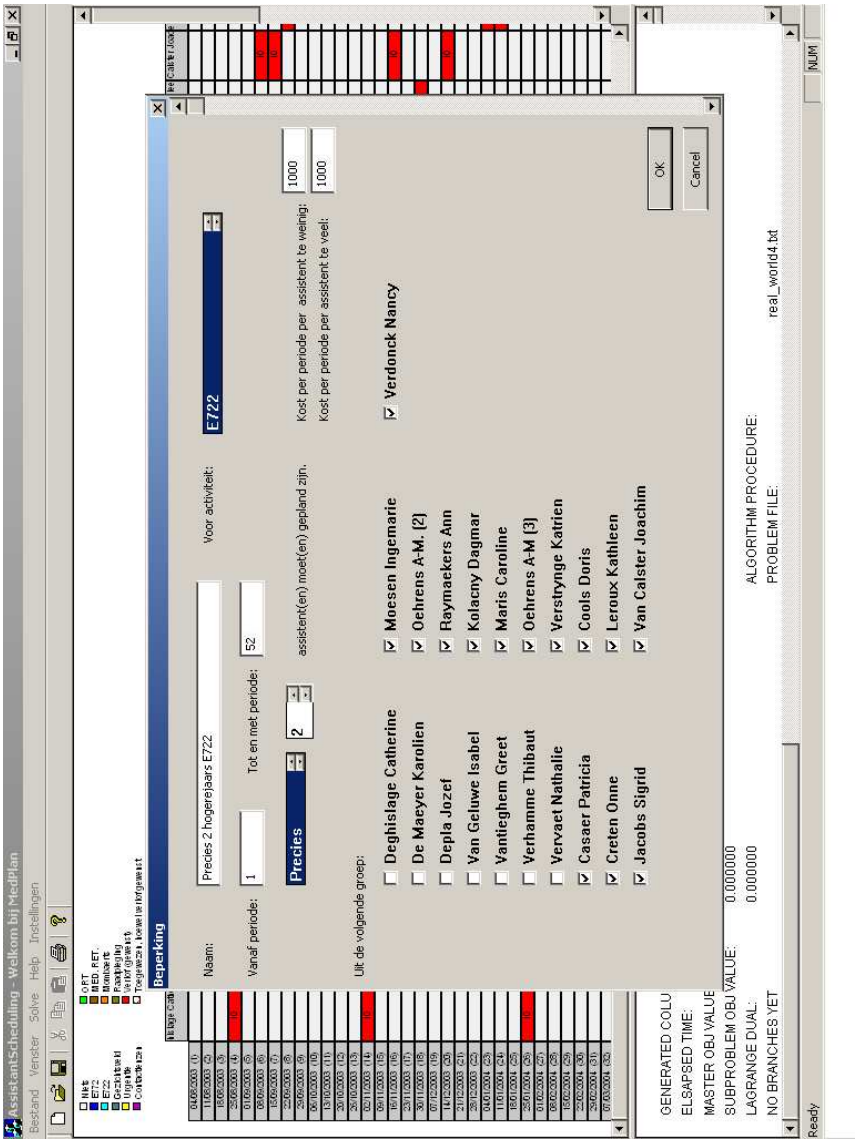

Figure 2.4: GUI: Specifying a coverage constraint

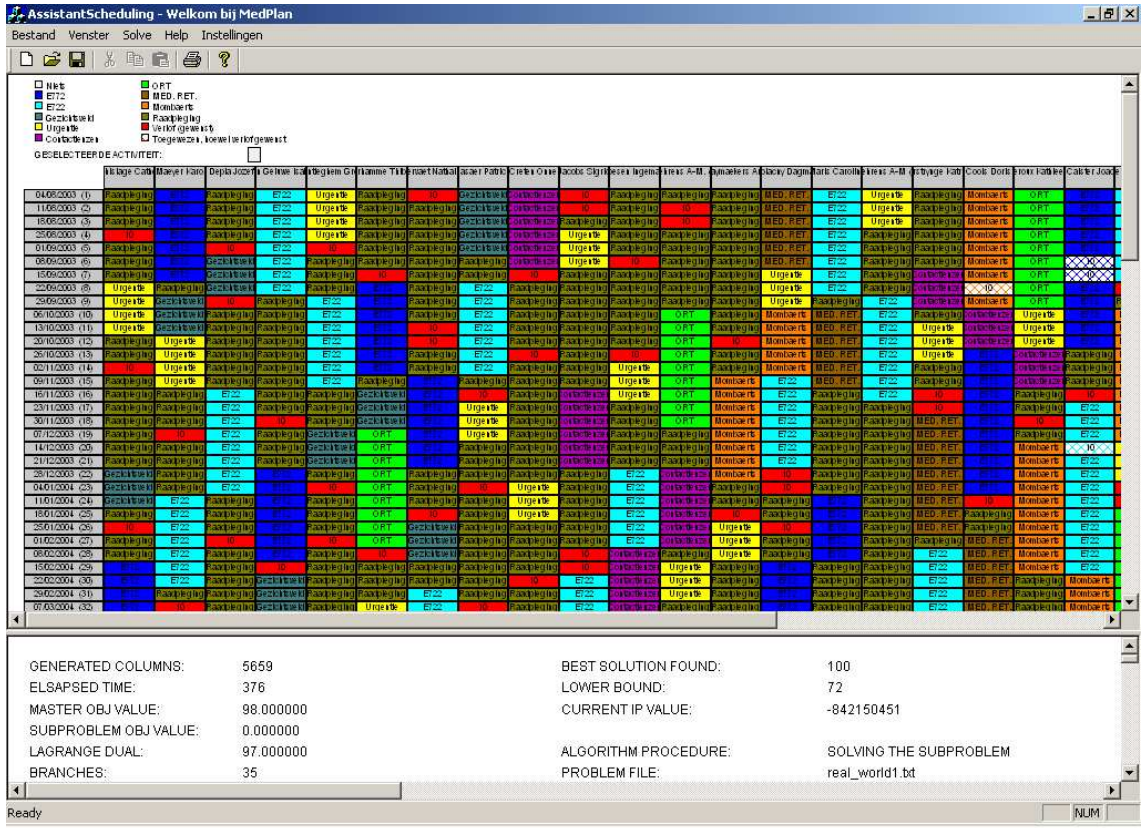

Figure 2.5: GUI: Visualization of a solution

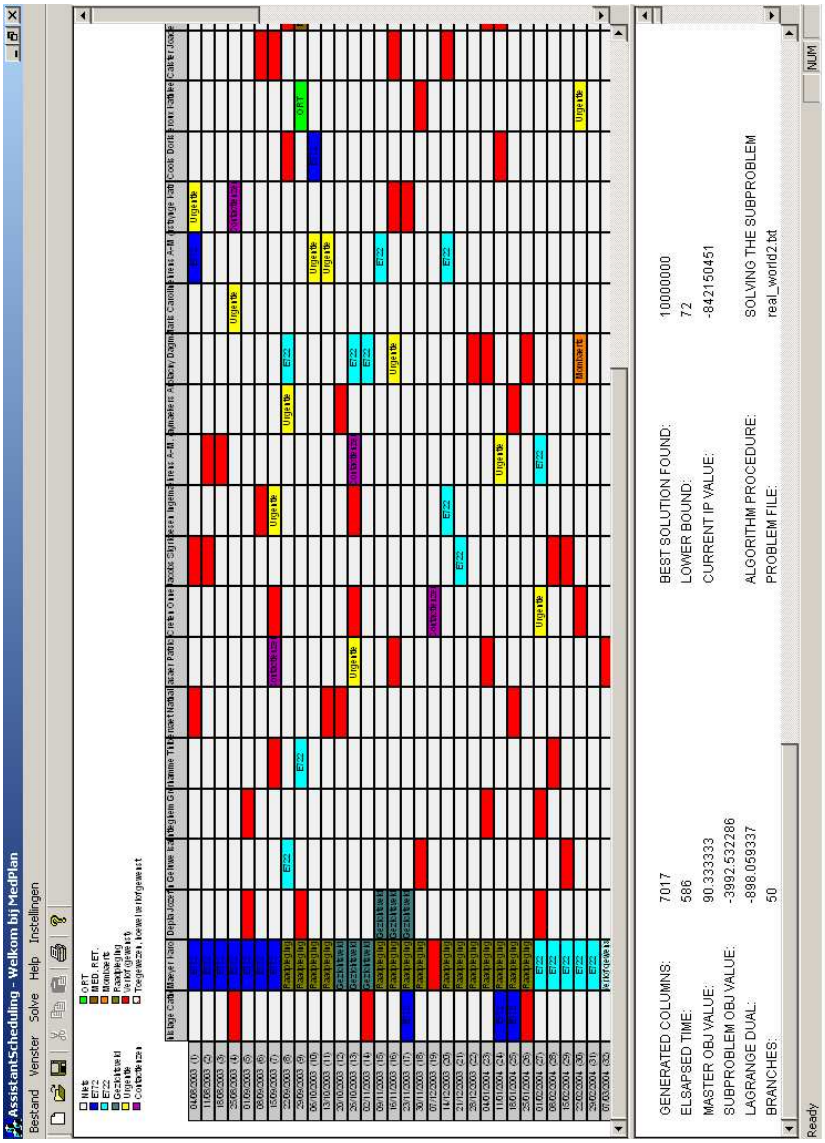

Figure 2.6: Visualization during algorithm run: the newly found column and the branching decisions indicated in color

some assignments undone and run the algorithm to check whether or not a feasible solution is still possible. This would be the case if, for instance, a trainee is required to restart a certain activity because a particular coverage constraint can no longer be satisfied given the partial schedule. This last feature contributes significantly to the willingness of schedulers to accept the software, since it recognizes the fact that they still have the last word. The software only assists in building the schedules, i.e., it helps in solving difficult 'combinatorial puzzles', but the final decisions are still made by (human) scheduler(s) and not by the PC.

2.9 Conclusions and future research

In this chapter, the problem of building long term trainee schedules has been studied. The first part of the chapter deals with a specific class of trainee scheduling problems for which a decomposition scheme on the activities could be applied. Column generation could be applied to find the LP relaxation of the new model and several branching schemes have been proposed to drive the solution into integrality. Extensive computational results have indicated how the different problem dimensions influence the problem difficulty and how the different speed-up techniques contribute to the efficiency of the solution procedure.

Next, the decomposition on the activities approach has been compared with a more traditional decomposition on the trainees approach. The computational tests revealed that decomposition on the trainees is clearly outperformed by decomposition on the activities. The modeling power of both decomposition techniques has also been discussed. Since most staff scheduling problems have a lot of constraints that apply at the level of individual staff members, decomposition on staff members could be used in a wider range of problems. In the rare case that most constraints apply at the level of the activity schedules, decomposition on the activities is more suitable. Activity-based decomposition is also appropriate if each combination of activity schedules automatically satisfies all individual staff member constraints. This was the case for the considered trainee scheduling problem in the first part of this chapter.

The following part has shown how the developed approach can easily be turned into an effective heuristic algorithm. Therefore, five heuristic extensions have been

proposed. The developed application was tested on two real-life problems and computational results are given. These results illustrate how the different heuristic extensions could improve the solution quality if the problem is too complex to find a (proven) optimal solution.

Finally, a graphical user interface (GUI) has been developed. The GUI allows for easy data input, constraint specification and modification of the algorithmic settings. Moreover, certain parts of the schedule could be frozen before the algorithm is run and proposed solutions could be easily modified.

Concerning future research topics, it would be interesting to identify other staff scheduling problems for which decomposition on the activities could be applied. Given the interesting computational properties, this approach could also be suitable to calculate lower bounds for a number of staff scheduling problems for which the above mentioned conditions do not hold. The idea is to remove a part of the individual staff member constraints in order to optimize the relaxed problem using activity-based column generation. Another interesting research direction includes the study of metaheuristic approaches (like, e.g., simulated annealing, tabu search, genetic algorithms or ant colony heuristics) for this problem and see how these compare to the heuristic branch-and-price procedure.

Chapter 3

Visualizing the demand for various resources as a function of the master surgery schedule: A case study

This chapter presents a software system that visualizes the impact of the master surgery schedule on the demand for various resources throughout the rest of the hospital. The master surgery schedule can be seen as the engine that drives the hospital. Therefore, it is very important for decision makers to have a clear image on how the demand for resources is linked to the surgery schedule. The software presented in this chapter enables schedulers to instantaneously view the impact of, e.g., an exchange of two block assignments in the master surgery schedule on the expected resource consumption pattern. A case study entailing a large Belgian surgery unit illustrates how the software can be used to assist in building better surgery schedules.

3.1 Introduction

The operating room can be seen as the engine that drives the hospital as the activities inside the operating room have a dramatic impact on many other activities within the hospital. For instance, patients undergoing surgery are expected to recover over a number of days. Consequently, bed capacity and nursing staff requirements are dependent on the operating room schedule. The software system described in this chapter visualizes the impact of the master surgery schedule on the demand for all kinds of resources like beds, staff (nurses, anaesthetists, etc.), specialized equipment, radiology and so on.

It has been widely accepted that visualization is a simple yet powerful tool for managing complex systems like health care service units. Strum et al. (1997) propose a resource coordination system for surgical services (RCSS) using distributed communications. They present user interfaces that are designed to mimic paper lists and worksheets used by health care providers. These providers enter and maintain patient-specific and site-specific data, which are broadcasted and displayed for all providers. The basic difference between RCSS and our system is that RCSS is designed to work online, preventing and solving resource capacity problems by effective communication, while our system works offline and is designed to facilitate the development of better long term cyclic surgery schedules. Carter (2000) describes the successful application of a commercial package, called ORSOS, which is an enterprise-wide surgery scheduling and resource management system. The system autonomically manages all of the hospitals' surgical staff, equipment and inventory using an engine that considers all of the clinical, financial and operational criteria that must be addressed for each surgical event. The difference with our system is that the emphasis lies on the third stage, the detailed elective surgery scheduling, while our system is designed for the second stage.

Simulation packages are often used to analyze and visualize surgical units. Good surveys of simulation approaches in health care clinics can be found in Klein et al. (1993), Jun et al. (1999) and Standridge (1999). Simulation models that focus on the bed occupancy can be found in Dumas (1984) and (1985) and Wright (1987). A specific simulation model for predicting nursing staff requirements has been described by Duraiswamy et al. (1981). Swisher et al. (2001) highlight the graphical visualization features of their object-oriented simulation package for health care

clinics. The advantage of simulation, compared to our system, is the capability to analyze stochastic processes and to model more complex discrete-event like relationships. The disadvantage is that building a good simulation model is often very time and cost intensive, which makes it less suitable for quickly analyzing simple what-if scenarios, e.g., for assisting in the development of a new cyclic surgery schedule.

In Chapter 1 we have argued that developing operating room schedules can be seen as a three stage process. The model described in this chapter (and also the models presented in the succeeding chapters) is situated in the second stage and as such distinguishes itself from studies situated in the first or the third stage.

The purpose of the system presented in this chapter is threefold. First, schedulers can use it for detecting resource conflicts and constructing workable schedules. Second, the system can greatly assist during the master surgery schedule bargaining process. Visualizing a resource conflict is often far more convincing than hours of discussion with unsatisfied surgeons for not being scheduled by their preferences. Third, the system can be of great value for persuading hospital managers to invest in extra resource capacity. Insufficient resource capacities may not always be visible at first sight. It may, for instance, be the case that, although enough resource capacity is available for the individually summed needs for all resources over all surgeons, still no schedule can be found that provides enough capacity of each resource for each surgeon at each time instance.

The remainder of this chapter is structured as follows. Section 3.2 explains the underlying model. Section 3.3 introduces the surgical unit that is the subject of the case study. Section 3.4 presents the graphical user interface of the software, providing the reader with a visualization of the surgery schedule and its impact on the resource consumption. Finally, Section 3.5 draws conclusions and lists some topics for future research.

3.2 Underlying model

Figure 3.1 contains the underlying model for the visualization software presented in this chapter. On top one can see a number of ovals representing the surgeons (or

surgical groups). Each surgeon obtains a number of blocks in the schedule. Each block allocation consumes a number of resources represented by the grey ovals. With each resource a consumption pattern can be associated that indicates for each time instance how many units are used. These time instances are relative to the moment of surgery. Time instance "0" is during the period of surgery. Time instance "-1" indicates one period earlier, e.g., certain types of surgery require preceding tests. Time instance "1" indicates one period later, e.g., the resources needed while the patient is waking up and recovering from surgery. These resource consumption patterns are indicated by the two-row strings at the bottom of Figure 3.1. The first row contains the time index i, the corresponding cell in the second row gives the required number of units d_i^k for resource k.

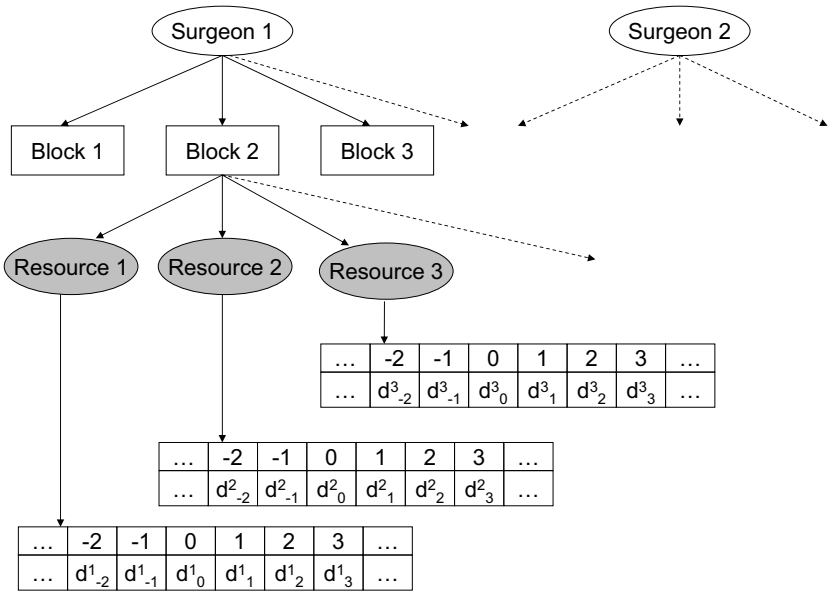

Figure 3.1: Underlying model

In the field of project scheduling, one makes a distinction between renewable and nonrenewable resources (see, e.g., Demeulemeester and Herroelen, 2002). Renewable resources are available on a period-by-period basis, that is the amount is renewable from period to period. Only the total resource use at every time instant is constrained. Typical examples of renewable resources include manpower, equipment, machines, tools and space. On the contrary, nonrenewable resources do not become repeatedly available. Instead, they have a limited consumption availability for the entire duration that the schedule is employed. Money is perhaps the best example of a nonrenewable resource: the overall budget to span a certain time period (e.g., one year) is frequently predetermined to a fixed amount of money.

Only renewable resources could be modeled in the visualization software presented hereafter. The granularity of the time axis may differ from resource to resource and is not necessarily identical to that of the surgery schedule. As non-renewable resources tend to coincide with case mix decision issues, they are left outside the scope of our visualization software.

Observe that the model does not deal with stochastic data: all resource consumption patterns are assumed to be deterministic. In Chapter 4, a theoretical model is proposed that can be seen as a generalization, as well as a particularization, of the model presented in this chapter. It can be seen as a generalization, because it also takes uncertainty into account. The model is, however, also more specific than this one, as beds are the only resource taken into consideration. The model starts from stochastic distributions for patient arrivals and a stochastic length of stay (LOS) associated with each type of surgery. The objective is to obtain a leveled bed occupancy distribution and the master surgery schedule is also the instrument to achieve this objective.

3.3 Case study

This case study concerns the day surgery center of the university hospital Gasthuisberg, situated in Leuven, Belgium. As the name suggests, the day surgery center processes only outpatient admissions. To give an idea of the size of this surgical unit, in 2004 12,778 surgical interventions have been performed, making up for

more than 15,000 hours of total net operating time.

Gasthuisberg's day surgery operating room complex consists of 8 rooms in which, in total, 27 different surgical entities, divided over 13 surgical and medical disciplines, have been assigned operating room time. Each operating room is open from Monday to Friday from 07.45 am till 4.00 pm. No elective surgery takes place during the weekends. Each operating room is allocated for at least half a day to the same surgeon. The current master surgery schedule can be called cyclic since it basically repeats each week with the exception of three block allocations that alter each week between two surgeons.

When building the master surgery schedule one has to take into consideration the impact on several resources. All these resources share the following properties:

- they are limited in capacity,
- they are expensive,
- their consumption pattern depends on the master surgery schedule.

In Gasthuisberg's day surgery operating room complex, twelve such resources could be identified. They can be distinguished in five groups: First of all, certain types of surgery require the patient to be lying and transported in a bed (1). Second, there are the human resources that consist of: three skill-specific groups of nurses (2, 3 and 4), anaesthetists (5) and anaesthetist-supervisors (6). Third, some surgical interventions involve expensive material resources: laporoscopic towers (7), artroscopic towers type 1 (8) and type 2 (9) and lasers type 1 (10) and type 2 (11). Finally, there is the radiology department (12).

3.4 Graphical user interface

In this section the graphical user interface (GUI) is presented. The GUI visualizes the surgery schedule and the resulting bed resource use for a given master surgery schedule. Moreover, it allows the user to modify an existing schedule and view the impact of a change in the schedule on the use of the various resources. Data like the schedule properties, the surgeon properties and the link between the resource

utilizations and the block allocations can easily be read in and modified. Figure 3.2 shows an overview of the GUI with the current surgery schedule for the odd weeks.

The main window is divided into two views. On the left, the master surgery schedule is shown. The columns in the grid represent the time periods from Monday am to Friday pm. The eight rows represent the eight operating rooms X1-X4 and Z1-Z4. Above the grid a legend with the surgical groups is shown. Each surgical group has its own color and style. In this case the style refers to the type of anaesthetic. If the patients are completely anaesthetized during surgery, the surgeon block is colored solidly. Otherwise, when the patients are not fully anaesthetized, the block is arced. The schedule can easily be built from scratch by dragging and dropping the surgeons to the timetable cells.

Each assignment introduces a demand for resources in the system. A subset of these resource utilizations is represented in the right view. Each resource has its own color and time horizon, of which the granularity does not necessarily coincide with that from the surgery cycle time horizon. In our case study, e.g., for the nursing resources on each day an extra time unit is added after the afternoon block. This extra resource unit represents the late shift. Furthermore, for each resource a capacity can be specified that is not necessarily fixed over the total time horizon. In the left view, the scheduler can easily exchange two block assignments by dragging and dropping. In the right view, it will be immediately clear how these changes influence the need for the various resources in the time horizon. In this way the scheduler can quickly detect possible resource conflicts and easily search for workable schedules. Figure 3.3 provides a more detailed view on the resource consumption patterns.

The second, third and fourth resource are groups of nurses, each having a different speciality (respectively “Group 1 NKO”, “Group 1 TRAUMA” and “Group 2”). Each block is colored in proportion to the capacity used. Observe that the need for nurses from “Group 1 NKO” exceeds the indicated capacity on Tuesday and on Friday. This, however, does not necessarily mean that there is a shortage of nurses during these days. The indicated capacities are just leveled targets. When the surgery schedule gives rise to peaks in the demand for nurses, it may be more difficult to schedule the nurses accordingly. In the example shown, nurses have to be shifted from low demand days (Wednesday and Thursday) to peak days (Tues-

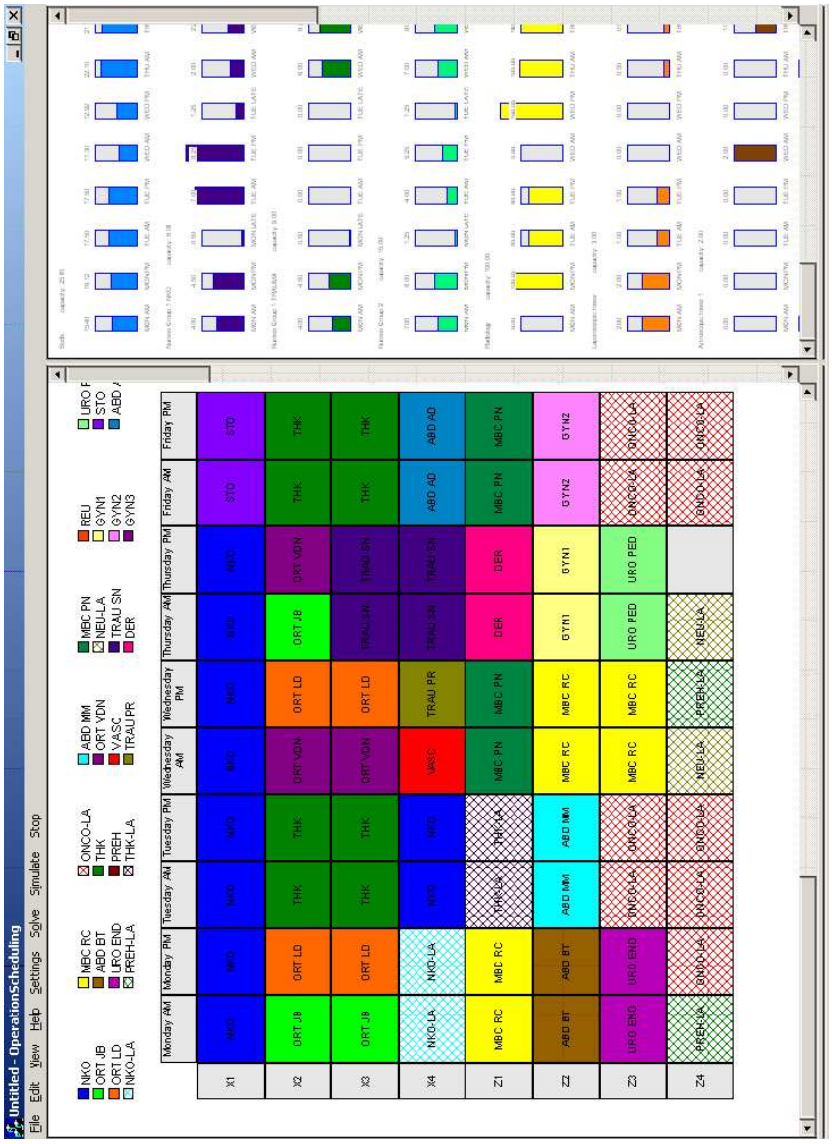

Figure 3.2: Overview of the GUI with current schedule in the odd weeks

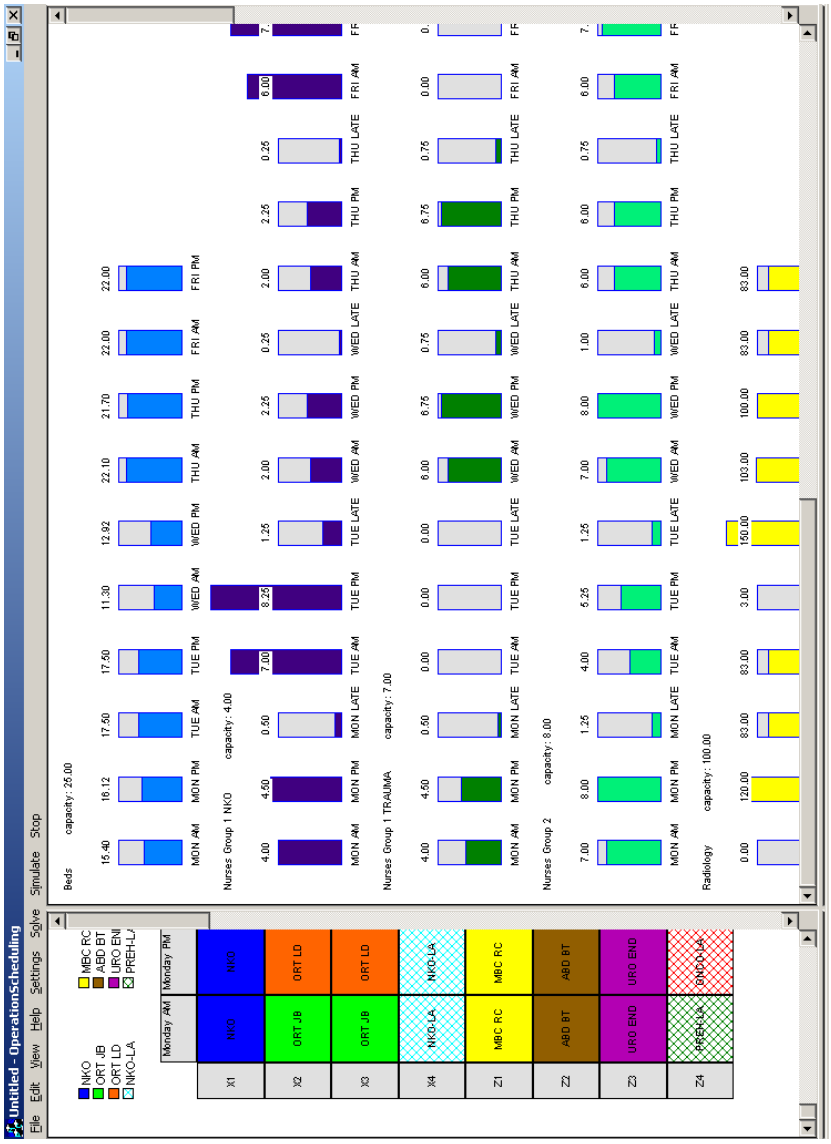

Figure 3.3: A closer view on the resource utilizations

day and Friday). To obtain efficient schedules, it is very important to have a good integration between the nurse scheduling process and the master surgery scheduling process. A specific model and algorithmic solution procedure to realize this integration is proposed in Chapter 6.

Using dialog boxes, the schedule, surgeon and resource properties could easily be modified. As an example some of the dialog boxes for editing the surgeon properties are presented in Figure 3.4. The left dialog box shows the surgeon basic properties and a list of the resources that are consumed by the selected surgeon. The user can select one of these resources to edit. The right dialog box then allows the user to indicate how many units and at what moment in time these resources are used by the surgeon (or surgical group). The time index 0 indicates the starting time of the block allocated to the surgeon. In the example shown in Figure 3.4, two nurses from "Group 1 NKO" are needed to cover the work during surgery time (time index 0) and 1/4 nurse is needed to provide services to operated patients one time period later (pm shift for am surgery or late shift for pm surgery).

The person that is responsible for the operating room schedule of the Gasthuisberg surgical day center evaluated the software during a couple of weeks. His main suggestion for improvement was the ability to have a clear view on all the resources used during each time period given a particular surgery schedule. Accordingly, this feature has been added. Figure 3.5 contains the same schedule, but this time the resource consumption is presented on a 'per day' view instead of on a 'per resource' view. The user can now easily switch between both views, dependent on the information required.

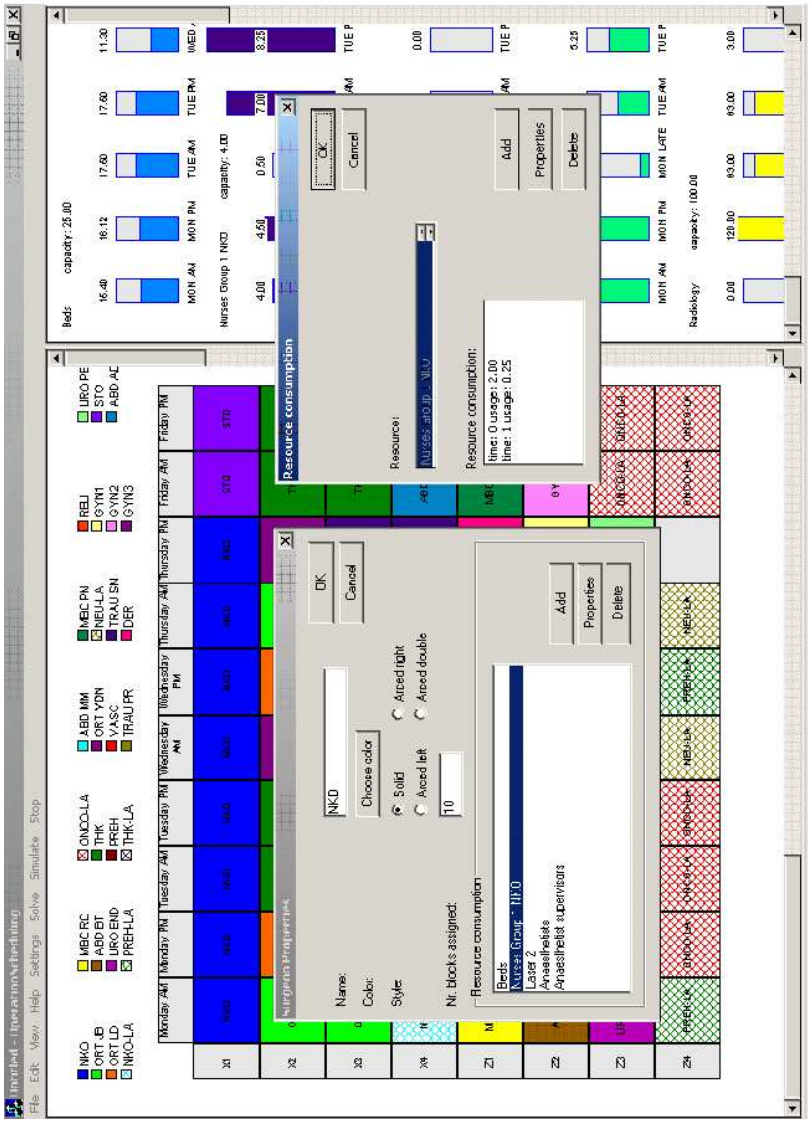

Figure 3.4: Editing the properties of a surgeon

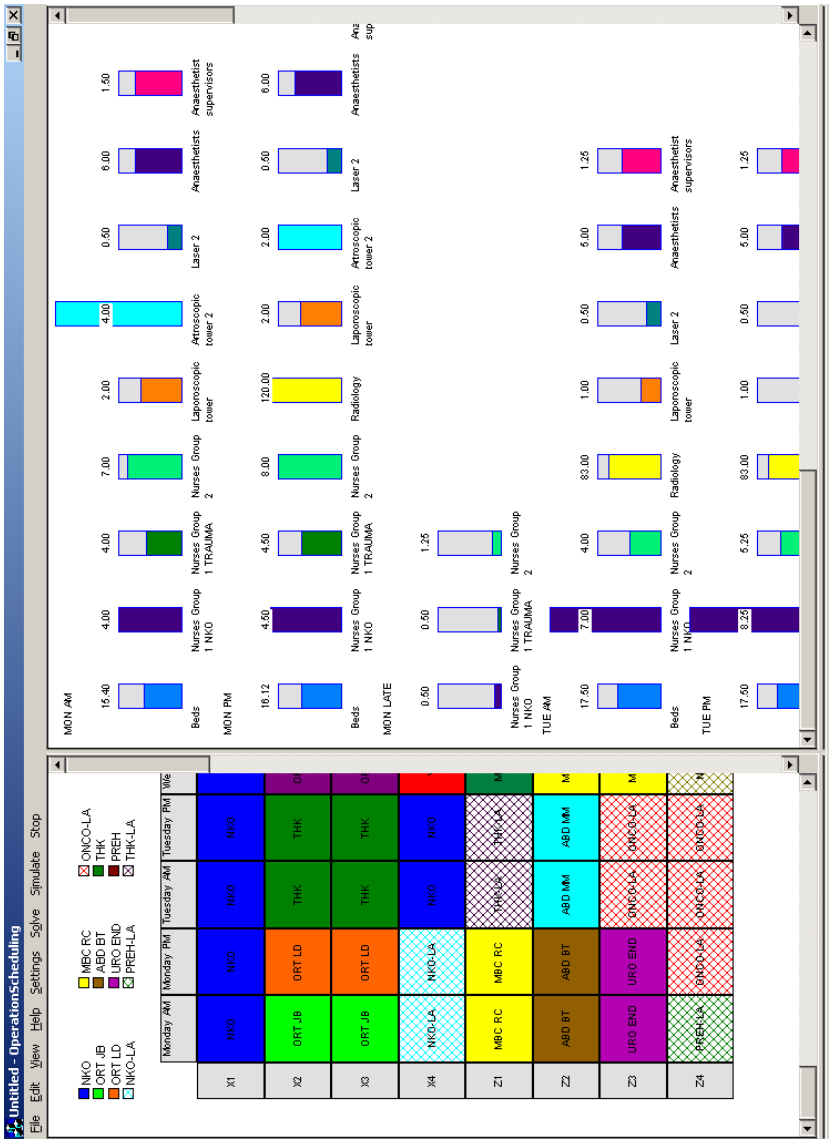

Figure 3.5: Resource consumption on a 'per day' view

3.5 Conclusions and future research

This chapter presents a visualization system for medical surgery units. Given a particular surgery schedule, the system allows for the visualization of the consumption patterns for a variety of resources. Changes in the schedule are immediately reflected in the periodic resource utilizations. The objective of the system is threefold. First of all, it facilitates the detection of resource conflicts and helps the scheduler to develop workable operating room schedules. Second, the system can greatly assist during the master surgery schedule bargaining process. Third, the system can be of great value for persuading hospital managers to invest in extra resource capacity.

The system is designed for the second stage in building surgery schedules which involves the development of a master surgery schedule. It does not provide an online visualization of available and occupied resources during the daily working of a surgery hospital. It is neither a simulation package for analyzing the existing system and a limited number of alternative scenarios. Instead, our system is deterministic and simple. The extremely intuitive graphical user interface makes it very easy to develop high-quality master surgery schedules. To this aim, schedulers can easily switch block allocations and immediately see the consequences on the consumption of various resources on a cyclic time axis.

The model has been extensively tested and evaluated in the surgical day center of a major Belgian university hospital. The system is considered to be very promising for facilitating the development of the master surgery schedule and for improving the efficiency of resource utilization.

In the current version of our software, all resources are of the renewable type and are treated similarly. Resources could, however, further be classified into certain resource categories having similar characteristics. Think, for instance, of resources that can be shared simultaneously by one or more surgeons whilst other resources cannot. Another example are resources with deterministic utilization, that is the load can be predicted accurately, opposed to resources of which the utilization is subject to high uncertainty. The use of equipment is typically deterministic, whereas the bed occupancy is in many cases difficult to predict, due to the uncertainty in the patient's length of stay. It would be interesting to specify several resource categories and enhance the visualization software with dedicated features

per resource category.

The remaining chapters of this dissertation further elaborate the fundamental idea that was presented in this chapter. First, Chapter 4 presents a number of algorithms that focus on the bed occupancy as a critical resource when building surgery schedules. The rather theoretical exposition expands the current model with the introduction of stochastic data in both the number of operated patients per block and the patient's length of stay. Next, in Chapter 5, these algorithms are applied on real-life data coming from a medium-size Belgian hospital. In conclusion, Chapter 6 exploits the relation between the nurse and surgery scheduling process and proposes an integrated approach to simultaneously develop operating room schedules and nurse rosters.

Chapter 4

Building cyclic master surgery schedules with leveled resulting bed occupancy

This chapter proposes and evaluates a number of models for building surgery schedules with leveled resulting bed occupancy. The developed models involve two types of constraints. Demand constraints ensure that each surgeon (or surgical group) obtains a specific number of operating room blocks. Capacity constraints limit the available blocks on each day. Furthermore, the number of operated patients per block and the length of stay of each operated patient are dependent on the type of surgery. Both are considered stochastic, following a multinomial distribution. We develop a number of mixed integer programming based heuristics and a meta-heuristic to minimize the expected total bed shortage and present computational results.

4.1 Introduction

As pointed out by Litvak and Long (2000), while non-elective cases contribute to the huge amount of variability in hospital environments, an important part of the variance, referred to as the artificial variance, can be controlled by applying well-thought-out scheduling policies to elective cases. This idea has already been mentioned in Chapter 1 where we gave an additional exposition on the difference between natural and artificial variability. The algorithms described in this chapter aim at leveling the resulting bed occupancy as a function of the cyclic master surgery schedule.

An important difference with the framework described in Chapter 3 is that this chapter introduces uncertainty. The developed models take into account stochastic numbers of patients per operating room block and a stochastic length of stay for each operated patient. The models enable to build a cyclic master surgery schedule for which the resulting bed occupancy is leveled as much as possible and for which performance measures as the daily expected bed occupancy, the variance on this occupancy, the expected bed shortage and the probability of a shortage on each day can be predicted.

To the best of our knowledge, so far no surgery scheduling models have been proposed that aim at this objective. However, one can distinguish between three classes of papers that are related to our work, but still have important differences.

A first class of papers, e.g., Carter (2002) and Litvak and Long (2000), acknowledges the impact of the surgery schedule on the demand for beds but does not propose concrete models for taking this into account.

A second class of papers deals with models for predicting these occupancies, but does not consider the master surgery schedule as an active tool to optimize the system performance. McManus et al. (2004), for instance, use queuing theory to model the need for critical care resources. They compared predictions from the model to observed performance of an intensive care unit and explored the sensitivity of the model to changes in bed availability that might result from sudden staffing shortages or from admission of patients with very long stays. Gorunescu et al. (2002) also present a queuing model for bed-occupancy management and

planning. Almost all simulation approaches (e.g., Klein et al., 1993; Jun et al., 1999; Harris, 1985) fall into this second class as usually only a limited number of scenarios are being tested. Besides the optimization component, a second important difference between our approach and simulation approaches is the derivation of analytical results, for instance the exact calculation of the mean and the variance of the daily bed occupancy.

A last class of papers considers the bed occupancy as a constraint rather than an objective when building surgery schedules. Blake and Carter (2002), for instance, consider bed availability as a constraint in their goal programming approach to allocate strategic resources in acute care hospitals. A second important difference is that this last work deals with case mix planning (first stage) instead of building master surgery schedules.

We model uncertainty by means of probabilistic distribution functions and optimize expected performance. This way of dealing with scheduling under uncertainty is often referred to as stochastic scheduling. Alternative ways of coping with uncertainty include fuzzy scheduling and robust scheduling. An in-depth discussion on the differences between stochastic and fuzzy approaches to multi-objective mathematical programming under uncertainty can be found in Slowinski and Teghem (1990). In fuzzy scheduling, uncertainty is modeled using the concept of so-called fuzzy sets. A fuzzy set is characterized by a membership function, which maps the members of the universe into the unit interval [0,1]. The value 0 means that the member is not included in the given set, 1 describes a fully included member. The values between 0 and 1 characterize fuzzy members. In the context of uncertainty, such a membership function models the statement of how possible it is for a certain event to occur. A quality exposition on scheduling under fuzziness is provided by Slowinski and Hapke (2000).

Robust or proactive scheduling is concerned with building schedules that are protected against the occurrence of unexpected events. A robust schedule is able to absorb some level of unforeseen events without rescheduling. Accordingly, robust scheduling aims at maximizing the stability of the schedule. Examples of robust scheduling techniques for project scheduling can be found in Leus (2003). Hans et al. (2005) propose several constructive and local search heuristics for the robust surgery loading problem. The objective is to assign the surgeries by the specialties

in such a way, that the risk of working in overtime is minimized, no surgeries are canceled, and at the same time the operating room capacity utilization can be improved.

This chapter proceeds as follows. Section 4.2 gives a general problem statement. Next, Section 4.3 shows how the mean and variance of the daily bed occupancies vary linearly with the decision variables. Section 4.4 develops a number of heuristic algorithms to solve the original problem. We distinguish between algorithms based on the linearized models, a quadratic programming approach and a simulated annealing approach. Section 4.5 provides computational results, while Section 4.6 tries to validate the assumptions made in the models by means of a simulation study. Finally, Section 4.7 states the most important conclusions of this chapter.

4.2 Problem Statement

The problem addressed in this chapter involves the construction of the master surgery schedule. The main objective is to minimize the expected shortage of one resource: beds. To make things clear, we will start with a simple example. We have a surgery schedule divided into a number of time blocks and a number of surgeons. Let us for simplicity suppose that each surgeon only performs one type of surgery. This assumption is not necessary for the hereafter developed algorithms, but is useful to explain the logic behind our model. Our model takes as input stochastic distributions on the number of operated patients and the patients' length of stay. Hence, if we assume one type of surgery for each surgeon, we can associate each surgeon with an ailment (or treatment) and directly transfer the stochastic distributions for this ailment to the surgeon. If surgeons perform different types of surgery we must either introduce 'dummy' surgeons (see Section 5.4) or work with composed distributions having a larger variability.

Furthermore, we assume that the number of patients operated on per time block depends on the type of surgery and that this number is deterministic (this assumption will be relaxed in Section 4.3.6) and fixed for each surgeon. Whereas perfect knowledge is assumed concerning the number of patients undergoing surgery, there is uncertainty concerning the length of stay (LOS) for each patient. The LOS is assumed to follow a multinomial distribution with parameters that depend on the

type of surgery. For instance, a patient recovering from appendix surgery leaves the hospital after 2 days with probability 20%, after 3 days with probability 50% and, finally, after 4 days with probability 30%. If a patient leaves after d days, (s)he occupies one bed for d days starting with the day of surgery. We are concerned with building a cyclic surgery schedule for which the total expected bed shortage (TEBS) is minimized. Cyclic schedules are schedules that are repeated after a certain time period (referred to as the cycle time). During such a cycle time there might be a number of time periods during which surgery cannot take place. These periods are referred to as inactive periods, the others are active. Typically, cycle times are multiples of weeks in which the weekends are inactive periods.

To state the problem mathematically, let x_{is} ($\forall i \in A$ and $s \in S$) be the number of blocks assigned to surgeon s on day i. Here A represents the set of active periods and S the set of surgeons. A block is defined as the smallest time unit for which a specific operating room can be allocated to a specific surgeon (or surgical group). Note that, due to large set-up times and costs, in real-life applications the number of blocks per day in one operating room is usually 1 or 2, i.e each surgical group has the operating room for at least half a day. Hence, considering more blocks can be seen as a way of considering more operating rooms as there is no difference from a computational point of view. Let r_s be the number of blocks required by each surgeon s. These numbers have been determined during case mix planning and are an input for the model. Let b_i be the maximal number of blocks available on day i. Then, our problem could be stated as follows (P1):

$$\text{Minimize } TEBS \tag{4.1}$$

subject to:

$$\sum_{i \in A} x_{is} = r_s \qquad \forall s \in S \tag{4.2}$$

$$\sum_{s \in S} x_{is} \leq b_i \qquad \forall i \in A \tag{4.3}$$

$$x_{is} \in \{0, 1, 2, \ldots, \min(r_s, b_i)\} \qquad \forall s \in S \text{ and } \forall i \in A \tag{4.4}$$

The objective function (4.1) minimizes the expected total bed shortage. Constraint set (4.2) says that each surgeon obtains the number of required blocks. Constraint set (4.3) ensures that the number of blocks assigned does not exceed the available number of blocks on each day. Finally, constraint set (4.4) defines x_{is} to be integer.

Observe that the model does not prohibit surgeons having more blocks on the same day. Hence, a surgeon might be required to operate at the same time in two or more different rooms (in case these blocks overlap in time). To justify this, it is important to keep in mind that surgeons are seen as surgical groups (consisting of more surgeons) rather than individual persons.

Let l be the length of the cycle time. The total expected bed shortage ($TEBS$) equals the sum of the expected bed shortages on each day of the cycle time:

$$TEBS = \sum_{i=1}^{l} EBS_i \tag{4.5}$$

with EBS_i the expected bed shortage on day i. Let U_{ijs} be a stochastic variable representing the number of occupied beds on day i resulting from surgery on day j performed by surgeon s. It can easily be shown that U_{ijs} follows a binomial probability distribution, referred to as $f(U_{ijs})$. Now, let Z_i be a stochastic variable representing the total number of occupied beds on day i. Hence,

$$Z_i = \sum_{s \in S} \sum_{j \in A} U_{ijs} \tag{4.6}$$

The probability distribution of Z_i is given by:

$$f(Z_i = z_i) = \sum_{h \in H^{z_i}} \Big(\prod_{U_{ijs} \in h} f(U_{ijs}) \Big) \tag{4.7}$$

with H^{z_i} the set of all combinations h of U_{ijs}'s summing up to z_i. Let c_i be the capacity of beds on day i. The expected shortage on day i is then as follows:

$$EBS_i = E[f(z_i|z_i > c_i)] = \sum_{z_i=c_i+1}^{\infty} (z_i - c_i) f(z_i) \tag{4.8}$$

Given a certain schedule, we can calculate this expected value. If the total number of combinations leading to a shortage is not too large, we could apply complete enumeration. If complete enumeration is too time consuming, we could calculate approximated values based on the central limit theorem which states that the sum of many independent random variables is approximately normally distributed.

Since EBS_i is not linearly dependent on the decision variables, we cannot find the optimal solution using a mixed integer program (MIP) solver. Therefore, we will try to substitute EBS_i by an expression that is linear in the decision variables, such that it becomes solvable with commercial MIP packages. Of course, we want the new objective to be as equivalent as possible with the real objective.

4.3 Linearization of the problem

4.3.1 Mean

First, instead of dealing with the distribution functions $f(Z_i = z_i)$, we will work with their mean values μ_i. Our assumption is that the larger the difference between c_i and μ_i, the smaller the EBS_i, the expected bed shortage on day i. Without loss of generalization, we assume the bed capacity c_i to be constant for all days i, i.e., $c_i = c$, $\forall i = 1, ..., l$. Our objective is now to minimize the maximal μ_i. This hopefully results in a flat distribution of the expected bed occupancy over all days of the week. In other words, the aim is to level the daily bed resource consumption as much as possible. To state our MIP, we first show that μ_i is linear in the decision variables x_{is}. Let D_{sd} be a stochastic variable representing the number of patients staying in the hospital exactly d days after one block of surgery by surgeon s. We obtain:

$$\begin{aligned}
\mu_i &= E(Z_i) && (4.9)\\
&= E\Big(\sum_{s\in S}\sum_{j\in A} U_{ijs}\Big) && (4.10)\\
&= \sum_{s\in S}\sum_{j\in A} E(U_{ijs}) && (4.11)\\
&= \sum_{s\in S}\sum_{j\in A}\Big(\sum_{d=dist(i,j)}^{m_s} E(D_{sd})\lceil d/l\rceil\Big)x_{js} && (4.12)\\
&= \sum_{s\in S}\sum_{j\in A}\Big(\sum_{d=dist(i,j)}^{m_s} p_{sd}n_s\lceil d/l\rceil\Big)x_{js} && (4.13)
\end{aligned}$$

with $dist(i,j)$ the distance between day i and day j in the week, defined as $i-j+1$ if day j precedes day i and $l+i-j+1$ otherwise, m_s the maximal number of days a patient can stay in the hospital after surgery by surgeon s, p_{sd} the probability a patient stays d days in the hospital after surgery by surgeon s and n_s the number of patients surgeon s can operate in one time block.

Expression (4.13) looks far more complicated than it is. We first note that the mean number of patients of surgeon s staying exactly d days in the hospital equals $p_{sd}n_s$ (mean of a binomial distribution with probability of 'success' p_{sd} and n_s trials). Obviously, a patient that leaves the hospital after d days occupies a bed from day 0 to day $d-1$. Hence, if we consider a particular day i after the day of surgery j we have to sum these expected values starting from the first LOS value reaching day i. This LOS value is given by $dist(i,j)$. For instance, if $i=3$ (Wednesday) and $j=1$ (Monday) we have $dist(i,j)=3-1+1=3$. So, all patients staying 3 days (Monday, Tuesday and Wednesday) or more make up the expected number of patients on Wednesday resulting from surgery on Monday. Obviously, when the LOS exceeds the cycle time l, the corresponding expected number of patients has to be added twice (or more), which explains the factor $\lceil d/l\rceil$.

Since $\sum_{d=dist(i,j)}^{m_s} p_{sd}n_s\lceil d/l\rceil$ is a constant, the new objective is linear in the decision variables. Let μ be the maximal μ_i. We then have the following MIP (MIP1):

$$\text{Minimize } \mu \tag{4.14}$$

subject to:

$$\sum_{i \in A} x_{is} = r_s \qquad \forall s \in S \tag{4.15}$$

$$\sum_{s \in S} x_{is} \leq b_i \qquad \forall i \in A \tag{4.16}$$

$$\mu_i = \sum_{s \in S} \sum_{j \in A} \Big(\sum_{d=dist(i,j)}^{m_s} p_{sd} n_s \lceil d/l \rceil \Big) x_{js} \qquad \forall i = 1, ..., l \tag{4.17}$$

$$\mu_i \leq \mu \qquad \forall i = 1, ..., l \tag{4.18}$$

$$x_{is} \in \{0, 1, 2, \ldots, \min(r_s, b_i)\} \qquad \forall s \in S \text{ and } \forall i \in A \tag{4.19}$$

$$\mu_i \geq 0 \qquad \forall i = 1, ..., l \tag{4.20}$$

$$\mu \geq 0 \tag{4.21}$$

Constraint set (4.17) defines the expected number of occupied beds on each day i. Constraint set (4.18) imposes that μ exceeds each μ_i which ensures that the objective minimizes the maximal expected bed occupancy μ.

4.3.2 Variance

MIP1 aims at a schedule for which the maximal expected bed occupancy is reduced as much as possible over the week. We could however increase the effectiveness of our model by also taking into account the variances of the Z_i variables. Indeed, a schedule resulting from solving MIP1 may exhibit huge differences in the variances of the Z_i's. Figure 4.1 illustrates this point.

In this example we consider a cycle time of 1 week. The expected bed occupancy distribution is quite level over all days of the week. However, the variance of the bed occupancy is much larger on Thursday than on all other days. Consequently, there is a fair chance of running out of beds each Thursday. The question thus arises if it would be possible to include the variance in the objective function of our MIP. Therefore, the variance of the Z_i's must be linear in the decision variables. It can be shown that this is true. For the mathematical derivation we refer the reader to Appendix A.

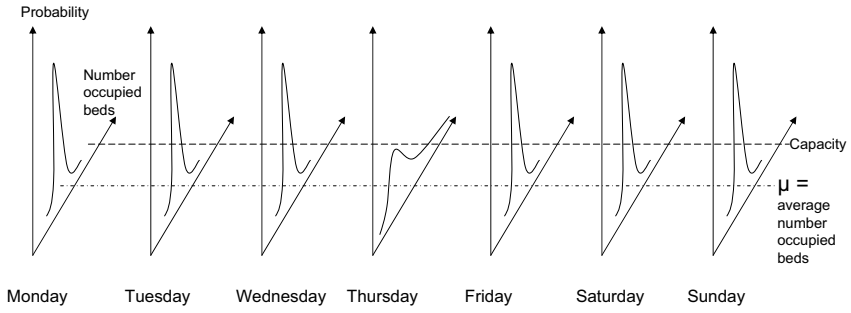

Figure 4.1: The role of variance

The formula is as follows:

$$var(Z_i) = \sum_{s\in S}\sum_{j\in A}\Big(\sum_{d=dist(i,j)}^{m_s} p_{sd}(1-p_{sd})n_s\lceil d/l\rceil - \sum_{d_1=dist(i,j)}^{m_s}\sum_{d_2=dist(i,j)}^{d_1-1} 2p_{sd_1}p_{sd_2}n_s\lceil d_2/l\rceil\Big)x_{js} \tag{4.22}$$

Let us illustrate this with a simple example. Consider the following distribution of the LOS for each patient of surgeon s:

Table 4.1: LOS distribution for example 1

LOS (Nr. of days)	2	3	4	10	11
probability	0.2	0.3	0.1	0.3	0.1

Assume a cycle time of 1 week. For illustrative purposes, we opted for a LOS distribution having a limited number of outcomes and a 'tail' exceeding the cycle

time. Although this example may not seem to be very realistic at first sight, it could represent a scenario in which the operated patients can be divided into two groups: the first group having no complications and leaving the hospital within 4 days and the second group having complications and staying much longer. Assume it is known that this surgeon can operate 10 patients per block. Now, suppose we assign one block on Monday to this surgeon. We illustrate the calculation of $E(U_{3,1,s})$ and $var(U_{3,1,s})$. Let $D_{sd'}$ denote the number of patients staying d days in the hospital who have undergone surgery in the previous week. We obtain:

$$\begin{aligned}
E(U_{3,1,s}) &= E(D_{s3} + D_{s4} + D_{s10} + D_{s11} + D_{s10'} + D_{s11'}) \\
&= E(D_{s3}) + E(D_{s4}) + E(D_{s10}) + E(D_{s11}) + E(D_{s10'}) + E(D_{s11'}) \\
&= E(D_{s3}) + E(D_{s4}) + 2E(D_{s10}) + 2E(D_{s11}) \\
&= 0.3 * 10 + 0.1 * 10 + 2 * 0.3 * 10 + 2 * 0.1 * 10 \\
&= 3 + 1 + 6 + 2 = 12
\end{aligned}$$

$$\begin{aligned}
var(U_{3,1,s}) =& var(D_{s3} + D_{s4} + D_{s10} + D_{s11} + D_{s10'} + D_{s11'}) \\
=& var(D_{s3}) + var(D_{s4}) + var(D_{s10}) + var(D_{s11}) + var(D_{s10'}) + var(D_{s11'}) \\
&+ 2cov(D_{s4}, D_{s3}) + 2cov(D_{s10}, D_{s3}) + 2cov(D_{s10}, D_{s4}) \\
&+ 2cov(D_{s11}, D_{s3}) + 2cov(D_{s11}, D_{s4}) + 2cov(D_{s11}, D_{s10}) \\
&+ 2cov(D_{s11'}, D_{s10'}) \\
=& var(D_{s3}) + var(D_{s4}) + 2var(D_{s10}) + 2var(D_{s11}) \\
&+ 2cov(D_{s4}, D_{s3}) + 2cov(D_{s10}, D_{s3}) + 2cov(D_{s10}, D_{s4}) \\
&+ 2cov(D_{s11}, D_{s3}) + 2cov(D_{s11}, D_{s4}) + 4cov(D_{s11}, D_{s10}) \\
=& 0.3 * 0.7 * 10 + 0.1 * 0.9 * 10 + 2 * 0.3 * 0.7 * 10 + 2 * 0.1 * 0.9 * 10 \\
&- 2 * 0.1 * 0.3 * 10 - 2 * 0.3 * 0.3 * 10 - 2 * 0.3 * 0.1 * 10 \\
&- 2 * 0.1 * 0.3 * 10 - 2 * 0.1 * 0.1 * 10 - 4 * 0.1 * 0.3 * 10 \\
=& 8.3 - 3 - 2 = 3.3
\end{aligned}$$

We extend MIP1 so that the variance is taken into account. The objective is now to minimize a maximal peak defined as a weighted sum of the mean and the variance

of the daily bed occupancy. Let σ_i^2 be the variance of Z_i. Let w_μ and w_{σ^2} be the weight expressing the relative importance of respectively leveling the mean and the variance of the bed occupancy. Consider a one unit increase in the mean occupancy on a particular day. The ratio w_μ over w_{σ^2} indicates the number of units that the variance of the bed occupancy on that particular day should be decreased in order to undo this increase. Let γ be the maximal weighted sum of mean and variance. We then obtain the following MIP (MIP2):

$$\text{Minimize } \gamma \tag{4.23}$$

subject to:

$$\sum_{i \in A} x_{is} = r_s \qquad \forall s \in S \tag{4.24}$$

$$\sum_{s \in S} x_{is} \leq b_i \qquad \forall i \in A \tag{4.25}$$

$$\mu_i = \sum_{s \in S} \sum_{j \in A} \Big(\sum_{d=dist(i,j)}^{m_s} p_{sd} n_s \lceil d/l \rceil \Big) x_{js} \qquad \forall i = 1, ..., l \tag{4.26}$$

$$\sigma_i^2 = \sum_{s \in S} \sum_{j \in A} \Big(\sum_{d=dist(i,j)}^{m_s} p_{sd}(1 - p_{sd}) n_s \lceil d/l \rceil - \sum_{d_1=dist(i,j)}^{m_s} \sum_{d_2=dist(i,j)}^{d_1-1} 2 p_{sd_1} p_{sd_2} n_s \lceil d_2/l \rceil \Big) x_{js} \qquad \forall i = 1, ..., l \tag{4.27}$$

$$w_\mu \mu_i + w_{\sigma^2} \sigma_i^2 \leq \gamma \qquad \forall i = 1, ..., l \tag{4.28}$$

$$x_{is} \in \{0, 1, 2, \ldots, \min(r_s, b_i)\} \qquad \forall s \in S \text{ and } \forall i \in A \tag{4.29}$$

$$\mu_i \geq 0, \sigma_i^2 \geq 0 \qquad \forall i = 1, ..., l \tag{4.30}$$

$$\gamma \geq 0 \tag{4.31}$$

Upon detection of a solution, the weights w_μ and w_{σ^2} can be adjusted in order to obtain a better solution in terms of the (non-linear) objective under consideration. If, for instance, the day with the highest weighted peak has a high variance, it might be useful to slightly increase w_{σ^2} and rerun the MIP optimizer. Preliminary tests indicated that 0.8 and 0.2 for respectively w_μ and w_{σ^2} are good weights for minimizing the total expected bed shortage (TEBS).

4.3.3 $\mathcal{NP}$-hardness proof of the linearized problem

In what follows an $\mathcal{NP}$-hardness proof for problem MIP1 is given. The $\mathcal{NP}$-hardness is proven by means of a transformation from 3-PARTITION. This problem can be described as follows:

3-PARTITION: Given a set $T = \{1, \ldots, 3t\}$ and positive integers $a_1, \ldots, a_{3t}$ and c with $\sum_{j \in T} a_j = tc$, can T be partitioned into t disjoint 3-element subsets T_i such that $\sum_{j \in T_i} a_j = c$ $(i = 1, \ldots, t)$?

This celebrated problem was the first number problem that was proven to be $\mathcal{NP}$-complete in the strong sense. A (very) small problem instance will illustrate this problem: the set T consists of 6 elements with corresponding values of 3, 3, 3, 4, 4 and 5. The values of t and c are obviously 2 (3*2=6 elements) and 11 (3+3+3+4+4+5 = 22 = 2*11), respectively. for this problem instance the answer is positive: T_1 could consist of elements 1, 2 and 6 with corresponding values of 3, 3 and 5, whereas the second set T_2 then consists of the remaining three elements 3, 4 and 5 with values 3, 4 and 4.

Given any instance of the 3-PARTITION problem, an instance of the problem MIP1 can be constructed in the following way:

- The cycle time (l) equals t; there are no inactive days ($A = \{1, \ldots, t\}$).
- The number of blocks per day (b_i) equals 3.
- The number of surgeons equals the number of different values in the set T.
- The number of patients each surgeon s can operate per block (n_s) equals the corresponding value.
- The number of requested blocks per surgeon (r_s) equals the number of times the corresponding value occurs in set T.
- The LOS of the patients is deterministic and equals 1 for each surgeon, i.e. $p_{s1} = 1, \forall s, p_{sd} = 0, \forall s, \forall d \neq 1$.

We show that 3-PARTITION has a solution if and only if there exists a feasible schedule with $\mu = c$.

Suppose that 3-PARTITION has a solution $\{T_1, \ldots, T_t\}$. A feasible schedule with value $\mu = c$ is then obtained as follows. Each set $T_1, \ldots, T_t$ represents an operating day containing 3 blocks at which the surgeons corresponding to the elements in the set are scheduled. The number of patients occupying a bed on each day amounts to c which is the sum of the operated patients during each day. In order to prove the optimality of the solution, we show that $\mu = c$ equals a lower bound. Since each patient stays exactly 1 day, the total LOS over all patients equals the total number of patients, $\sum_s n_s = \sum_{j \in T} a_j = tc$. If we manage to distribute all these patients perfectly balanced over the cycle time, we obtain a solution of $(\sum_{j \in T} a_j)/t = tc/t = c$. It follows that $\mu = c$ is a lower bound to our problem.

Conversely, suppose that there is a feasible schedule with value $\mu = c$. First of all, three blocks must have been assigned at each day, since the total number of requested blocks equals the total number of available blocks (i.e. $\sum_{s \in S} r_s = \sum_{i=1,\ldots,l} b_i = 3t$). By definition, we have for each day i: $\mu_i \leq \mu$. Now, since each patient stays exactly 1 day, the total LOS over all patients equals the total number of patients, $\sum_s n_s = \sum_{j \in T} a_j = tc$. Hence, if the schedule would have a day i for which $\mu_i < \mu$, then there must be another day having a $\mu_i > \mu$. By definition, this is not possible and thus each day must have a μ_i equal to $\mu = c$. Hence, each day $i = 1, ..., t$ represents a set of 3 elements (surgeon-block assignments) with the sum of their values (nr. of operated patients) equal to c. This is a solution to 3-PARTITION. Since MIP1 is a special case of MIP2, MIP2 is also $\mathcal{NP}$-hard in the strong sense. Q.E.D.

4.3.4 Special cases

In this section a number of special cases of model MIP2 are discussed. When w_{σ^2} equals 0, the variance is ignored and model MIP1 will result. When w_μ equals 0, the mean is ignored and MIP2 will minimize the maximal variance of the daily bed occupancy. This means that the resulting bed occupancy may exhibit peaks on certain days of the week. However, these peaks will be easily predictable. This model is appropriate if the resource under consideration can easily be scheduled to anticipate the peak demands. An example of such a flexible resource is non-specialized

manpower or beds merely seen as physical units (material resource). In many hospitals, an institution may have X beds physically available, but only $(X - Y)$ beds actually staffed. The number of staffed beds (involving highly skilled personnel) is generally not adaptable at short term. The model developed in Chapter 6 that explicitly integrates the nurse and surgery scheduling process, is more appropriate to deal with this issue.

The relative importance of w_{σ^2} and w_μ might be dependent on the presence (or absence) of an external stochastic process consuming the resource under consideration. In our example, beds might be occupied by emergency cases. Consider first the situation in which there is no such external process. Suppose we set w_{σ^2} equal to 0 and find an 'optimally' leveled solution. However, given unevenly distributed variances, there are certain days in which there is a fair chance of bed shortage. We might obtain a better solution by slightly increasing w_{σ^2}. Assume that in the new solution, although the mean bed day occupancies exhibit larger differences, the sum of the probabilities of bed shortages is much smaller. Hence, in this situation a positive value for w_{σ^2} is clearly better in the absence of external stochastic processes. However, if we do allow for external processes to consume resources, this conclusion might not hold any more. Since no single model will ever include all sources of variability in hospital environments, this is certainly an interesting point for further research.

4.3.5 Percentile minimization

The variance could be incorporated in a slightly different way. Instead of calculating the true mean and the true variance and minimizing the maximum of the weighted sum, one could directly calculate the contribution of each decision variable x_{is} to some kind of weighted measure. Therefore, we take the contribution to the mean and add n_{stdev} times the square root of the contribution to the variance. For instance, the contribution for x_{js} would be:

$$\sum_{d=dist(i,j)}^{m_s} p_{sd} n_s \lceil d/l \rceil + n_{stdev} \Big(\sum_{d=dist(i,j)}^{m_s} p_{sd}(1-p_{sd}) n_s \lceil d/l \rceil - \sum_{d_1=dist(i,j)}^{m_s} \sum_{d_2=dist(i,j)}^{d_1-1} 2 p_{sd_1} p_{sd_2} n_s \lceil d_2/l \rceil \Big)^{\frac{1}{2}} \tag{4.32}$$

The model is then totally equivalent with MIP1 (4.14-4.21) except for the coefficients of constraint (4.17). Although referred to as percentile minimization, the model does not necessarily minimize the highest percentile peak. Minimizing the highest percentile peak is equivalent to minimizing the highest tail distribution and is a non-linear problem. Instead, we try to measure the contribution of each variable to each day's percentile with a linear weight and solve the problem with a linear optimizer. We choose to take the root of the variance contributions, because standard deviations are more common when referring to distribution tails.

4.3.6 Stochastic n_s

An important drawback of our model is the assumption of deterministic numbers of patients (n_s). In this section we extend our model so that it can handle stochastic n_s's and we prove that it is still possible to express both the mean and the variance as linear combinations of the decision variables.

Introducing stochastic n_s's following a multinomial distribution does not destroy the linearity of both average and variance. Hence, instead of assuming deterministic patient numbers, we can deal with uncertainty: for instance for a particular surgeon the number of operated patients equals 7 with probability 10%, 8 with probability 20%, 9 with probability 40% and 10 with probability 30%. We show how the expressions for both mean and variance are extended so that they incorporate this additional stochastic information. For the mathematical derivation the reader is referred to Appendix B. Let N_s be a stochastic variable representing the number of patients for surgeon s. $k = 1, ..., q_s$ are the different (discrete) states of this variable with h_{sk} being the probability and n_{sk} the corresponding number of patients in state k for patient s. The formulas are as follows:

$$E(U_{ijs}) = E[E(U_{ijs}|N_s)] \tag{4.33}$$

$$= \sum_{k=1}^{q_s} h_{sk}\Big(\sum_{d=dist(i,j)}^{m_s} p_{sd} n_{sk} \lceil d/l \rceil \Big) \tag{4.34}$$

$$\begin{aligned} var(U_{ijs}) = & \sum_{k=1}^{q_s} h_{sk}\Big(\sum_{d=dist(i,j)}^{m_s} p_{sd}(1-p_{sd}) n_{sk} \lceil d/l \rceil \\ & - \sum_{d_1=dist(i,j)}^{m_s} \sum_{d_2=dist(i,j)}^{d_1-1} 2 p_{sd_1} p_{sd_2} n_{sk} \lceil d_2/l \rceil \Big) \\ & + \sum_{k=1}^{q_s} h_{sk}\Big(\sum_{d=dist(i,j)}^{m_s} p_{sd} n_{sk} \lceil d/l \rceil - \sum_{k=1}^{q_s} h_{sk}\Big(\sum_{d=dist(i,j)}^{m_s} p_{sd} n_{sq} \lceil d/l \rceil \Big)\Big)^2 \end{aligned} \tag{4.35}$$

In conclusion, incorporating numbers of patients following a multinomial discrete probability distribution preserves the linearity of both the mean and the variance of the daily bed occupancy. Hence, the above outlined MIP's can perfectly incorporate this source of uncertainty.

4.4 Solving the original problem

MIP2 could be solved with a commercial MIP solver. Preliminary tests indicated that the LP relaxation gap of MIP2 is fairly small. This suggests that it will be difficult to develop a specific (branch-and-bound) algorithm that could solve the problem more efficiently. Moreover, the problem is $\mathcal{NP}$-hard as has been proven in Section 4.3.3. Nevertheless, a number of interesting research questions remain:

1. Do the proposed integer programming models provide good solutions to the original problem P1 (4.1-4.4)?

2. Would it be possible to use these models in order to develop a heuristic that provides better results?

3. Which of the presented models/heuristics is best suited to solve the original problem P1?

4. Is the best choice dependent on certain problem dimensions?

5. How do the results compare to a metaheuristic approach in which the objective function is evaluated directly?

4.4.1 Objective function

To solve P1, it is necessary to evaluate objective function (4.1). To do this, the exact bed usage probability distributions for each day, given a particular surgery schedule, must be found. Unfortunately, computing these general discrete distribution functions involves the enumeration of an exponential number of probability states, which is computationally very hard. Accordingly, we will employ a simplification, making use of the central limit theorem. According to this theorem, each variable which is the sum of a number of independent variables, is approximately normally distributed with mean equal to the sum of the independent means and variance equal to the sum of the independent variances. Recall that the independent means and variances can easily be calculated exactly. Hence, for calculating the shortage probabilities we can simply make use of the standard cumulative normal distribution functions. For calculating the expected shortages we have to apply numerical integration. For instance, to calculate the expected shortage for day i, we compute the following integral:

$$EBS_i \approx \int_{c_i+0.5}^{+\infty} (z_i - c_i) \frac{1}{\sigma_i \sqrt{2\pi}} e^{-\frac{(z_i-\mu_i)^2}{2\sigma_i^2}} dz_i \tag{4.36}$$

This expression sums up all shortages $(z_i - c_i)$ multiplied by the corresponding probabilities. The integral starts at $c_i + 0.5$ (and not at c_i or at $c_i + 1$) to take into account a continuity correction for approaching a discrete function with a continuous one. These integrals were calculated by the numerical integration routines provided in *GNU Scientific Library* (GSL) version 1.3 (Galassi et al., 2003).

In what follows, three heuristics will be presented that aim at the minimization of this objective: a repetitive MIP heuristic, a quadratic MIP heuristic and a local search heuristic (simulated annealing).

4.4.2 Repetitive MIP heuristic

As the name suggests, the repetitive MIP heuristic involves the successive solving of a number of MIP's. After each solution an extra constraint is added to the model, which limits the search space. For the moment we will only concentrate on the averages and thus neglect the impact of the variance. This is motivated by the observation that the average and variance of each Z_i are positively correlated and hence low averages tend to go together with low variances and vice versa. We implemented two repetitive MIP heuristics, to which we refer as REPMIP1 and REPMIP2 respectively.

REPMIP1 works as follows:

1. $TEBS = \infty$.
2. Solve MIP1 (4.14)-(4.21). If the found schedule results in a lower total expected bed shortage, save it as being the best found. Let $\hat{\mu}$ be the optimal objective value and let i be the day with the maximal peak ($\mu_i = \hat{\mu}$), i.e., the day for which the corresponding constraint in constraint set (4.18) is binding.
3. Add an extra constraint to the model: $\mu_i \leq \hat{\mu} + \epsilon$.
4. Make μ_i no longer contribute to the objective function. Therefore, delete $\mu_i \leq \mu$ out of constraint set (4.18).
5. Go back to step 2. Repeat this until a certain stop criterion is met.

The idea is that after the minimization of the highest peak, the second highest peak is to be minimized, while the peak of the highest day is kept below a certain limit. Next, the third highest peak is minimized with constraints on the first two peaks and so on. . . . The value of ϵ determines to which amount the previous peak(s) can be exceeded. If ϵ equals 0, the search space is limited most from MIP to MIP. ϵ can be made dependent on the progression of the algorithm. The solution of each MIP provides a surgery schedule which could be evaluated by calculating the total expected bed shortage (TEBS), for which we do a number of numerical integrations (4.36). The best schedule is saved.

REPMIP2 works as follows:

1. $TEBS = \infty$.
2. Solve MIP1 (4.14)-(4.21). If the found schedule results in a lower total expected bed shortage, save it as being the best found. Let $\check{\mu}_i$ be the lowest bed occupancy peak and let i be the day with this minimal peak.
3. Add an extra constraint to the model: $\mu_i \geq \check{\mu}_i + \epsilon$.
4. Solve the adapted model. If the found schedule results in a lower total expected bed shortage, save it as being the best found.
5. Increase the right hand side value of the constraint, added in step 3 with ϵ over the current usage of beds on day i.
6. Go back to step 4. Repeat this until a certain stop criterion is met.

The idea is that after the minimization of the highest peak, the lowest peak is identified. Next, the model is resolved with an extra constraint which prohibits the current solution by implying an increase in the lowest peak. The aim is that the overcapacity in this lowest peak is divided over all other days but the peak day. ϵ determines to which amount the previous off-peak(s) has to be exceeded. A typical value for ϵ is 0.01. Typical end criteria include the detection of an infeasible model and/or the peak of the lowest day exceeding a certain limit (e.g., the overall average bed occupancy). The solution of each MIP provides a surgery schedule which could be evaluated by calculating the total expected bed shortage (TEBS), for which we do a number of numerical integrations (4.36). The best schedule is saved.

The repetitive MIP models can be seen as non-archimedean (or preemptive) weighted goal programming approaches (see, e.g., Blake and Carter (2003)). An important difference is, however, that in these repetitive MIP models the objective function optimized by the separate MIPs serves only as a guideline for the real (non-linear) objective. After each MIP optimization, the solution is evaluated in terms of the TEBS objective and only when this solution is better, the solution is being saved. Moreover, given a strictly positive value for ϵ, a higher priority goal (minimizing the highest peak) may be degraded in favor of a lower priority goal (minimizing the second highest peak). This is the case when the best found solution contains a

peak which is higher than the maximal peak found by solving one or more of the individual MIPs. Hence, it might be perfectly possible that a solution found by solving one of the early MIPs is weaker (in terms of TEBS) than a solution found by one of the later MIPs. However, since the solution space is very complicated, it is impossible to state that the TEBS solution value arising from solving a particular MIP will always be dominated by a solution found by solving another particular MIP.

4.4.3 Quadratic MIP heuristic

In this heuristic, variances are ignored and only the means are taken into account. We solve again a MIP, however the objective function is now quadratic (QMIP):

$$\text{Minimize } \sum_{i \in A} \mu_i^2 \tag{4.37}$$

subject to:

$$\sum_{i \in A} x_{is} = r_s \qquad \forall s \in S \tag{4.38}$$

$$\sum_{s \in S} x_{is} \leq b_i \qquad \forall i \in A \tag{4.39}$$

$$\mu_i = \sum_{s \in S} \sum_{j \in A} \Big(\sum_{d=dist(i,j)}^{m_s} p_{sd} n_s (\lfloor d/l \rfloor + 1) \Big) x_{js} \qquad \forall i = 1, ..., l \tag{4.40}$$

$$x_{is} \in \{0, 1, 2, \ldots, \min(r_s, b_i)\} \qquad \forall s \in S \text{ and } \forall i \in A \tag{4.41}$$

$$\mu_i \geq 0 \qquad \forall i = 1, ..., l \tag{4.42}$$

$$\mu \geq 0 \tag{4.43}$$

Since $\sum_{i \in A} \mu_i$ is constant and hence independent of the surgery schedule, this model explicitly tries to level the peaks as much as possible. Note that the minimization of $x_1^2 + x_2^2$, subject to $x_1 + x_2 = a$ results in $x_1 = x_2 = \frac{a}{2}$. Note also that $\sum_{i \in A} \sigma_i^2$ is constant and hence independent of the surgery schedule, thus we might also take into account the variances. This might be appropriate for resources for which explicit leveling of the variances is important. For minimizing the total expected bed shortage, which can be seen as "leveling the distribution functions", preliminary results indicated that leveling the variances results in poor solutions.

Hence, the quadratic MIP has only been tested with respect to the averages. Again, we evaluate the resulting surgery schedule by calculating the objective function by computing a number of integrals (4.36).

4.4.4 Simulated annealing

Simulated annealing (SA) is a technique to find a good solution to an optimization problem by trying random variations of the current solution. A worse variation is accepted as the new solution with a probability that decreases as the computation proceeds. The slower the cooling schedule, or rate of decrease, the more likely the algorithm is to find an optimal or near-optimal solution. This technique stems from thermal annealing which aims to obtain perfect crystallizations by a slow enough temperature reduction to give atoms the time to attain the lowest energy state. The search tries to avoid local minima by jumping out of them early in the computation. Towards the end of the computation, when the temperature, or probability of accepting a worse solution, is nearly zero, this simply seeks the bottom of the local minimum. The chance of getting a good solution can be traded off with computation time by slowing down the cooling schedule. The slower the cooling, the higher the chance of finding the optimum solution, but the longer the run time. Thus effective use of this technique depends on finding a cooling schedule that gets good enough solutions without taking too much time. The algorithm is based upon that of Metropolis et al. (1958), which was originally proposed as a means of finding the equilibrium configuration of a collection of atoms at a given temperature. The connection between this algorithm and mathematical minimization was first noted by Pincus (1970), but it was Kirkpatrick et al. (1983) who proposed that it forms the basis of a search technique for combinatorial (and other) problems. Good theoretic expositions on simulated annealing can also be found in Huang et al. (1986) and Van Laarhoven and Aarts (1988).

A basic SA implementation is used. Our neighborhood is defined as all solutions which could be obtained after swapping two surgery blocks from the current solution. The first block is chosen randomly. The second block is the first encountered block for which a swap results in an improvement (decrease) of the objective value. If no such block can be found, the block leading to the smallest increase is chosen. Since swaps between one surgeon and swaps between one day have no impact on the objective function, these swaps are not taken into account. In order to decide

whether or not to accept a worse solution, a standard Boltzman function is evaluated. Let T denote the temperature and Δf the decrease in objective function. For swaps with negative Δf the probability of acceptance is given by $e^{\frac{\Delta f}{T}}$. Of course, the best found schedule is saved.

The advantage of SA over the previous two methods is that the true objective can immediately be evaluated. In contrast, the repetitive MIP heuristic optimizes a series of linear objective functions which hopefully result in a schedule that minimizes the true objective. Similarly, the quadratic MIP heuristic evaluates a quadratic objective instead of the true objective. The main drawback of SA is that experiments are required to find good values for T and the temperature decrease function. The probability of a worse solution being accepted should be large at the start of the search and small towards the end.

4.5 Computational experiment

4.5.1 Test set

To study the computational performance of the heuristics, a test set was composed. All test problems involve a cycle time of 7 days in which the last two days are not available to allocate operating room time (weekend). Seven factors were identified that could have an impact on the complexity of the problem. These are: (1) the number of time blocks per day, (2) the number of surgeons, (3) the division of requested blocks per surgeon, (4) the number of operated patients per surgeon, (5) the probability of a no show as a measure of the variability in this number, (6) the length of stay (LOS) distribution and finally (7) the bed capacity. If we consider two settings for each factor and repeat each factor combination 3 times, we obtain $2^7 * 3 = 384$ test instances. Table 4.2 contains the settings for these seven factors. Some of the factor settings require some further explanation.

The number of blocks per day is drawn from a uniform distribution with bounds 3 and 6 in the first setting and 7 and 12 in the second setting. The third factor indicates whether or not the requested blocks are evenly distributed among all surgeons; e.g., if there are 20 time blocks and 5 surgeons, each surgeon requires 4 time blocks in the evenly distributed case, whereas in the unevenly distributed

Table 4.2: Design of experiment

Factor setting	Nr. blocks per day	Nr. surgeons	Division req. blocks	Nr. patients per surgeon	Prob. no show	LOS	Capacity
1	3-6	3-7	evenly distributed	3-5	5%	2-5	105%
2	7-12	8-15	not evenly distributed	3-12	10%	2-12	110%

case huge differences can occur. Factor 5 defines the probability of a no show. The higher this probability, the higher the variability in the number of operated patients distribution for each surgeon. For the LOS in factor 6 we simulated exponential distributions (made discrete by use of binomial distributions) with mean dependent on the factor setting. Finally, the capacity was set as follows. First we calculate the total bed occupancy, i.e., sum up all (expected) LOS days of all (expected) patients of all surgeons. This number was divided by 7 in order to obtain the absolute minimum required capacity. Next, depending on the factor setting this capacity was increased with 5 or 10%.

4.5.2 Tested heuristics

The heuristic algorithms summarized in Table 4.3 have been tested on these 384 test instances. Preliminary tests indicated that SA2 needs very large computation times. The reason is that the evaluation of the true objective (via numerical integration) is very time consuming. Therefore, a third SA heuristic (SA3) was implemented in which the objective is a weighted sum of the squared average daily bed occupancies (as in QP) and the total shortage probability. This new objective can be evaluated instantly and hence many more iterations of SA can take place. Since the total squared sum of daily average bed occupancies is much larger than the total shortage probability, this first measure is normalized so that it falls in a range from 0 (minimum) to 1 (maximum). The end criterion of SA3 is the same as in SA2 (1000 iterations). Additionally, a new heuristic was written in which the

start solution is given by the solution found by the QP heuristic followed by 250 iterations of SA (QP+SA). Here, the evaluation function is again the true objective.

4.5.3 Computational Results

The heuristics were implemented in Visual C++ and linked with CPLEX 8.1 (ILOG, 2002) as a callable optimization library to perform linear and quadratic optimization. All our experiments were performed on a 2.4 GHz Pentium 4 PC with the Windows XP operating system. Table 4.4 contains the results of our experiment. This table contains average values (over all 384 test instances) for the total expected bed shortages (TEBS) and average values and standard deviations

Table 4.3: Tested heuristics

Abbrev.	Description
MINMU	Minimize average peak (=MIP1)(4.14-4.21)
MINWEIGHTED	Minimize weighted peak
	(=MIP2 with $w_\mu = 0.8$ and $w_{\sigma^2} = 0.2$) (4.23-4.31)
REPMIP1	Repetitive MIP model 1 with ϵ=1% of previous peak
REPMIP2	Repetitive MIP model 2 with ϵ=0.01
QP	Quadratic Programming model (4.37-4.43)
MINMUPERC	Same as MINMU, but now based on percentiles (see 4.3.5)
	$n_{stdev} = 0.2$
REPMIP1PERC	Idem for REPMIP1
REPMIP2PERC	Idem for REPMIP2
QPPERC	Idem for QP
SA1	Simulated annealing (4.4.4)
	objective=min. total expected shortage
	initial temperature=500
	temperature update interval=10 iterations
	temperature update function=0.95*previous temperature
	end criterion=max. time all previous heuristics
SA2	See SA1
	except for end criterion=1000 iterations
SA3	See SA2
	except for objective=QP and total shortage probability
SA+QP	SA with end criterion=250 iterations and starting solution from QP

for the computation times (in milliseconds). The standard deviations give an indication of the variability of the computation times for each heuristic.

Figures 4.2 and 4.3 visualize this table. Note that the Y-axis in Figure 4.3 has a logarithmic scale. From these figures we can draw a number of conclusions. First of all, if we look at the expected shortages, we see that SA2 and SA+QP find the best solutions (no significant differences), followed by REPMIP2, REPMIP2PERC, QP and QPPERC. Additionally, a repeated measures analysis was done with SAS to draw well-founded conclusions. F-tests (Type III) on the different contrasts indicated that the solutions found by SA2 were significantly ($\alpha = 0.05$) better than those found in REPMIP2, REPMIP2PERC, QP and QPPERC, between which no significant difference could be found. The results from REPMIP1 and REPMIP1PERC are significantly worse than the previous four heuristics. Finally, MINMU, MINWEIGHTED, MINMUPERC and SA1 performed significantly worse than all previous heuristics, but again no significant differences could be found between them. SA3 performs significantly worse than all other heuristics.

With respect to the computation times, four groups can be distinguished (from smallest to largest computation time): (1) the quadratic MIP heuristics (QP and QPPERC), (2) the single MIP heuristics (MINMU, MINWEIGHTED and MINMU-

Table 4.4: Computational results

Heuristic	Avg. exp. shortage (TEBS)	Avg. comp. time (ms)	St. dev. comp. time (ms)
MINMU	9.346	76.510	228.334
MINMUPERC	9.362	78.518	405.307
MINWEIGHTED	9.219	86.174	195.423
REPMIP1	7.853	17833.776	321353.376
REPMIP1PERC	7.941	3981.865	42299.126
REPMIP2	7.278	2624.906	5145.229
REPMIP2PERC	7.278	2168.659	3888.637
QP	7.312	27.951	20.946
QPPERC	7.464	26.443	19.029
SA1	9.536	22109.503	323544.954
SA2	6.698	56808.042	20574.437
SA3	11.513	230.773	87.025
QP+SA	6.740	12386.804	4858.314

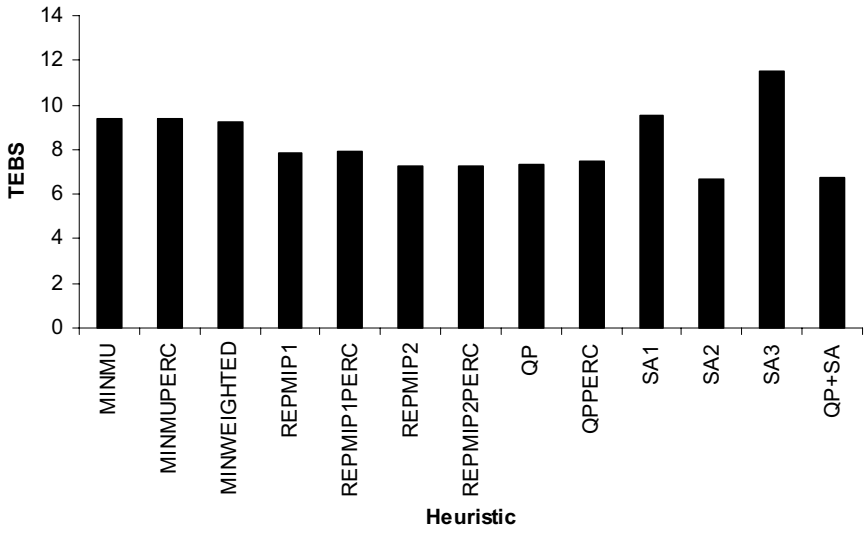

Figure 4.2: Comparison heuristics results

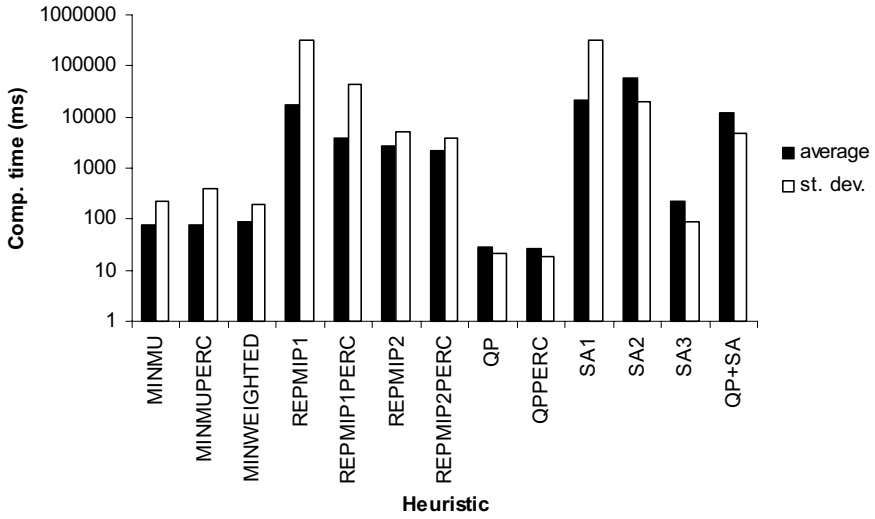

Figure 4.3: Comparison of heuristic computation times

PERC) and SA3, (3) the repetitive MIP heuristics (REPMIP1, REPMIP1PERC, REPMIP2 and REPMIP2PERC), SA1 and SA+QP and (4) SA2. Recall that the computation time given to SA1 equals the largest of the MIP heuristics and hence, SA1 is obviously situated in the third group. From the standard deviations it may be concluded that the computation time of REPMIP1 and SA2 are highly variable. Analyzing computation times in SAS yielded no significant difference between QP and QPPERC. The quadratic MIP heuristics outperform all other heuristics, although no significant differences could be found with REPMIP1, REPMIPPERC and SA1, due to the large variability in these data.

The most important conclusion is that the SA approach outperforms the MIP approaches in terms of solution quality, but is outperformed with regard to the needed computation time. Therefore, when both solution quality and computational effort are important, SA initiated with the solution found by a QP, seems to be the most appropriate solution approach. A second important observation is that the quadratic MIP heuristics dominate the repetitive MIP heuristics with regard to both solution quality and computational effort.

The impact of the different factor settings on the computation time is dependent on the applied heuristic. The effects have been tested using F-tests (type III). Table 4.5 provides the p-values of the different factors for each heuristic. Significant factors ($\alpha = 0.05$) are indicated with a *.

It is possible to distinguish between five groups. For the first group, consisting of the single MIP heuristics (MINMU, MINMUPERC and MINWEIGHTED), only the first two factors (the number of blocks per day and the number of surgeons) have a significant (positive) impact on the computation time. Due to the huge variability in computation times, no significant factors could be found for REPMIP1 and REPMIP1PERC. Since SA1 gets the largest computation time of the MIP heuristics, this heuristic obviously also belongs to this second group. REPMIP2 and REPMIP2PERC are situated in a third group for which a third factor becomes significant: the number of patients per surgeon. Also here there is a positive influence on the computation time. For the quadratic MIP heuristics (QP and QPPERC) yet another significant factor is added: the LOS (Length Of Stay of the patients). The influence of this factor is however negative. Hence, the longer the patients stay, the smaller the needed computation time to solve the quadratic

Table 4.5: Impact of factor settings: p-values

Factor	Nr. blocks	Nr.	Division	Nr. patients	Prob.	LOS	Capacity
Heur.	per day	surgeons	req. blocks	per surgeon	no show		
MINMU	0.0233*	< .0001*	0.2996	0.3956	0.0966	0.7456	0.8820
MINMUPERC	0.0903	0.0069*	0.6391	0.1143	0.2167	0.2060	0.3018
MINWEIGHTED	0.0142*	< .0001*	0.3489	0.5962	0.2792	0.1506	0.2724
REPMIP1	0.2945	0.2826	0.3144	0.3118	0.3322	0.2987	0.3093
REPMIP1PERC	0.1018	0.0775	0.1808	0.5891	0.5045	0.1184	0.1239
REPMIP2	< .0001*	< .0001*	0.3977	0.0286*	0.3931	0.9029	0.1704
REPMIP2PERC	< .0001*	< .0001*	0.3116	0.0030*	0.4988	0.4152	0.1612
QP	< .0001*	< .0001*	0.1083	0.0062*	0.6412	0.0186*	0.1918
QPPERC	< .0001*	< .0001*	0.0776	< .0001*	0.6630	0.0011*	0.9245
SA1	0.2174	0.2053	0.2680	0.3859	0.3945	0.2521	0.2500
SA2	< .0001*	< .0001*	0.0031*	< .0001*	0.2587	0.0006*	0.6735
SA3	< .0001*	< .0001*	0.0066*	0.0204*	0.4934	0.2370	0.7232
QP+SA	< .0001*	< .0001*	0.0008*	0.0043*	0.1545	0.0014*	0.4748

program. The fifth group consists of the remaining SA heuristics (SA2, SA3 and SA+QP, in which the end criterion is determined by a fixed number of SA iterations). Here also factor 3 (whether or not the blocks are equally divided over the surgeons) becomes significant. It turns out that SA can solve the problem faster when the blocks are not equally divided, which is not surprising since the number of possible exchanges is larger when all surgeons are equally represented and hence more evaluations need to be done per iteration. This also explains why this factor is not significant in SA3, for which the computationally expensive evaluation function is replaced with an easily computable one. The probability of a no show and (over)capacity do not play any role in the complexity of the problem, no matter which heuristic is applied.

4.6 Simulation study

Recall that in order to calculate expected shortages, the bed occupancy distributions are approached with normal distribution functions (see Section 4.4.1). Alternatively, the found schedules could have been evaluated using simulation. The reason why this was not done in the computational experiments described earlier is that (reliable) simulation takes too much computation time. However, to verify the accuracy of our results, a simulation experiment was done in which the predicted values (averages, variances and shortages) are compared with simulated values. In this part we summarize the findings of this experiment.
The experiment involved all 384 test instances. Each problem was again solved with the quadratic programming heuristic (QP). For each problem the total average and total variance of the bed occupancy (summed up over all 7 days) and total bed shortage resulting from the found schedule are calculated both through the theoretical results as outlined above and obtained through simulation:

1. Predicted values: the average and the variance are calculated using the theoretical formulas derived above. Expected shortages are calculated by approaching the bed occupancy distributions with normal distributions and applying numerical integration as described above.

2. Simulated values: the average, the variance and the shortages are calculated by simulating 1000 periods, taking into account a warm-up period in order to reach a steady state.

The experiment provided three series (averages, variances and expected shortages) of predicted and simulated data. These series were compared using a paired Student T-test (two-tailed). In the left part of Table 4.6 the results are given. The extremely small p-values for both the variance and the expected shortage indicate that these predicted values are different from the simulated ones. It turns out that the predicted variances are larger and hence also the predicted shortages are larger than the simulated ones.

The reason for this discrepancy is that the theoretical results do not take into account autocorrelation in the data. Indeed, a subset of the patients occupying a bed at period t will also occupy a bed at period $t+1$, namely those patients that stay longer than a cycle in the hospital. This means that the number of patients in the hospital at period $t+1$ can partly be explained by the number at period t. In other words, both numbers are dependent. When simulating more periods, the difference between numbers of occupied beds of subsequent periods differ less than expected from theoretical results, making the true variance smaller than the predicted one. In order to verify this explanation, the T-tests are repeated, but now only including those instances having the first setting of factor 6 (i.e. with LOS below the cycle time). The results are indicated in the right column of Table 4.6. As was expected, all p-values are now sufficiently high, indicating that the assumption of a (structural) difference between the predicted and the simulated data can be rejected.

Table 4.6: Predicted versus simulated data

	All 384 instances			Only 192 instances with LOS < cycle time		
	Pred.	Sim.	p-value	Pred.	Sim.	p-value
Avg. bed occupancy	967.85	967.82	0.4102	651.53	651.47	0.20
Avg. var. bed occupancy	170.02	151.24	< .0001*	130.02	129.95	0.76
Avg. total bed shortage	7.29	7.14	< .0001*	11.34	11.33	0.80

4.7 Conclusions

The purpose of this chapter is to propose and compare models and algorithms for building cyclic surgery schedules. Compared to existing approaches this is the first work in which concrete models are presented that aim at leveling the resulting bed occupancy and enable to predict performance measures as the daily expected bed occupancy, the variance on this occupancy, the expected bed shortage and the probability of a shortage on each day. The models take into account stochastic numbers of patients per operating room block and a stochastic length of stay for each operated patient.

One can distinguish between two approaches: a MIP based approach and a metaheuristic approach. In the first approach the non-linear objective function is being replaced with a linear (or quadratic) one and the resulting models are solved with a state-of-the art MIP solver. Models have been proposed that aim at the minimization of the highest expected bed occupancy peak, highest bed occupancy variance or a combination of both. Additionally, a number of repetitive MIP solving algorithms have been developed. The second approach preserves the original objective function and searches a good solution by means of a metaheuristic (simulated annealing) approach. All algorithms have been extensively tested and their results compared. The best solutions are found with the simulated annealing approach. However, this approach also takes the longest computation times. Concerning the MIP based approaches, the best results are obtained with the quadratic programming (QP) models in terms of both solution quality and computation time. A hybrid approach in which a simulated annealing search is initiated with a schedule found by a quadratic program yields good results with regard to both solution quality and computation time.

The overall conclusion is that the best results are obtained by a metaheuristic approach in which the true objective is evaluated. However, MIP approaches involving linearized and/or quadratic objective functions entail important advantages compared to metaheuristic approaches. First of all, these approaches manage to find good solutions within small computational effort. Second, a MIP model can easily be tuned to meet specific requirements. Incorporating an extra real-life restriction can often be done by simply adding a constraint to the MIP, whereas metaheuristics usually require some extra coding in order to cope with the modified problem.

Moreover, the different weights can easily be adapted so that the model delivers solutions that are appropriate in specific cases. For instance, when the peak variances are the main problem, the (quadratic) MIP with the weight for the variance equal to 1 may be appropriate. Finally, the single and quadratic MIP approaches allow the scheduler to find answers to questions as "What is the lowest maximal peak in the mean (variance) bed occupancy that can possibly be achieved?" or "What is the most leveled mean bed occupancy possible?". The answers to these questions can provide decision-makers (schedulers) with important insights into the characteristics of the system.

The developed models are very basic. Only two types of constraints have been considered: surgery demand and operating room capacity constraints. For real-life applications a number of additional constraints are required such as workforce capacity constraints (anaesthetists, nursing staff), surgeons preference constraints (e.g., all blocks at maximal two different days), material requirement constraints, transition constraints (change of equipment from one surgery group to another) etc. It would be interesting to implement these models for a real-life case to see to what extent they can improve existing practices. This is the first and most important item for future research. From a theoretical point of view, it would be interesting to see which extensions could easily be handled by which solution approach and which not. Furthermore, the impact of these extensions on both solution quality and computation time could be researched.

Chapter 5

Building cyclic master surgery schedules with leveled resulting bed occupancy: A case study

This chapter describes a real-life application of the algorithms for building cyclic master surgery schedules with leveled resulting bed occupancy proposed in Chapter 4. The study starts from detailed information on all elective surgery interventions during a 1-year period in a medium-sized Belgian hospital. For each surgeon-hospitalization unit combination multinomial distribution functions are derived for both the number of operated patients per operating room block and the length of stay of each operated patient. These distribution functions serve as the input for the algorithms. Leveling is achieved by either mixed integer programming techniques involving the solution of a min-max optimization problem and a quadratic optimization problem, or a simulated annealing heuristic that minimizes the total probability of bed shortage or, alternatively, the total expected bed shortage.

5.1 Introduction

The purpose of this chapter is to present a real-life application of the theoretical models proposed in Chapter 4. Recall that these models enable us to build a cyclic master surgery schedule for which the resulting bed occupancy is leveled as much as possible and for which performance measures as the daily expected bed occupancy, the variance on this occupancy, the expected bed shortage and the probability of a shortage on each day can be predicted. Multinomial distribution functions are assumed for both the number of patients per operating room block and the length of stay (LOS) of each operated patient. The models applied in this case study are slightly extended implementations of the theoretical ones. The most important extension includes that more than one hospitalization unit is considered, leading to probability distributions for each surgeon-hospitalization unit combination. Also, block sizes may vary or, in other words, room allocations to surgeons can have variable durations. This extension has some consequences for the simulated annealing approach for which we have added a corresponding neighborhood move (see further). Finally, a real-life constraint is added that prevents individual surgeons from being scheduled in different rooms at the same time.

The rest of this chapter is structured as follows. Section 5.2 gives an outline of the theoretical background of the models applied in this study. Section 5.3 gives some more information on the hospital that has provided the data for this case study. Section 5.4 contains a discussion of the input analysis. More specifically, it is explained how the multinomial probability distributions are fitted for both the number of operated patients and the LOS of each operated patient for each surgeon-hospitalization unit combination. Section 5.5 contains a presentation of the graphical user interface that was built on top of the algorithms to visualize the operation and performance of the system. Section 5.6 discusses the results obtained by applying the different approaches while Section 5.7 draws conclusions and lists some topics for future research.

5.2 Theoretical background

The algorithms that are applied to build the master surgery schedule can be divided into two classes. The first class consists of mixed integer programming approaches. A distinction is made between linear min-max optimization approaches

and a quadratic optimization approach. The second class consists of a simulated annealing heuristic that minimizes the total probability of bed shortage or, alternatively, the total expected bed shortage. The general principle behind both the MIP approaches and the metaheuristic approach is the same. Using the information on the stochastic distribution functions of the number of operated patients per block as well as the length of stay of each operated patient, both the mean and the variance of the daily bed occupancies by the elective cases can be calculated exactly for each hospitalization unit. To this purpose, the contribution of each surgeon-block allocation to the mean and variance of the daily bed occupancy of each hospitalization unit must be known. The respective formulas to calculate these contributions have been derived in Chapter 4.

The way in which the mean and variance of the daily bed occupancies are used to build a good cyclic master surgery schedule differs between the different approaches. In the linear MIP approaches, the maximum of the daily mean (variance of the) bed occupancies is minimized. In the quadratic MIP approach, the daily mean (variance of the) occupancies are explicitly leveled by minimizing the quadratic sum. The models are solved by a state-of-the-art mixed integer programming optimizer.

For the simulated annealing approach, shortage probabilities are calculated by assuming normally distributed bed occupancies, making use of the central limit theorem. Additionally, by applying numerical integration techniques, expected shortages can be calculated. The objective function then involves the minimization of either the total shortage probability or the total expected shortage. To achieve this objective the algorithm iteratively explores neighbor solutions. A neighborhood move either involves an exchange of full block allocations (this move may include several surgeons) or involves an exchange between individual surgeon allocations to blocks. If an exchange leads to a better solution, the change is accepted. Otherwise, the change is rejected with an increasing probability towards the end of the search process.

5.3 Case study

The case study presented entails the Virga Jesse Hospital, situated in Hasselt, Belgium. The 2004 annual report of this medium-sized hospital shows an important

increase in activities. In 2004 the number of inpatient admissions has grown to a historical record of 21,923. In the same year the total revenues increased with 13 million Euro up to 163 million Euro. Also the number of outpatient admissions (referred to as day hospitalizations) has risen, while the average length of stay per patient has decreased. The election of the Belgian HR manager of the year and the laureate of the prestigious Belgian Tyco health care price for excellence in health care management are the evidences of Virga Jesse's top-class service.

Virga Jesse's central operating room complex consists of 9 rooms in which a total of 46 surgeons have been assigned operating room time. These surgeons are classified into 15 different surgical groups with respect to the specialism. Each operating room is open from Monday to Friday for 8.5 hours. Up to now, no elective surgery takes place during the weekends. The operated patients recover in one of the 25 hospitalization units of which only 10 units have served more than 100 elective cases in 2004. The models applied in this study involve the development of a (cyclic) master surgery schedule with leveled bed occupancy in these 10 major hospitalization units.

5.4 Input analysis

Both the MIP based approaches and the simulated annealing approach require as input for each surgeon-hospitalization unit combination the probability distributions of the number of patients per block and the LOS for each operated patient. The theoretical models assume multinomial distributions, often referred to as empirical discrete probability distributions. These general probability distributions can easily be constructed from a database containing the detailed information on all surgical interventions that have been performed in a reasonably long time period (e.g., one year). Table 5.1 contains a snapshot of the (relevant) fields of the input file.

A procedure has been written that reads in these data records provided as an ASCII text file and automatically constructs the probability distributions for both the number of patients per block and the LOS per patient for each surgeon-hospitalization unit combination. This procedure simply counts the number of cases on each day for each surgeon-hospitalization unit combination. When these

Table 5.1: Snapshot of the input file containing detailed information on all surgical interventions in 2004

OR_NR	SURGEON	ROOM	HOSP. UNIT	DATE_IN	DATE_OUT
23005838	PUTE	Operatiezaal 04	3200	2/01/2004 8:00	2/01/2004 17:00
23116828	DTRG	Operatiezaal 09	3200	2/01/2004 8:00	2/01/2004 17:00
23408780	VDVG	Operatiezaal 03	2150	2/01/2004 8:00	5/01/2004 15:00
23409553	BOES	Operatiezaal 05	2160	2/01/2004 8:00	5/01/2004 15:19
23382108	PUTE	Operatiezaal 04	3200	2/01/2004 8:05	2/01/2004 17:00
23383582	LENH	Operatiezaal 08	3200	2/01/2004 8:05	2/01/2004 17:00
23409151	PUTE	Operatiezaal 04	3200	2/01/2004 8:10	2/01/2004 17:00
23408550	PUTE	Operatiezaal 04	3200	2/01/2004 8:15	2/01/2004 17:00
23382105	PUTE	Operatiezaal 04	3200	2/01/2004 8:20	2/01/2004 17:00
23408576	VDKJ	Operatiezaal 06	3200	2/01/2004 8:20	2/01/2004 17:00
...					

numbers are divided by the total number of respective surgery days, the probabilities for the number of operated patients per block that recover in the corresponding hospitalization unit are obtained. Since totals are made per day, it is implicitly assumed that a surgeon cannot be assigned to more than one block per day. This assumption holds in our case study as well as for many other hospitals. The same reasoning is applied for constructing the LOS distributions, but now obviously no intermediate day totals have to be made.

Only elective (planned) interventions are taken into account. The reason why the non-elective (emergency) cases are not retained is twofold. First, the occurrence as well as the recovery period of non-elective, emergency cases is, by definition, highly unpredictable and hence it would make little or no sense to fit a probability distribution to them. Second, non-elective cases often take place in blocks not preserved for the surgeon performing the surgery. Taking them into account would lead to a biased distribution for the number of patients per operating room block.

Table 5.2 shows an example of the output of this procedure, i.e., the derived probability distributions, for one particular surgeon. It must be clear at this point that the LOS distributions are specific for each surgeon-hospitalization unit combination. This is a very realistic basic assumption since the patient recovery time is

usually strongly related to this unique combination as patients operated by the same surgeon and recovering in the same hospitalization unit often suffer from similar ailments. Of course, surgeons can perform different surgical treatments in one block, but the proportions of these treatments are often reasonably constant.

Before applying this procedure, the surgeons and the existing schedule have to be read in manually. The existing schedule is needed to determine wether the case is elective or non-elective. If the intervention takes place on a day during which a block is preserved for the surgeon, it is considered to be an elective case. Otherwise, it is considered to be a non-elective case. A problem arises when a surgeon is assigned to more blocks having different durations. In this case, a 'dummy' surgeon is introduced for each different block duration. For instance, consider a surgeon who has been assigned one block of 8.5 hours on Monday and one block of 4 hours on Tuesday. In our approach, distributions will be derived for the Monday block as well as for the Tuesday block by introducing a 'dummy' surgeon for the latter. This implies that block durations are considered to be fixed when searching for better schedules. It also implies that hours cannot be exchanged between blocks. Only shifting of total blocks will be allowed.

The choice for this approach is justified as follows. First of all, a 'block' is probably the best unit for deriving the probability distributions. A smaller unit (e.g., an hour) is in our view less effective to fit the real distributions. Second, a block that extends twice as long as another block, assigned to the same surgeon, does not necessarily include twice the number of patients. Hence, not introducing a dummy surgeon would lead to derived probability distributions basically representing a mixture of two or more distributions. Third, working with fixed block durations entails some interesting computational features. It enables us to *a priori* calculate the per surgeon bed occupancy contributions. These contributions are needed as input for the mathematical programming models. With variable block sizes on the other hand, one could only calculate these contributions when the number of hours assigned per block is known. This would dramatically complicate the problem. Fourth, the graphical user interface is kept extremely simple as block assignments and exchanges can easily be done by dragging and dropping. Finally, from a practical point of view, most of the surgeons have no different block durations and hence relatively few dummy surgeons have to be introduced.

Table 5.2: Example of nr. patient and LOS distributions for three hospitalization units for surgeon DUPA

SURGEON	HOSP. UNIT	NR. PATIENTS		LOS	
		NR. PAT.	PROB.	NR. DAYS	PROB.
DUPA	2160	0	0.20	3	0.20
		1	0.38	4	0.02
		2	0.34	5	0.02
		3	0.06	6	0.03
		4	0.02	7	0.28
				8	0.21
				9	0.21
				10	0.03
				12	0.02
	2601	0	0.56	4	0.03
		1	0.34	7	0.04
		2	0.06	8	0.41
		3	0.04	9	0.45
				10	0.07
	3200	0	0.16	1	1
		1	0.10		
		2	0.22		
		3	0.30		
		4	0.12		
		5	0.08		
		6	0.02		

5.5 Graphical user interface

In this section the graphical user interface (GUI) is presented. The GUI visualizes the surgery schedule and the resulting bed usage occupancy distributions for a given schedule. Moreover, it allows the user to modify an existing schedule and to view the impact of a change in the schedule on the bed occupancy. Data like the schedule properties, the surgeon properties and the hospitalization properties can easily be read in and modified. Automation features include the deduction of the probability distributions for patient numbers and lengths of stay from a database (as described in Section 5.4) and the optimization of the schedule with respect to certain objective measures. Figure 5.1 shows an overview of the GUI with an empty surgery schedule.

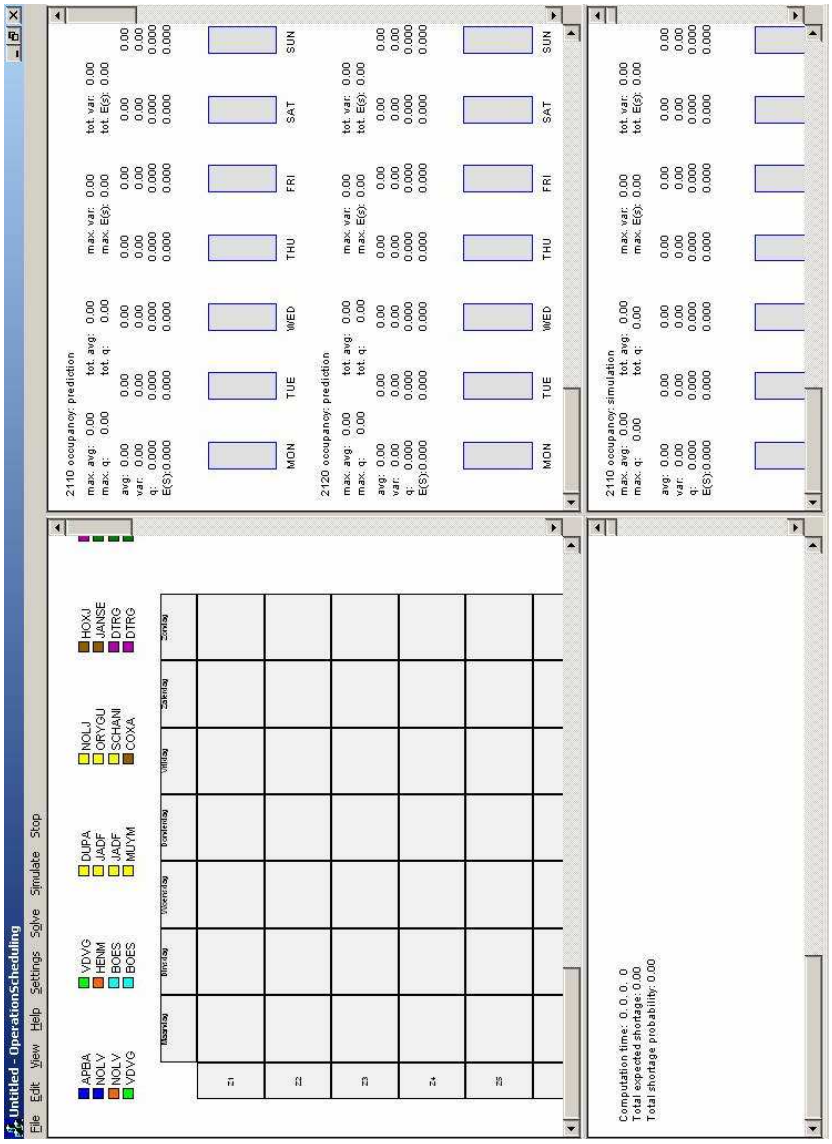

Figure 5.1: Overview of the GUI with empty schedule

The main window is divided into four views. In the upper left pane the (empty) master surgery schedule is shown. The seven columns in the grid represent the seven days of the week. The nine rows represent the nine operating rooms. Each room is open on each weekday for 8.5 hours. A subset of the surgeons is shown above the grid. The schedule could now be built easily from scratch by dragging and dropping the surgeons to the timetable cells. Of course, a room can also be assigned for a limited number of hours instead of the full 8.5 hours. Each assignment introduces a patient flow in the system, which is reflected by an increase in the bed occupancy of one or more hospitalization units on one or more days. This is represented in the upper right pane. Schedules could also be built automatically while aiming at certain optimization objectives. The computational results (like solution time, solution quality, etc.) are given in the lower left pane. Finally, the right bottom pane is a simulation pane. A simulation run could be done in order to validate the theoretical basic assumptions (mainly the central limit theorem) of the model. To this purpose it can be verified whether the predicted bed occupancies (and shortages) obtained by calculation are similar to the ones obtained by simulation. Figure 5.2 shows the current master surgery schedule with resulting bed occupancy (only three hospitalization units are shown). The small T-ending bars on top of each colored occupancy box indicate the standard deviations of the bed occupancy distributions on the corresponding days at the corresponding hospitalization units.

Using dialog boxes, the schedule, surgeon and hospitalization unit properties could easily be modified. As an example some of the dialog boxes for editing the surgeon properties are represented in Figure 5.3. The left dialog box shows the surgeon basic properties and a list of the hospitalization units to which patients of the selected surgeons flow. The user can select one of these units to edit. The upper right dialog box then allows the user to choose between the number of patients distribution or the LOS distribution for editing. The lower right dialog box finally allows the user to edit individual distribution values (number and probability) of the LOS distribution for this particular surgeon-hospitalization unit combination.

Concerning the automation procedures, one basically can choose between two approaches: a mixed integer programming procedure, either a linear or a quadratic one, that aims at leveling the bed occupancy of one or more hospitalization units, or a simulated annealing approach that directly tries to minimize the total shortage

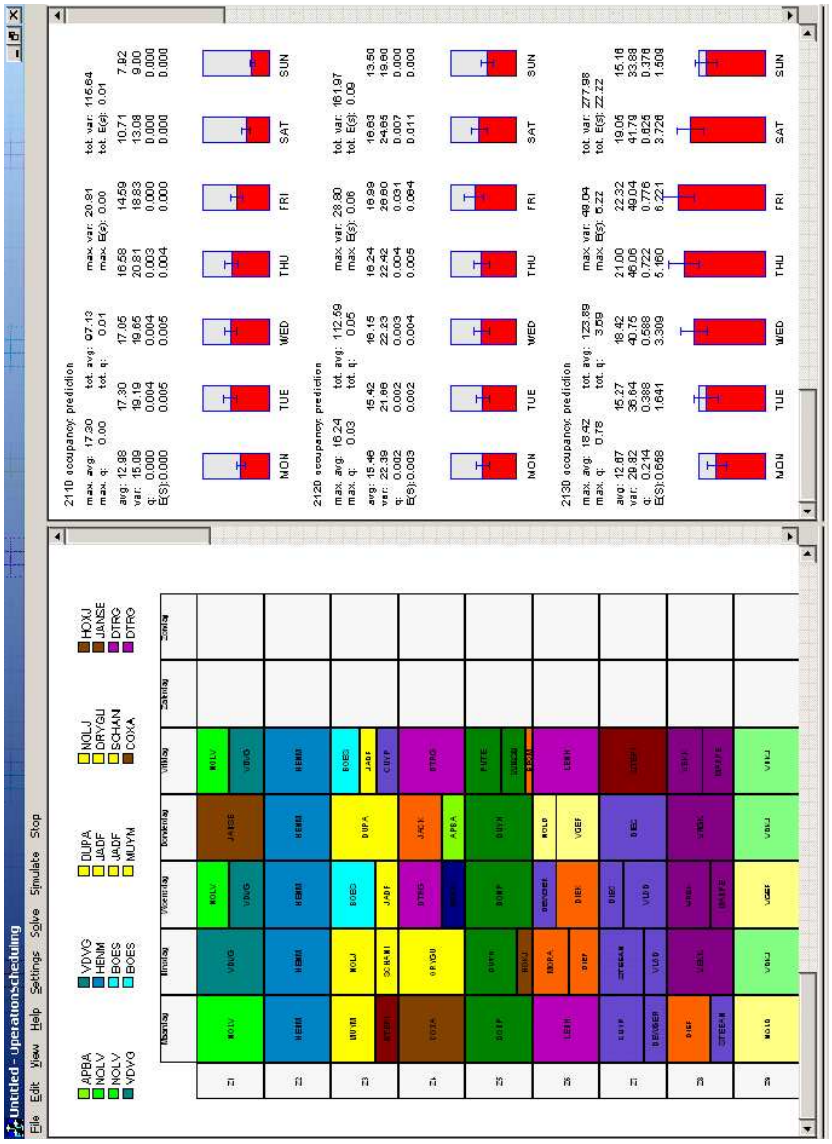

Figure 5.2: Current master surgery schedule with resulting bed occupancy (only three hospitalization units shown)

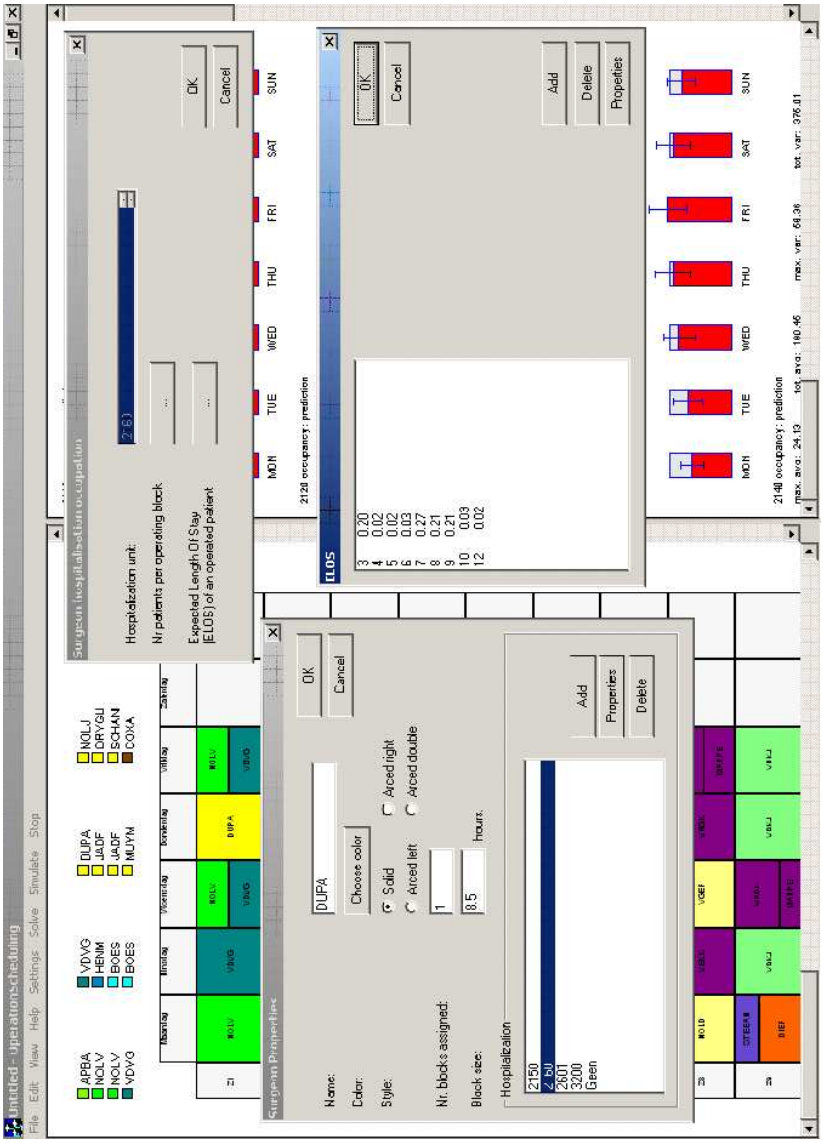

Figure 5.3: Editing the properties of a surgeon

probability or the total expected shortage. To provide additional details for the integer programming procedure, the dialog box shown in Figure 5.4 is displayed. The user can choose between the linear or the quadratic variant, specify the maximum running time limit and provide the objective function weights of the respective hospitalization units (2110, 2120, 2130,...) for the mean (first column) as well as the variance (second column). The weights represent the relative importance of the leveled bed occupancy for the respective hospitalization units; e.g., a weight of 1 (0) indicates that a particular hospitalization unit is (not) taken into account.

When the maximal time limit is reached, the dialog box displayed in Figure 5.5 pops up. The user obtains information about the optimality status and can choose either to stop the algorithm or to continue the search for an additional time span. The given information includes the current best found objective value, the lower bound, the gap, that is the difference between the current solution and the lower bound expressed as a percentage of the second, and the number of explored and non-explored nodes in the search tree.

To start a simulated annealing (SA) procedure, the user has to specify a number of general SA settings (initial temperature, temperature update interval and update factor), give the stop criteria (max. nr. iterations, max. time limit), indicate the

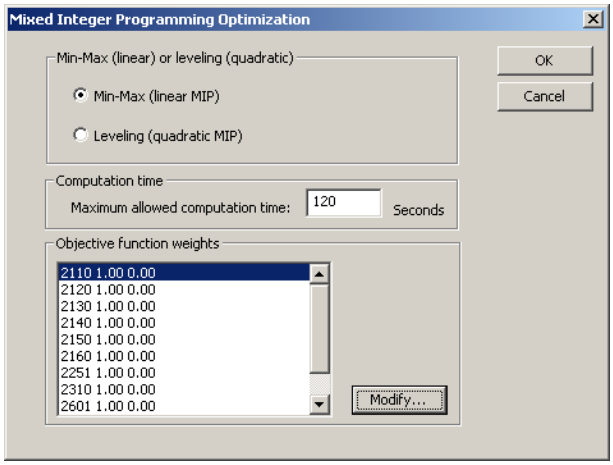

Figure 5.4: Dialog box: Starting a MIP

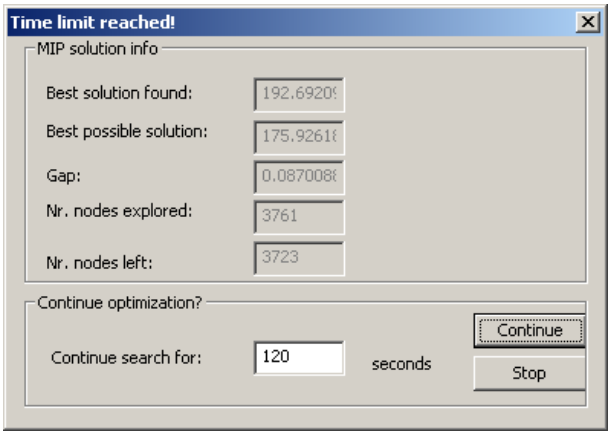

Figure 5.5: Dialog box: Upon completion of the MIP

probability of a whole block move (which automatically determines the probability of a one surgeon move) and choose between the two objective functions. The dialog box to provide this information is shown in Figure 5.6.

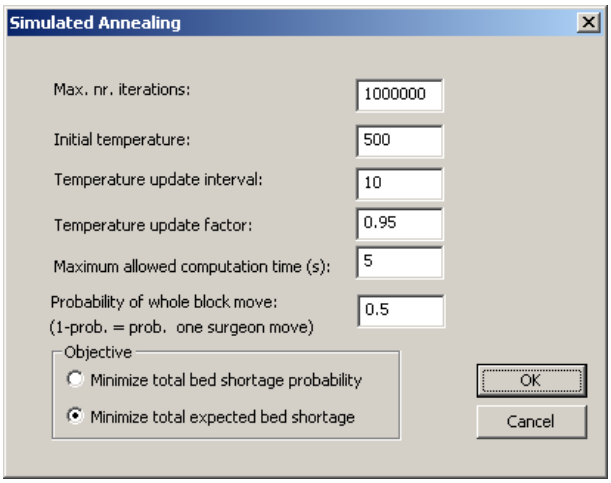

Figure 5.6: Dialog box: Starting a simulated annealing procedure

5.6 Results

Rather than trying to find the overall best master surgery schedule for the Virga Jesse Hospital, which is a subjective matter after all and hence makes little sense anyway, we discuss and compare the results of a number of different algorithm runs.

As can be seen in Figure 5.2, some problems may arise at hospitalization unit 2130 (third unit, shown at the bottom), where there is a high peak occupancy on Friday leading to a positive expected bed shortage. An optimization procedure that exclusively focuses on this hospitalization unit could turn out to be useful to solve the problem. The resulting schedule of a linear MIP, aiming at the minimization of the maximum mean occupancy peak of unit 2130 is shown in Figure 5.7. It should be clear that the bed occupancy in unit 2130 is now much more leveled over the week. Also hospitalization unit 2140 suffers from large differences in the bed occupancy peaks. This asks for a scheduling procedure that simultaneously focusses on the leveling of the bed occupancy distributions in units 2130 and 2140. To this purpose, a linear MIP procedure that minimizes the weighted maximum peak of the bed occupancies in units 2130 and 2140 could be applied. However, as we already presented a solution based on a linear MIP procedure (see Figure 5.7), we present the results of a quadratic MIP with weights 1 for units 2130 and 2140 and 0 for all other units. Figure 5.8 contains the resulting schedule.

Finally, Figure 5.9 displays the schedule that results from applying a simulated annealing procedure with the settings shown in Figure 5.6. This procedure tries to minimize the total expected bed shortage, taking all hospitalization units into consideration.

It is difficult to objectively compare the quality of the generated schedules, as there is no once and for all objective measure to make this comparison. To build a quality schedule or at least to improve the current schedule, one has to study the current practices and determine the most appropriate objective function and automation procedure. For instance, if capacity problems always occur at the same hospitalization unit, a linear or quadratic MIP procedure that focusses on this unit will probably render the best results. The visualization of the bed occupancies can of course assist in determining the appropriate model. However, there might be a different explanation for the variability in these occupancies rather than the vari-

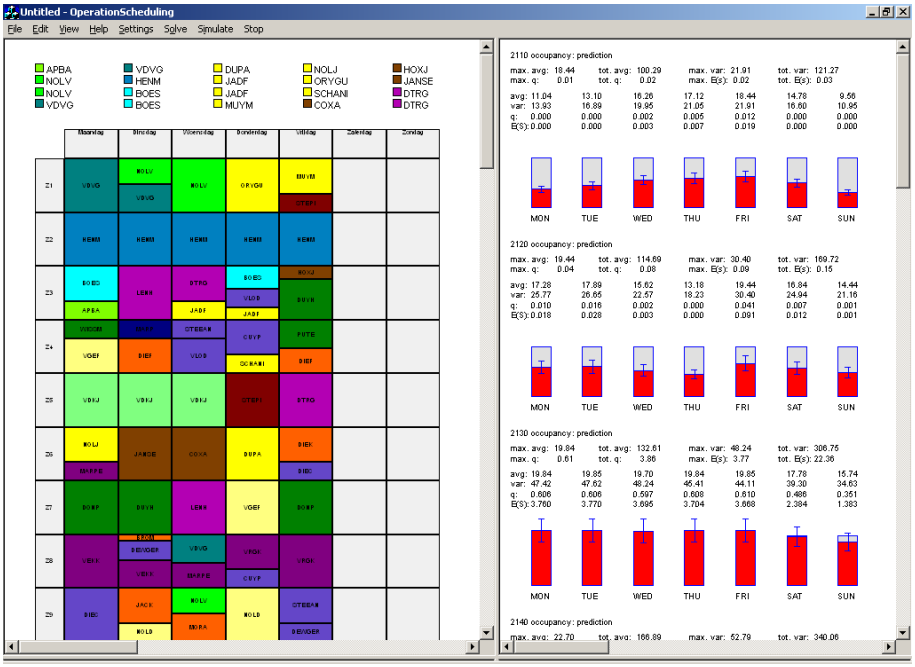

Figure 5.7: The results of a linear MIP to level the mean bed occupancy of hospitalization unit 2130 (shown in the lower right)

ability in surgery admissions and LOS.

Suppose we take the expected number of bed shortages over all ten hospitalization units included in this study as the one and only objective measure. The results of several optimization procedures are shown in Table 5.3. The first line indicates the total expected bed shortage in the current schedule. This number (37.82) indicates that, over all hospitalization units, more than 5 beds per day are lacking in the assigned hospitalization unit and hence have to be found in another hospitalization unit. A possible explanation for this remarkably high number is as follows. Recall that we derived the per surgeon probability distributions from the daily records of a database containing all surgical interventions during a 1-year period. The resulting variability in these derived distributions is, however, probably higher than the real-life variability. Indeed, in real life it is probably the case that a surgeon ad-

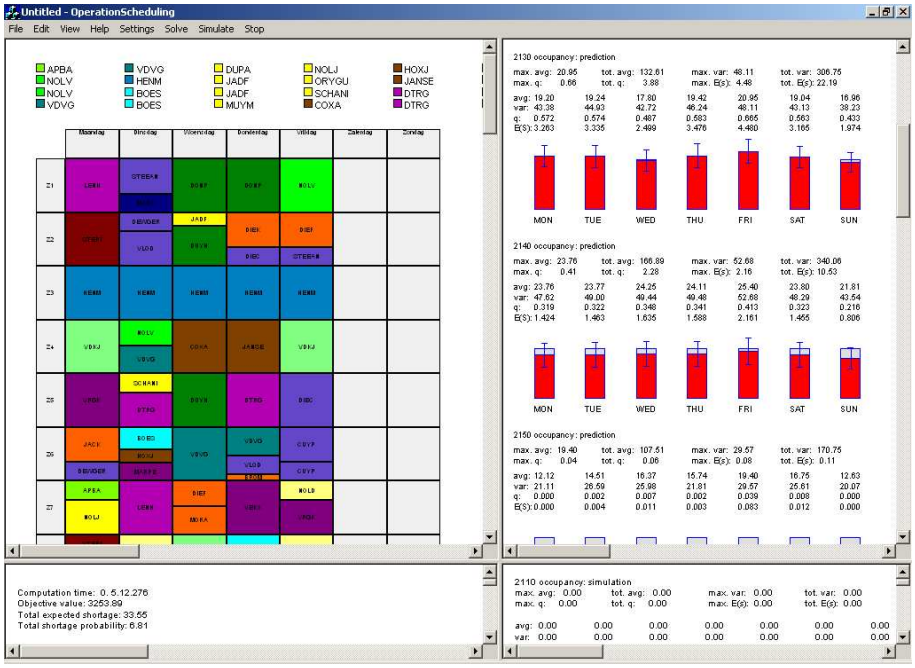

Figure 5.8: The results of a quadratic MIP to level the mean bed occupancy of hospitalization units 2130 and 2140

mits or rejects patients as a function of the remaining bed capacity at the relevant hospitalization unit at that moment. In other words, an important part of the variability can be taken care of by appropriate admission of elective cases during the third stage of the surgery scheduling process, which involves the detailed planning of the individual elective cases in the allocated blocks. Obviously, in the concern of both patient and surgeon the postponement of surgery is best avoided as much as possible. Therefore, methods for a careful design of the master surgery schedule, as presented in this study, are still valuable.

Table 5.3 shows that the total expected bed shortage drops from 37.82 to 34.44 if a linear MIP approach is used in which the maximal mean bed occupancy of the bottleneck hospitalization unit 2130 is used. The shortage decreases further to 34.12 if also the variance of the bed occupancy in this unit is taken into consideration. If

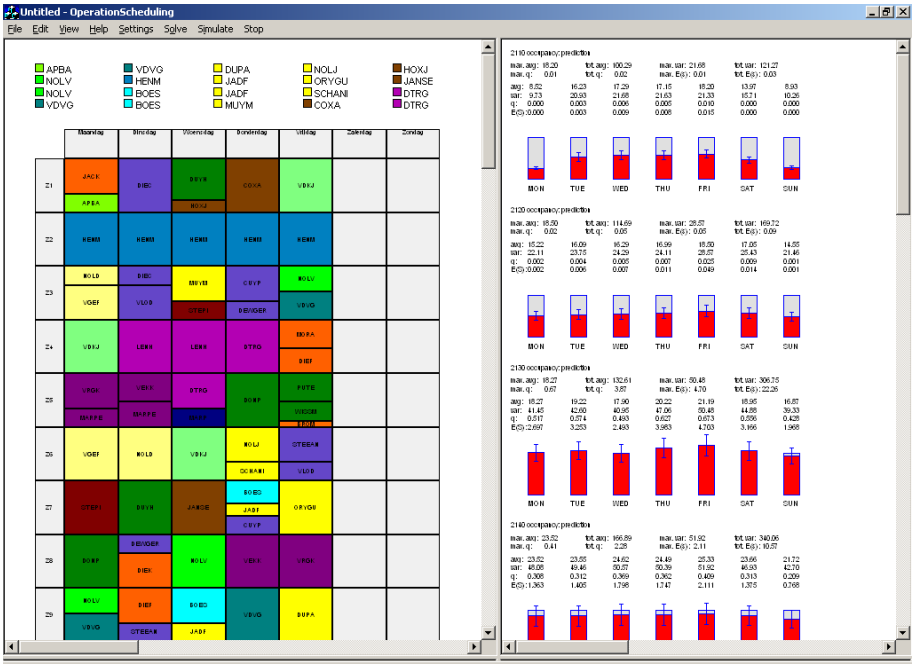

Figure 5.9: The results of a simulated annealing algorithm that minimizes the total expected bed shortage

Table 5.3: Minimizing the total expected bed shortage

Procedure	Total exp. shortage	Comp. time (s)
Current schedule	37.82	-
Lin. MIP MIN-MAX mean 2130	34.44	120
Lin. MIP MIN-MAX mean + var. 2130	34.12	120
Quad. MIP mean 2130, 2140	33.55	120
Quad. MIP mean all units	33.29	360
SA	33.12	120

a quadratic MIP procedure is applied on the two most heavily loaded units 2130 and 2140, the expected shortage decreases to 33.55 and if all units are taken into account to 33.29. Finally, application of a simulated annealing procedure that directly aims at this objective results in a schedule with expected bed shortage equal

to 33.12.

It must be clear that this result does not necessarily mean that the schedule displayed in Figure 5.9 is the best schedule for Virga Jesse. It is just a possible schedule that has the best score (at least amongst the schedules resulting from the few procedures we have tested) on one measure. The real power of the software lies in the visualization of the schedule and the resulting bed occupancy, the ease with which schedules can be built and the capability it provides to carry out an in-depth analysis of the existing system. Using the software, managers can find answers to questions like "what is the most leveled bed occupancy possible at hospitalization unit X?" or "which schedule simultaneously levels the bed occupancy in units X and Y?".

5.7 Conclusions and future research

This chapter has illustrated how the models developed in Chapter 4 could be applied to a real-life case. To this purpose, the required input data, namely the distribution functions for the number of operated patients as well as for the length of stays, have been derived from the central database containing detailed information on all surgical cases during a 1-year period in a medium-sized Belgian hospital. The graphical user interface hides the algorithmic procedures and makes it easy to build a schedule having particular features like the most leveled bed occupancy in a certain hospitalization unit or the smallest overall bed shortage probability. Depending on the hospital's situation, and in particular on the problems it is facing, a procedure can be chosen to build a new master surgery schedule. Additionally, the application can provide managers with important insights into the behavior of the system.

The models make abstraction of the differences between the operating rooms, i.e., for the resulting bed occupancy it is completely irrelevant whether a surgeon is allocated to room 1 or to room 2. As a consequence, the models often result in a schedule in which surgeons from different disciplines have to share one operating room, sometimes even on the same day. However, in many cases surgeons of the same group prefer to be scheduled in the same room for this includes several practical benefits. Of course, shifting surgeons to other rooms within the same day

has no impact upon the bed occupancies. Hence, the scheduler can easily swap room allocations within the same day in order to group surgeons having a similar discipline into the same room as much as possible. Instead of doing this manually, a post-improvement heuristic could be written to do the job. Alternatively, room restrictions could be an integral part of the optimization model, either in the objective function or as additional constraints. This is an interesting direction for further research.

A second shortcoming of the proposed models is the assumption that the generated schedule is completely repeated each cycle time. It might be more efficient for those surgeons having assigned a small number of hours of operating time per cycle to aggregate these hours and only operate once in two or three cycles. Obviously, a prolongation of the cycle time would deal with this issue. However, surgeons also like to have their schedule as simple as possible which entails as few changes as possible from week to week. The latter is not guaranteed if we simply apply the models described above with a longer cycle time. Blake et al. (2002) overcome this problem as follows. In a first phase, they relax the surgery demand constraints in their integer programming model, taking as an objective for their cycle master surgery schedule the minimization of the undersupply of target operating room hours (see also Blake and Donald, 2002). In a second phase, a post-improvement heuristic is run that tries to further improve this objective by introducing some changes in the schedule from week to week. A similar heuristic could be written to accompany our cyclic surgery scheduling models. Alternatively, we could work the other way around; i.e., constructing a schedule with a longer cycle time (e.g., two or three weeks) and afterwards running an improvement algorithm (either heuristic or exact) that tries to minimize the changes in this schedule from week to week with no (or few) impact upon our objective function.

Chapter 6

Integrating nurse and surgery scheduling

A common problem at hospitals is the extreme variation in daily (even hourly) workload pressure for nurses. The operating room is considered to be the main engine and hence the main generator of variance in the hospital. It is our belief that integrating the operating room scheduling process with the nurse scheduling process is a simple, yet effective way to achieve considerable savings in staffing costs. The purpose of this chapter is threefold. First of all, we present a concrete model that integrates both the nurse and the operating room scheduling process. Second, we show how the column generation technique approach, one of the most employed exact methods for solving nurse scheduling problems, can easily cope with this model extension. Third, by means of a large number of computational experiments we provide an idea of the cost saving opportunities and required solution times.

6.1 Introduction

As already has been demonstrated in Section 1.1 of Chapter 1, cost pressures on hospitals have increased dramatically during the last decades. This emphasis on cost containment has forced hospital executives to run their organizations in a more business-like manner. The constant challenge is to provide high-quality service at ever reduced costs. In order to achieve this purpose inefficient use of resources

should be identified and actions should be taken to eliminate these sources of waste. Operations research techniques are increasingly being used to assist in this complicated task.

As nursing services account for an important part of a hospital's annual operating budget, concentrating on this resource can lead to substantial savings. The situation is exacerbated by an acute shortage of nurses in all western countries, said to be 120,000 today and expected to grow to 808,000 by 2020 in the United States (US) alone (USDHHS, 2002). Hence, it is of vital importance that nurses are used as much as possible at the right time and at the right place. This goal is hard to achieve because of two reasons. The first one is inherent in service organizations for which human resources outnumber all other types of resources. Unlike machines, staff schedules are restricted by collective agreement requirements. These form an important hindrance for the flexibility with which nurses are scheduled.

A second reason is the presence of variability. Variability is probably the main obstacle to efficient delivery of health care and reducing it is one of the major concerns in current health care management (Litvak and Long, 2000). Compared with many industrial production environments, hospitals are much more stochastic by nature. Indeed, human bodies are undoubtedly more complex than any other artificially made product. Consequently, the arrivals of patients, the occurrence of complications and the patients' recovery times in hospitals are usually more uncertain than the demand for products, the occurrence of machine break-downs or product failures and the repairing times in production systems. One common problem at hospitals is the extreme variation in daily (even hourly) workload pressure for nurses. On days when the workload is too high, the quality of care decreases because it is too costly to staff for peak loads. On days when the workload is too low, there is waste. Fortunately, the situation is not as chaotic as it seems to be at first sight. As pointed out by Litvak and Long (2000), an important amount of the variability can effectively be managed and reduced by a thorough analysis of the existing system and by appropriate decision taking. Special emphasis is put on the operating room since it is considered to be the main engine and hence the main generator of variance in the hospital. It is our belief that integrating the operating room schedule process into the nurse scheduling process is a simple yet effective way to achieve considerable savings in staffing costs.

Nurse scheduling problems are frequently encountered in the operations research literature. To the best of our knowledge, all the proposed models consider the nurse scheduling problem as a separate problem, i.e., not related to any other activity in the hospital. In this chapter we will describe a more general approach in which the demand constraints are dependent on the operating room schedule and hence become a part of the decision process.

The operations research literature is replete with examples of integer programming techniques being applied to operating room scheduling problems. As already mentioned, these studies can be categorized based on the stage of the scheduling process to which it applies. The surgery scheduling part in this chapter is again situated in the second stage, that entails the development of a master surgery schedule.

The methodology presented in this chapter has some similarities with models for integrating the scheduling of project tasks and employees (Alfares and Bailey, 1997; Alfares et al., 1999). Although several authors mention the interdependency between the surgery scheduling process and the development of nurse rosters, as far as we know, no models have been proposed to integrate both areas of decision making. Litvak and Long (2000) underline the negative impact of variability in hospital environments. They consider the operating room as the engine that drives the hospital. Consequently, the activities inside the operating room heavily determine the fluctuations in resource demands throughout the rest of the hospital. A poor operating room schedule could for instance be directly responsible for the occurrence of (contra-productive) peaks in the demand for certain types of resources. Chapter 3 has introduced this idea by presenting a software package to visualize the use of various resources as a function of the operating room schedule. In the subsequent chapters we have focussed on the bed occupancy. Chapter 4 proposed a number of integer programming models for building robust surgery schedules for which the resulting expected bed shortage is minimized, while Chapter 5 described a real-life application of these models. Beds are, however, not the only important resource in hospitals. On the contrary, beds can only be called an important resource, in the sense of expensive and thus limited, as far as it concerns staffed beds. A leveled bed occupancy leads in many cases to a leveled workload pattern which on its turn usually leads to a favorable workload pattern for nurses. However, since the scheduling of nurses by itself already entails a lot of constraints, one cannot really judge the quality of a surgery schedule with respect to the resulting workload distribution,

without explicitly taking these constraints into account. Hence, an integrative approach of both scheduling fields is required in order to build a high-quality surgery schedule that aims at a reduction in the staffing costs.

In this chapter the master surgery schedule is being considered as the main generator of the workload of the nurses. In order to couple both scheduling environments, the objective in the surgery schedule process will be to construct a favorable workload distribution for the nurses.

This chapter is organized as follows. In Section 6.2 a general overview of the model together with a branch-and-price solution approach is presented. Section 6.3 provides more details on both pricing problems, while a general overview of the branch-and-price algorithm is given in Section 6.4. Section 6.5 discusses a specific branching scheme. In Section 6.6 some computational issues are discussed and in Section 6.7 extensive computational results are given. Finally, Section 6.8 draws conclusions and lists some topics for further research.

6.2 Model description

6.2.1 Visualization of the idea

Consider a hospital department confronted with the nurse scheduling problem displayed in Figure 6.1. In this figure, the nurse scheduling problem is visualized in a table in which the rows represent the time horizon. Assume that we have to build a schedule for four weeks (28 days) in which each day consists of three shifts (morning, day and night shift). For each shift the number of nurses required to do the work is known. These demand values are indicated in the utmost right column in Figure 6.1. The other columns represent the roster lines by which an individual nurse can be scheduled. An X indicates that the nurse is scheduled to work during the corresponding shift. Of course, not every permutation of X's represents a feasible roster line, as several constraints apply on the individual roster lines. For instance, a night shift cannot be followed by a morning shift on the next day, because the nurses need sufficient time to rest. A second example includes a maximum limit on the number of working shifts for a given time period, e.g., a nurse cannot work more than 20 days out of 28. The problem from which we start our reasoning can be formulated as follows: given a time horizon, a particular workload for each shift

in this time horizon and a particular set of constraints on individual roster lines, what is the minimum number of nurses needed to cover all the work and how should we schedule these nurses?

Suppose that we found, either manually or with the help of a software package, an optimal solution for this problem. Hence, we know the minimum number of nurses, say N, needed to cover the demand and also how to schedule these N individual nurses. The question that challenges us is whether it would be possible to run the same system with less nurses. Obviously, if the problem remains unchanged, it will be impossible to do the same amount of work with less nurses, as N, being the optimal solution, is the minimum number of nurses needed. In order to cut down the number of nurses, we need to change the problem, but without changing the actual system. One possibility is to relax the individual roster line restrictions. As this would, however, include a decline of the nurses' social attainments and hence

		Nurses						
Day	Shift	1	2	3	...	n		Demand
1	Morning	x				x	≥	10
	Day		x				≥	8
	Night			x			≥	4
2	Morning	x					≥	12
	Day		x				≥	9
	Night					x	≥	5
3	Morning	x					≥	11
	Day			x			≥	11
	Night		x			x	≥	3
...	...						...	...
28	Morning			x			≥	10
	Day	x					≥	9
	Night		x			x	≥	3

Figure 6.1: Example of a nurse scheduling problem

entails a depreciation of the nurse profession, it would not solve the nurse shortage problem described in Section 6.1. On the contrary, less people would feel attracted by the nursing profession, making the problem only worse. A better way to decrease the number of nurses needed involves a rearrangement of the demand values over the different shifts. Indeed, it may be possible that the given workload distribution is just unfavorable to nicely fit a set of nurses to it. But how can one rearrange these demand values? To answer this question, one has to realize where these values come from, or, in other words, how these workload demands are determined.

The key observation for this research is that the distribution of the nurse workload over time is directly linked to the master surgery schedule. To see this, have a look at Figure 6.2. The left of this figure shows a particular workload pattern, while the right contains a partial surgery schedule. In the surgery schedule the columns indicate the time dimension, which is cyclic in days (Monday to Friday). The rows represent the different operating rooms R1-R4. We focus on the workload of a particular shift, e.g., the day shift of the third day in the time horizon. Assume this is a Tuesday. In this shift, eleven nurses are needed to perform the required

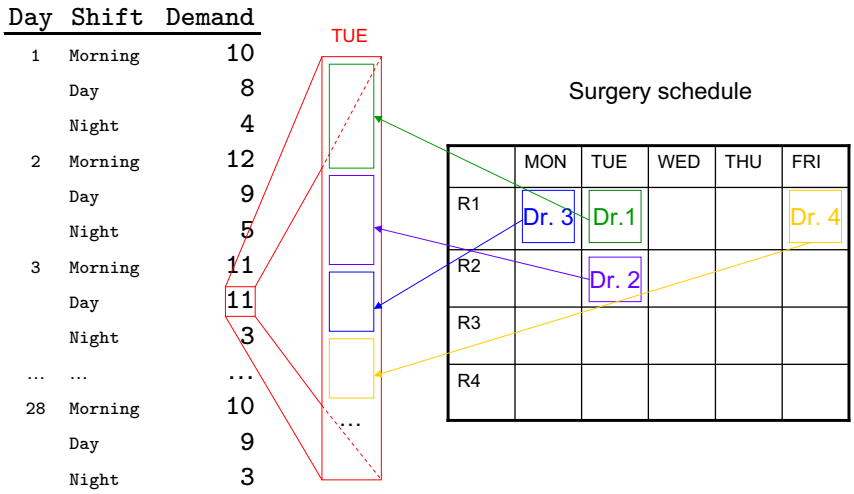

Figure 6.2: The surgery schedule determines the nurses' workload

work. A deeper analysis of the work content in this shift clarifies that a first part of the work goes to patients of surgeon 1, who is scheduled to operate on the same day, namely a Tuesday. Also a second part of the work goes to patients who are operated on this day (by surgeon 2). However, work may also go to patients who are operated on a previous day (surgeon 3) or even during a previous cycle (surgeon 4), but who are still in the hospital to recover from surgery.

Hence, it is not difficult to see that a modification in the surgery schedule can lead to a change in the workload distribution. Another way of viewing this is to observe that each workload pattern corresponds to a surgery schedule as indicated in Figure 6.3.

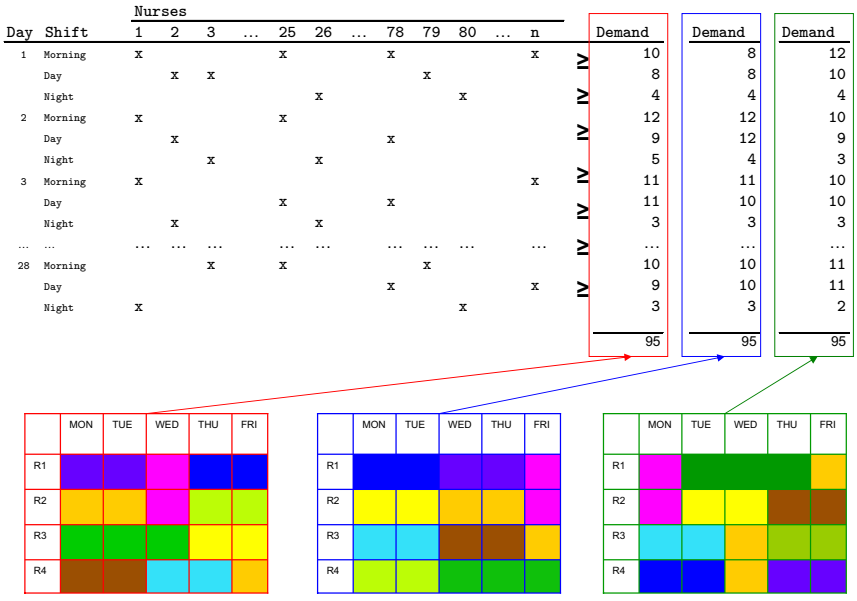

Figure 6.3: Each workload pattern corresponds to a surgery schedule

6.2.2 Schematic overview

Figure 6.4 contains a schematic overview of the general idea outlined in this chapter. First have a look at the nurse scheduling process at the right of this figure. The input for the nurse scheduling process consists of the restrictions implied on the individual nurse roster lines on the one hand and the workload distribution over time on the other hand. The workload distribution itself is determined by the master surgery schedule. In order to be able to deduce the workload from the surgery schedule one also has to know the workload contributions of each specific type of surgery. The dotted arrow at the bottom indicates the feedback that could be given from the nurse scheduling process to the surgery scheduling process in order to produce more favorable surgery schedules with respect to the resulting workloads. However, the freedom in modifying the surgery schedule is limited, since the master surgery schedule itself is restricted by a set of specific surgery constraints (e.g., capacity and demand constraints). It must be clear, however, that integrating the surgery scheduling process with the nurse scheduling process provides more flexibility in building the nurse schedules, since one has an instrument to make the workload distribution fit for the nurse schedules.

In what follows we will describe a mathematical model for implementing this idea. Therefore, we start by stating the standard nurse scheduling problem and discuss the column generation solution procedure for solving it. Then, we extend this model with the extra decision of the nurse scheduling process and show how the column generation solution procedure can easily cope with this extension. We focus on the minimization of the total required number of nurses. The reason for this objective is that it allows for a quantitative measure of the resulting benefits, i.e., the decrease in staffing cost. Obviously, this quantitative benefit can easily be turned into a qualitative benefit by employing the saved nurse(s) on moments when they are most needed.

6.2.3 The nurse scheduling problem

The nurse scheduling problem (NSP) consists of generating a configuration of individual schedules over a given time horizon (see Section 2.1). The configuration of nurse schedules is generated so as to fulfill collective agreement requirements and the hospital staffing demand coverage while minimizing the salary cost. Coverage constraints state how many nurses of appropriate skills have to be scheduled for

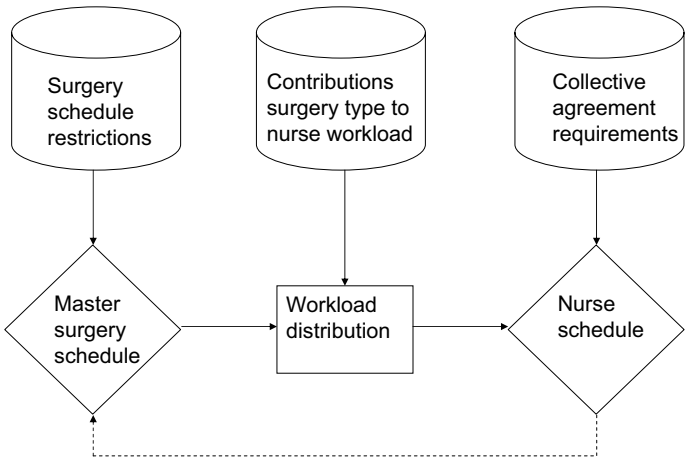

Figure 6.4: Schematic overview of the general idea

each demand period. For ease of exposition and without loss of generalization we consider all nurses equally-skilled throughout the remainder of this chapter.

Collective agreement requirements are rules that define acceptable schedules for individual nurses in terms of total workload, holidays, weekends off and shift transitions (e.g., a morning shift after a night shift is not allowed). These rules cannot be violated and dramatically reduce the set of feasible individual roster lines. Obviously, when building nurse schedules also a set of individual constraints, often called preference constraints, have to be taken into account. For instance, some nurses prefer to do night shifts, others do not. Again, for ease of exposition and without loss of generalization, we make abstraction of these differences in individual preferences and only consider those restrictions that are stated in the collective agreement rules and that consequently apply to all nurses. Hence, we present an integrated model that can be used to find optimal schedules for a homogeneous set of nurses.

In what follows we state the standard set covering model, which is often used for this type of problem. Let J be the set of feasible roster lines j and I be the set

of demand periods i. Let $d_i \in \{0, 1, 2, \dots\}$, $\forall i \in I$, denote the required number of nurses scheduled during period i. Furthermore, let a_{ij} be 1 if roster line j contains an active shift during period i and 0 otherwise. The general integer decision variable x_j, $\forall j \in J$, indicates the number of individual nurses that are scheduled by roster line j. Then, the nurse scheduling problem (NSP) can be stated as follows:

$$\text{Minimize} \sum_{j \in J} x_j \tag{6.1}$$

subject to:

$$\sum_{j \in J} a_{ij} x_j \geq d_i \qquad \forall i \in I \tag{6.2}$$

$$x_j \in \{0, 1, 2, \dots\} \qquad \forall j \in J \tag{6.3}$$

6.2.4 Solution procedure for the nurse scheduling problem

The integer program (IP) (6.1)-(6.3) is solved by first solving the linear programming relaxation and then using a branching scheme to drive the solution into integrality. As the number of possible roster lines an individual can work is usually too large to allow complete a-priori enumeration, column generation is often applied to solve the LP relaxation. Typically, the pricing step involves the solution of a dynamic programming shortest path problem (also called the *subproblem*) to find the legal column with the most negative reduced cost. Let π_i, $\forall i \in I$, denote the dual price of constraint (6.2). Then, the reduced cost of a new column (roster line) j is given by:

$$1 - \sum_{i \in I} a_{ij} \pi_i \tag{6.4}$$

A brief discussion of the solution procedure for this subproblem is given in Section 6.3.1. The process of adding new columns continues until no more columns *price out*, i.e., no more columns with negative reduced cost can be found. However, at that point the solution is not necessarily integral and applying a standard branch-and-bound procedure to the restricted master with its existing columns will not guarantee an optimal (or feasible) solution. Therefore, a branching scheme has

to be applied to drive the solution into integrality. After branching, new columns might price out favorably and hence have to be added to the model.

Since it does not lie in the scope of this work to discuss effective branching schemes for the NSP, we will not go into details about this, but instead refer the reader to the specialized literature. Barnhart et al. (1998) discuss appropriate branching strategies for solving a mixed integer program (MIP) using column generation. Since NSP (6.1)-(6.3) has identical restrictions on subsets (i.e., there are no subsets having a separate convexity constraint like in the trainee scheduling problem of Chapter 2), developing a branching scheme is a complex issue. Conventional integer programming branching on variables is not effective for reasons of symmetry and also because fixing variables destroys the structure of the subproblem. Vanderbeck and Wolsey (1996) developed a general rule in which one is branching on the constraints (see also Vanderbeck, 2000). The drawback is that the branching constraints cannot be used to eliminate variables and have to be added to the formulation explicitly. Hence, each branching constraint will contribute an additional dual variable to the reduced cost, complicating the pricing problem.

6.2.5 The generalized nurse scheduling problem

In the NSP the right hand side values of the coverage constraints (i.e., the d_i's in formulation (6.1)-(6.3)) are considered to be fixed. Nevertheless, coverage constraints are based on workload estimations that entail the summations of individual patient *workload contributions*. An individual patient workload contribution is determined by the *patient type*. The patient type can generally be described by three dimensions. The first dimension is the type of surgery the patient has undergone. The second is the number of periods the patient has already recovered. The third is the period to which the workload applies. For instance, some ailments may require increased care during nights.

The number and type of the patients that are present in the hospital at each moment in time is largely determined by the operating room schedule. Obviously, due to emergency cases and uncertainty in patient show-ups, patient recovery times etc., exact estimations are not possible. However, an in-depth analysis of the operating room schedule enables hospital executives to make a quite accurate prediction of the workload of the nurses. Moreover, they can reshape the workload distribution

by modifying the operating room schedule. In the long term, case mix planning decisions determine the overall workload. In the shorter term, the cyclic master surgery schedule determines the workload distribution over time.

The generalized nurse scheduling problem (GNSP) takes into account this extra dimension. Instead of assuming the demand values to be fixed, the GNSP considers them to be dependent on the number and type of patients undergoing surgery in the hospital at each moment. By manipulating the master surgery schedule hospital management can create (and choose between) a number of different workload distributions, further referred to as *workload patterns*. Let K denote the set of possible workload patterns that could be generated by modifying the surgery schedule. These will be obtained by enumerating all possible ways of assigning operating blocks to the different surgeons, subject to surgery demand and to capacity restrictions (for more details see Section 6.3.2). Each workload pattern k is described by a number of periodic demands $d_{ik} \in \{0, 1, 2, \dots\}$, $\forall i \in I$. Let z_k be 1 if the surgery schedule that corresponds to workload k is chosen and 0 otherwise. Then, the problem can be stated as follows:

$$\text{Minimize} \sum_{j \in J} x_j \tag{6.5}$$

subject to:

$$\sum_{j \in J} a_{ij} x_j \geq \sum_{k \in K} d_{ik} z_k \qquad \forall i \in I \tag{6.6}$$

$$\sum_{k \in K} z_k = 1 \tag{6.7}$$

$$x_j \in \{0, 1, 2, \dots\} \qquad \forall j \in J \tag{6.8}$$

$$z_k \in \{0, 1\} \qquad \forall k \in K \tag{6.9}$$

Constraint (6.7), further referred to as the workload convexity constraint, implies that exactly one workload pattern has to be chosen. In a feasible solution all z_k's but one equal 0. Hence, in constraint (6.6) only the corresponding d_{ik}'s are added in the right hand side values. It is easy to see that the NSP is a special case of the GNSP in which one z_k is fixed to be 1.

6.2.6 Solution procedure for the generalized nurse scheduling problem

In this part we show that the column generation approach to solve the LP relaxation of NSP can easily be extended to cope with the GNSP. Similarly to the roster lines, the number of possible workload patterns is usually too large to allow for complete a-priori enumeration. Also here the process starts with a limited subset of workload patterns and new patterns (columns) are added as needed. Therefore a second subproblem has to be solved. The generation of a new workload pattern boils down to the construction of a new master surgery schedule. The subproblem is constrained by a set of specific surgery schedule restrictions. Its objective is the minimization of the reduced cost of a new workload pattern. Let γ denote the dual price of the workload pattern convexity constraint (6.7). Then the reduced cost of a new workload pattern k is given by:

$$0 - \gamma + \sum_{i \in I} \pi_i d_{ik} \tag{6.10}$$

Obviously, the appropriate solution approach to price out a new workload pattern strongly depends on the characteristics of the master surgery schedule. In this chapter the workload pattern pricing problem is formulated as an IP and solved using a state-of-the-art optimization package (CPLEX). More details on this formulation can be found in Section 6.3.2.

6.3 Pricing problems

6.3.1 Generating a new roster line

Although the generation of a new roster line happens in a standard way (shortest path problem solved with recursive dynamic programming) (see, e.g., Caprara et al., 2003) and its exact implementation is not really necessary for understanding the general idea of this chapter, we briefly discuss the procedure. First, we summarize the restrictions that apply to a roster line.

As already mentioned earlier, this work is only concerned with collective agreement requirements and leaves individual preferences out of consideration. Specifically, we take into account five types of requirements when building a new roster line. First of all, a nurse cannot work more than one shift per day. Second, the overall

number of *active days*, i.e., days in which the roster line contains an *active shift* ('*day*', '*evening*' or '*night*'), cannot exceed a certain limit. Third, the maximum number of *consecutive* working days is also constrained. The same holds for the maximum number of consecutive rest days. A sequence of working days is further referred to as a *block*. Fourth, the number of so-called unpopular shifts (night shifts, weekend shifts) is limited for each roster line. Fifth, in a block certain shift transitions are not allowed. For instance, a nurse cannot switch from, say, a night shift to a morning shift without having a rest first.

Generating a new roster line is typically done using a dynamic programming recursion. To this aim, we define a table giving the minimum cost that can be achieved in days 1 to d by a roster line that, starting from a situation in which on day d a shift s is scheduled and in which between days d to n a certain number of active shifts f occurred, a certain number of unpopular shifts g occurred and a number of consecutive working or rest days h (including day d) is assigned. Formally, the entries of the table are of the form

$$\tau(d, f, g, s, h),$$

defined for $d = 1..n$, $f = 0..f_{max}$, $g = 0..g_{max}$, $s \in S$, $h = 0..h_{max}$ where n denotes the number of days in the scheduling horizon, f_{max} denotes the maximum number of working days in a roster line, g_{max} is the maximum penalty in terms of unpopular shifts, S is the set of shift types ("*day*", "*evening*", "*night*", "*rest*") and h_{max} is the maximum of both the maximum number of consecutive working days (h_1^{max}) and the maximum number of consecutive rest days (h_2^{max}). Let $p_{d,s}$ be the penalty cost for assigning an unpopular shift (d, s). Let A denote the set of allowed shift transitions (s, s') between two consecutive days on. We consider demand periods as being subsets of the shifts, i.e., no demand period can be spread over more than one shift. However, a shift can consist of more demand periods. Let $Q_{(d,s)}$ be the set of demand periods i that fall into shift (d, s). Let $\lambda_{d,s}$ be the total dual cost of a shift (d, s), i.e., $\lambda_{d,s} = \sum_{i \in Q_{(d,s)}} \pi_i$.

The computation of the entries in the table is done by starting at the beginning of the time horizon and by working forward by considering an insertion of a shift type s on the next day d of the roster line associated with an entry already computed. Therefore, we make use of recursive algorithm 2.

Algorithm 2 RECURSION(d, f, g, s, h)

if (d=0) **then**
  return 0; {*beginning of time horizon reached*}
else if ($\tau(d, f, g, s, h) \neq 999999999$) **then**
  return $\tau(d, f, g, s, h)$; {*state already visited, can be pruned*}
else
  $cost \leftarrow +\infty$;
  $min\_cost \leftarrow +\infty$;
  for (all shifts $\bar{s} \in S \backslash \{"rest"\}$) **do**
    if ($g + p_{d-1,\bar{s}} \leq g_{max}$) **AND** ($(\bar{s}, s) \in A$) **AND** ($f < f_{max}$) **then**
      if ($s \neq "rest"$) **then**
        if ($h < h^1_{max}$) **then**
          $cost \leftarrow \lambda_{d,s}$+RECURSION$(d-1, f+1, g+p_{d-1,\bar{s}}, \bar{s}, h+1)$; {*successive active shift*}
        end if
      else if ($s = "rest"$) **then**
        $cost \leftarrow$RECURSION$(d-1, f+1, g+p_{d-1,\bar{s}}, \bar{s}, 1)$; {*start active shift*}
      end if
    end if
    if ($cost < min\_cost$) **then**
      $min\_cost \leftarrow cost$;
    end if
  end for
  if ($s \neq "rest"$) **then**
    $cost \leftarrow \lambda_{d,s}$+ RECURSION$(d-1, f, g, "rest", 1)$; {*start rest*}
  else if ($s = "rest"$) **then**
    if ($h < h^2_{max}$) **then**
      $cost \leftarrow$ RECURSION$(d-1, f, g, "rest", h+1)$; {*successive rest*}
    end if
  end if
  if ($cost < min\_cost$) **then**
    $min\_cost \leftarrow cost$;
  end if
  return $\tau(d, f, g, s, h) \leftarrow min\_cost$;
end if

Before starting the recursion all entries of table $\tau(d, f, g, s, h)$ are initialized to 999999999. The minimal reduced cost of a new roster line can now easily be calculated by starting the recursion on day n and minimizing over each shift type (see algorithm 3).

Algorithm 3 FIND-NEW-ROSTER-LINE

```
{initialize all entries of τ}
for (d = 1 to n) do
  for (f = 0 to f_max) do
    for (g = 0 to g_max) do
      for (all shifts s ∈ S) do
        for (h = 0 to h_max) do
          τ(d, f, g, s, h) ← 999999999;
        end for
      end for
    end for
  end for
end for
cost ← +∞;
min_cost ← +∞;
{start the recursion}
for (all shifts s̄ ∈ S\{"rest"}) do
  if (p_{n,s̄} ≤ g_max) then
    cost ←RECURSION(n, 1, p_{n,s̄}, s̄, 1); {end with an active shift}
  end if
  if (cost < min_cost) then
    min_cost ← cost;
  end if
end for
cost ←RECURSION(n, 0, 0, "rest", 1); {end with a rest}
if (cost < min_cost) then
  min_cost ← cost;
end if
```

Once all the calculations are done, the best new roster line can easily be constructed backward. The overall space complexity of the dynamic programming recursion is

$$O(n \cdot f_{max} \cdot g_{max} \cdot |S| \cdot h_{max})$$

whereas the time complexity is (in the case that there are no forbidden shift transitions),

$$O(n \cdot f_{max} \cdot g_{max} \cdot |S| \cdot h_{max} \cdot |S|)$$

since each entry of the table is updated by considering up to $O(|S|)$ other entries.

6.3.2 Generating a new workload pattern

Each workload pattern corresponds to a particular surgery schedule. Hence, a new workload pattern can be obtained by building a new surgery schedule. The capacity preserved for the different surgeons (or, more generally, surgery groups) is already determined by the case mix planning (first stage, long term) and considered to be fixed in our application. Elective case scheduling (third stage) is also left out of consideration for two reasons. First of all, the impact of each specific elective case on the workload is rather limited. It is the type of surgery that determines the workload contribution, not the individual case. Second, it is very hard to predict the precise impact of the individual cases on the workload contribution at the moment that the nurse rosters have to be built. Often, at that moment, an important part of the elective surgery scheduling is still to be done.

The master surgery schedule is considered to be the tool for manipulating the workload distribution over time. This work is concerned with *cyclic* master surgery schedules. Cyclic schedules are schedules that are repeated after a certain time period (referred to as the cycle time). During such a cycle time there might be a number of time periods during which surgery cannot take place. These periods are referred to as the inactive periods, the others are active. Typically, cycle times are multitudes of weeks in which the weekends are inactive periods.

In our application, a new surgery schedule is built by solving an integer program. To find a new workload pattern with minimal reduced cost given the current set of roster lines and workload patterns, the objective function minimizes the dual price vector of the demand constraints (6.6) multiplied by the new demands. We deal with two types of constraint. Surgery demand constraints determine how many

blocks must be preserved for each surgeon. Capacity constraints ensure that the number of blocks assigned during each period do not exceed the available capacity. Let y_{rt} ($\forall r \in R$ and $t \in T$) be the number of blocks assigned to surgeon r in period t where T represents the set of active periods and R the set of surgeons. Let q_r be the number of blocks required by each surgeon r. Let b_t be the maximal number of blocks available in period t. Let $w_{rti} \in \Re^+$ denote the contribution to the workload of demand period i of assigning one block to surgeon r in period t. Then, the integer program to construct a new surgery schedule (and at the same time price out a new workload pattern k) is as follows:

$$\text{Minimize} \sum_{i \in I} \pi_i d_{ik} \tag{6.11}$$

subject to:

$$\sum_{t \in T} y_{rt} = q_r \qquad \forall r \in R \tag{6.12}$$

$$\sum_{r \in R} y_{rt} \leq b_t \qquad \forall t \in T \tag{6.13}$$

$$\sum_{r \in R} \sum_{t \in T} w_{rti} y_{rt} \leq d_{ik} \qquad \forall i \in I \tag{6.14}$$

$$y_{rt} \in \{0, 1, 2, \ldots, \min(q_r, b_t)\} \qquad \forall r \in R, \forall t \in T \tag{6.15}$$

$$d_{ik} \in \{0, 1, 2, \ldots\} \qquad \forall i \in I \tag{6.16}$$

The objective function (6.11) minimizes the reduced cost of a new workload pattern. Observe that the periodic demands d_{ik} are now an integral part of the decision process, whereas these are merely coefficients in the master problem (6.5)-(6.9). Constraint set (6.12) implies that each surgeon obtains the number of required blocks. Constraint set (6.13) ensures that the number of blocks assigned does not exceed the available number of blocks in each period. Constraint set (6.14) triggers the d_{ik}'s to the appropriate integer values. Finally, constraint sets (6.15) and (6.16) define y_{rt} and d_{ik} to be integer.

At first sight, constraint set (6.16) which requires the periodic demands d_{ik} to be integral seems to be redundant from a formulation point of view. Indeed, due to constraint (6.6) and the fact that $a_{ij} \in \{0, 1\}$ and $x_j \in \{0, 1, 2, \ldots\}$ fractional demand values d_{ik} would also be covered by the upper integer number of nurses.

The reason why we require the d_{ik}'s to be integral is to improve the computational efficiency of the overall branch-and-price algorithm. We come back to this issue in Section 6.6.1.

6.4 Overview of the branch-and-price algorithm

Algorithm 4 contains the pseudocode of the branch-and-price algorithm to solve the GNSP, while Figure 6.5 gives a schematic overview. The algorithm starts with a heuristic in order to find an initial solution. The heuristic generates only one workload pattern. This is done by building a surgery schedule for which the sum of the resulting quadratic demand values is minimized. The idea is to level the workload distribution as much as possible over the time horizon and as such to avoid the occurrence of peaks in the workload. This approach turned out to be beneficial for the surgery scheduling problem in which the expected shortage of beds has to be minimized (see Chapter 4).

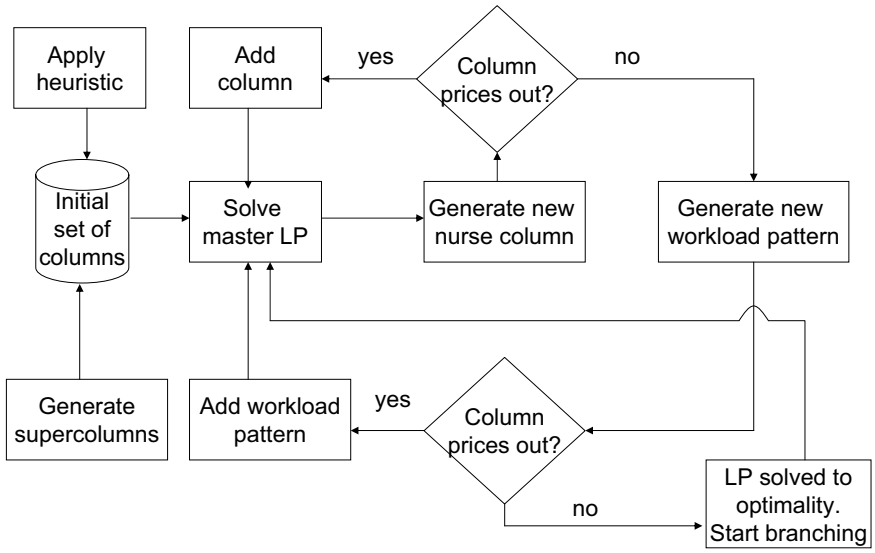

Figure 6.5: Schematic overview of the GNSP branch-and-price algorithm

Algorithm 4 BRANCH-AND-PRICE

```
apply heuristic to find initial solution;
if (solution found) then
  register nurse schedule and surgery schedule;
  upper_bound ← best solution found;
  initiate master with the columns making up the initial solution and (|I| + 1) supercolumns;
else
  upper_bound ← +∞;
  initiate master with |I| + 1 supercolumns;
end if
lower_bound ← −∞;
stop ← FALSE;
while (stop=FALSE) do
  LP_opt_found ← FALSE;
  {solve LP with column generation}
  while (LP_opt_found=FALSE) do
    LP_opt_found ← TRUE;
    improving_roster_line_found ← TRUE;
    while (improving_roster_line_found=TRUE) do
      RC_j ← FIND-NEW-ROSTER-LINE(j);
      if (RC_j < 0) then
        add new roster line to master;
        LP_opt_found ← FALSE;
        LP_opt ← SOLVE-MASTER-LP();
      else
        improving_roster_line_found ← FALSE;
      end if
    end while
    RC_k ← FIND-NEW-WORKLOAD-PATTERN(k);
    if (RC_k < 0) then
      add new workload pattern to master;
      LP_opt_found ← FALSE;
      LP_opt ← SOLVE-MASTER-LP();
    end if
  end while{LP solved to optimality}
  if (fractional_z) then
    expand node; {replace node by two child nodes}
  else if (LP_opt<best_integral_z) then
    best_integral_z ← LP_opt;
  end if
  if (no more nodes) then
    stop ← TRUE;
  else
    explore next node; {best-first}
    lower_bound ← bound_best_node;
    if (lower_bound ≥ upper_bound OR lower_bound ≥ best_integer_z) then
      stop ← TRUE;
    end if
  end if
  IP_opt ← SOLVE-MASTER-IP();
  if (IP_opt < upper_bound) then
    upper_bound ← IP_opt;
    register nurse schedule and surgery schedule;
  end if
end while
```

The surgery schedule is built with a mixed integer program (MIP) in which the constraints are given by (6.12)-(6.15) (replacing the d_{ik}'s by d_i's) and the objective is:

$$\text{Minimize} \sum_{i \in I} d_i^2$$

with d_i the required number of nurses in period i. To speed up the heuristic, the d_i's are not required to be integral. Instead, we round each d_i to the next upper integer after solution of the quadratic MIP. Given this workload pattern, new roster lines are added until the set of roster lines (one nurse scheduled by each roster line) completely satisfies the coverage constraints. A new roster line is found by solving exactly the same shortest path problem as in Section 6.3.1, but replacing the dual prices π_i by the remaining right hand side values d_i. As such each new roster line cuts the peaks in the remaining workload pattern until all demand is covered.

After detection of an initial solution, the objective value is saved as an upper bound and both the surgery schedule and the nurse schedule are registered. The columns making up the initial solution are entered into the master together with a number of supercolumns, which are needed to ensure feasibility of the master in each stage of the branch-and-bound algorithm.

The algorithm starts with the LP optimization loop in which, iteratively, a number of new roster lines and one new workload pattern are added until no more columns price out. Observe that roster lines are added until no more lines with negative reduced cost can be found, whereas only one workload pattern is generated, after which the generation of new roster lines restarts. This approach turned out to be the most successful, given the generally larger computation times to price out a new workload pattern.

Upon detection of the LP optimum, the solution is checked for fractional z_k's (workload patterns). If there still are fractional z_k's, branching is applied in order to drive the solution into an integral z solution (i.e., with only one z_k equal to 1 and all other equal to 0). The algorithm does not branch until an integral x_j (roster line) solution, because branching schemes for the x_j variables are not straightforward to implement and significantly complicate the roster line subproblem. Moreover, it provides no extra value for the extended model, which is the subject of this chapter.

Instead, we report lower and upper bounds for the required number of nurses to cover demand. The lower bound is the best possible solution with exactly one z_k equal to 1, however one for which the x_j's are not necessarily integral. Hence, the solution represented by the lower bound might not be interpretable in terms of the nurse schedule (e.g., schedule 2.5 nurses following roster line j). The upper bound on the other hand is the best found overall integer solution (with also integrality of the x_j's), which is fully interpretable.

In order to increase the lower bound as much as possible, the branch-and-bound tree is traversed in a best-search way. After each move in the tree, the master problem is solved with required integrality on both the x_j's and the z_k's. Because the integral master problem is often computationally very intensive, the MIP optimizer is interrupted after a specified time interval (e.g., 10 seconds). If a better solution is found, the upper bound decreases and as such the gap between the lower and upper bound tightens.

6.5 Branching

For reasons that are explained earlier, this work is only concerned with a branching scheme for driving the z_k's to integrality and leaves the x_j's out of consideration. We apply a constraint branching scheme (Ryan and Foster, 1981) which works as follows.

First we search for the highest fractional z_k. Let this be $z_{k'}$. Then we select another $z_k > 0$, say $z_{k''}$, and take the first period i for which $d_{ik'} \neq d_{ik''}$. If no such period exists, both z_k's represent essentially the same workload patterns and hence one of them can be set to 0 while its fractional value is added to the other one. Suppose we found period i' as the branching period with $d_{i'k'} < d_{i'k''}$. Then, we create two nodes in the branch-and-bound tree. In the left node we imply $d_{i'k} \leq d_{i'k'}$ and in the right node we imply $d_{i'k} \geq d_{i'k'} + 1$. Figure 6.6 visualizes this branching scheme. Else if $d_{i'k'} > d_{i'k''}$ we imply $d_{i'k} \leq d_{i'k''}$ in the left node and $d_{i'k} \geq d_{i'k''} + 1$ in the right node.

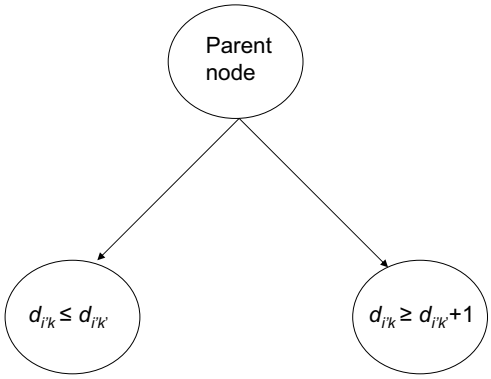

Figure 6.6: Binary branching scheme in the case of $d_{i'k'} < d_{i'k''}$

6.6 Computational performance issues

In this section we present some techniques that helped to improve the computational efficiency of the algorithm.

6.6.1 Integral versus fractional demand values

It has already been mentioned at the end of Section 6.3.2 that we imply the d_{ik}'s to be integral in the workload pattern pricing problem. Although this is not necessary from a formulation point of view, it has a substantially positive impact on the overall computational efficiency of the algorithm.

Implying integrality of the d_{ik}'s affects the computation time in two ways. On the one hand, there is a negative impact, because the pricing problem itself becomes more complex. On the other hand, there is a positive impact as far fewer columns can be found with negative reduced cost. Preliminary results indicate that this positive effect dramatically exceeds the negative effect. Consequently, the master LP is solved much faster when integrality of the d_{ik}'s is implied. Moreover, requiring integral demand values in the workload patterns makes the LP optimal solution substantially less fractional in terms of the x_j's. Hence, finding a global optimum

(with both integrality on the z_k's and on the x_j's) turns out to be much easier. In our application the gap between the lower and upper bound becomes much smaller.

6.6.2 Upper bound pruning for the workload pattern pricing problem

Basically, we are no longer interested in finding the column with the lowest reduced cost from the moment we know that this reduced cost will be positive anyway. Hence, we can act as if we already found a solution with reduced cost 0 by providing an appropriate upper bound. For the workload pattern subproblem, this observation yields dramatic time savings.

The reduced cost expression (6.4) consists of a fixed part and a variable part. By setting the upper bound equal to the fixed part with reverse sign, we act as if we found already a new column with reduced cost equal to 0. The reduced cost of a workload pattern is given by $0 - \gamma + \sum_{i \in I} \pi_i d_{ik}$. Consequently, we provide γ as an upper bound in the integer program (6.11)-(6.16).

Note that, since generating a new roster line is done using a backward dynamic recursion, upper bound pruning cannot be applied here. As an alternative, we wrote an A* algorithm (an enumeration approach entailing a forward recursion including both dynamic pruning and pruning based on bound comparisons; good discussions of A* can be found in Nilsson, 1980 and Barr et al., 1989). Dynamic pruning occurs if a state has already been visited at lower cost. For pruning based on bound comparisons we need an upper and a lower bound for the best new roster line. Since the reduced cost of a new roster line is given by $1 - \sum_{i \in I} a_{ij} \pi_i$, we can provide -1 as an initial upper bound in the A* algorithm. Obviously, this bound is decreased each time a better roster line is found. Starting from a certain day, a lower bound on the minimal cost path could be obtained by selecting for each remaining day the shift with the lowest total of corresponding dual prices, i.e:

$$\text{MIN}\Big\{ \underset{s \in S \setminus \{"rest"\}}{\text{MIN}} \{\lambda_{d,i}\}, 0 \Big\} \qquad \forall d$$

and summing up only the $(f_{max} - f)$ lowest values amongst these. In other words, for calculating the lower bound, we relax all constraints but the not-more-than-one-shift-per-day constraint and the maximum number of active days constraint.

Preliminary tests, however, indicated that the A* algorithm is outperformed by the backward dynamic recursion. Hence, the time saved from upper bound pruning in the A* algorithm is inferior to the time won by visiting each state only once in the purely dynamic backward recursion.

6.6.3 Two-phase approach for the workload pattern pricing problem

During the LP optimization loop it is not necessary to find the column with the most negative reduced cost, any column with negative reduced cost will do. Again, particularly for the computationally intensive workload pattern pricing problem, using this observation dramatically decreases the computation times. To guarantee optimality of the LP solution, a two-phase approach is applied for the workload pattern pricing problem. In the first phase, a certain time limit is set for the MIP optimizer. Only if no new workload pattern is found with negative reduced cost within this time limit, the algorithm enters the second phase. In this phase the time limit is undone and the optimizer is required to search until a feasible solution is found with negative reduced cost or it is proven that such a column does not exist.

6.6.4 Lagrange dual pruning

It is well known that Lagrangian relaxation can complement column generation in that it can be used in every iteration of the column generation scheme to compute a lower bound to the original problem with little additional computational effort (see, e.g., Van den Akker et al., 2002; Vanderbeck and Wolsey, 1996). If this lower bound exceeds an already found upper bound, the column generation phase can end without any risk of missing the optimum. Using the information from solving the reduced master and the information provided by solving the pricing problem for a new workload pattern k, it can be shown (see, e.g., Hans, 2001) that a lower bound is given by $\delta + RC_k\theta_k$ where δ is the objective value of the reduced master, RC_k is the reduced cost of a newly found workload pattern k and θ_k is a binary variable equal to 1 when RC_k is non-negative and set to zero, otherwise. This lower bound is referred to as the Lagrangian lower bound, since it can be shown that it equals the bound obtained by Lagrange relaxation.

Obviously, if the pricing procedure finds a negative reduced cost column during the first phase and hence does not enter the second phase (see Section 6.6.3) this lower bound cannot be used, because the workload pattern pricing problem has not been solved to optimality. Using CPLEX, it is very easy to set upper bounds, time limits and limits on the number of feasible solutions. Moreover, it can easily be verified if either the problem has been solved to optimality or optimization has prematurely ended because of an insufficient time limit.

6.7 Results

6.7.1 Test set

To test the algorithm, we generated a test set in the same way as we did in Chapter 4 for testing the surgery scheduling leveling algorithms (see Section 4.5.1). All surgery scheduling problems in this set involve a cycle time of 7 days. The last two days are not available to allocate operating room time (weekend), which is common practice. The problems differ with respect to five factors. These are: (1) the number of time blocks per day, (2) the number of surgeons, (3) the division of requested blocks per surgeon, (4) the number of operated patients per surgeon and finally (5) the length of stay (LOS) distribution. For these five factors, we used the same settings as in Table 4.2. If we consider two settings for each factor and repeat each factor combination 3 times, we obtain $2^5 * 3 = 96$ test instances.

Next, we generated some weights w_{rti} defining the contributions to the workload of period i of allocating a block to surgeon r in period t. These weights vary linearly with the number of patients of surgeon r operated in period t that are still in the hospital in period i. The patient's workload contribution generally decreases the longer the patient has already recovered in the hospital. In our test set the workload demand periods coincide with the shifts. Furthermore, we set the contribution to a 'day' shift two times as large as the one to an 'evening' shift and four times as large as the one to a 'night' shift. Obviously, although attempting to represent realistic scenarios, these contributions are chosen somewhat arbitrarily.

Third, we composed a set of collective agreement rules that apply on individual roster lines. The scheduling horizon amounted to 4 weeks or 28 days ($= n$). The maximum number of days an active shift could be scheduled ('day', 'evening' or

'night') was set to 20 (= f_{max}). Shifts during the weekends were marked as unpopular shifts: day and evening shifts got a penalty of 1, night shifts got a penalty of 2. The maximum number of consecutive working days was set to 6 (=$h_1^{max} = h_{max}$) and the maximum number of consecutive rest days was set to 3 (= h_2^{max}). Furthermore, we distinguished between two scenarios: a hard constrained scenario and a flexible one. Collective agreement rules in the hard constrained scenario differ from those in the flexible scenario on the following two points:

- In the hard constrained scenario there is only one shift type allowed within each block. In other words, no shift transitions between different shift types can occur without scheduling a rest first. In the flexible scenario all shift transitions are allowed, except the following three: a 'night' shift followed by a 'day' shift, a 'night' shift followed by an 'evening' shift or an 'evening' shift followed by a 'day' shift.

- In the hard constrained scenario the maximal penalty with respect to unpopular shifts is set to 4, whereas in the flexible scenario it is set to 8 (=g_{max}).

The branch-and-price algorithm was coded in C++ and linked with the CPLEX callable optimization library version 8.1 (ILOG, 2002). The tests were done on a 2.4 GHz Pentium 4 PC under the Windows XP operating system.

6.7.2 Savings

Table 6.1 contains the lower and upper bounds for both the NSP and the GNSP. In the NSP a surgery schedule is generated randomly. The resulting workload pattern contains the (fixed) right-hand side values of the coverage constraints. Then, the NSP is solved using column generation. In the GNSP new surgery schedules (and hence resulting workload patterns) are generated during search if needed. We distinguish between the flexible and the hard constrained scenario. To give an idea of the variability, the detailed bounds are provided for the first 9 and the last 9 problems of the problem set. The last line contains the average bounds over the whole set. Observe that the name of each problem (d*ijklm_n*) contains the information about the surgery scheduling subproblem: i stands for the setting of the first factor in Table 4.2 (0 for the first setting, 1 for the second), j for the second one, etc., and n for the iteration number.

From these results, one may conclude the following. First have a look at the upper bounds, which are after all the solutions that will be worked with. Although it is not guaranteed that the upper bound will be better (one might be lucky in the NSP and find the same or even a better overall integer solution), the upper bounds for the GNSP are generally better than those for the NSP. We compared them using a one-tailed paired T-test. The extremely small p-values obtained indicate that the differences are statistically significant both for the flexible and for the hard constrained case. The same results are obtained for the lower bounds. Unlike the upper bounds, the GNSP lower bounds are of course guaranteed to be at least as good as the NSP lower bounds.

When comparing the lower bounds for the NSP with the upper bounds for the GNSP, both scenarios entail different conclusions. The average lower bound for the NSP is lower than the average upper bound for the GNSP in the flexible scenario, whereas the reverse is true in the hard constrained scenario. Both differences turned out to be significant using a one-tailed paired T-test (again extremely small p-values). This observation can easily be explained. The stricter the collective agreement rules, the harder it is to nicely fit the nurse rosters into the required workload pattern in the NSP. As the workload pattern can be adapted in the GNSP, the GNSP includes more possible savings in the case of severe collective agreement requirements.

6.7.3 Interpretation of the savings

In the previous section we concluded that integrating the surgery scheduling process with the nurse scheduling process may yield important savings in terms of required nurses to hire. In this section we identify the source of these savings. Therefore, we provide an answer to the question: "Where lies the waste if one is considering the surgery schedule (and hence the workload distribution) as being fixed?" It turns out that the origin of the waste is twofold.

First, an unfavorable workload pattern may contain many workload demands that slightly exceed the workforce of x nurses, but that are dramatically inferior to the workforce of $x+1$ nurses. In terms of the d_{ik}'s one could think of many d_{ik}'s having a small decimal part, e.g., 6.1, 8.2, 4.05, etc. This type of waste is referred to as the waste due to the workforce surplus per shift. In many hospitals this kind of waste

is taken care of by simply scheduling x nurses instead of $x + 1$ nurses during those shifts. The result is a group of overworked nurses and an almost certain decrease in the quality of care. This illustrates how the GNSP approach can also be very useful for optimizing qualitative instead of quantitative objectives.

Second, waste also originates from the inflexibility of the roster lines, due to strict general agreement requirements. Because of this, no set of roster lines can be found that perfectly fit with the workload demand. This source of waste is further referred to as waste due to the inflexibility of roster lines.

Table 6.2 gives an overview of the importance of both sources of waste. We again distinguish between the flexible scenario and the hard constrained scenario. For each scenario there are three columns. The first column contains the total waste in terms of overstaffing in the NSP compared with the GNSP. These numbers are obtained by subtracting the upper bounds for the GNSP from those for the NSP. The second and third columns indicate the parts of this total waste that are due to the workforce surplus per shift and to the inflexibility of roster lines. These numbers can easily be calculated as follows. First, for both the NSP and the GNSP we make the sum of the (integral) demands of the chosen workload pattern. Call this number the total required workforce ($= \sum_{i \in I} d_i$ for the NSP and $\sum_{i \in I} \sum_{k \in K} d_{ik} z_k$ for the GNSP). Next, divide this number by the workforce per nurse ($= f_{max}$ in our application). This gives the minimal number of nurses that would be needed and can be obtained in the case of fully flexible roster lines. The difference between these numbers for the NSP and GNSP is the waste due to the workforce surplus per shift. The difference between the total waste and the waste due to the workforce surplus per shift is the waste due to the inflexibility of roster lines. Observe that these wastes may be negative (e.g., the waste due to workforce surplus per shift for problem d00000_2 is -1). This situation occurs when the gain with respect to one source of waste is so large that the best found solution for the GNSP includes a limited sacrifice with respect to the other source of waste.

The results in Table 6.2 clearly indicate that the importance of the source of waste strongly depends on the strictness of the general agreement requirements. The stricter these requirements are, the larger is the share of the waste due to the inflexibility of the roster lines.

6.7.4 Computational results

Table 6.3 and Table 6.4 contain the computational results for the flexible respectively hard constrained scenario. For the NSP, both the computation time and the number of generated roster lines are given. For the GNSP also the number of generated demand patterns and the number of nodes in the branch-and-bound tree are provided.

Obviously, the required computation times for the GNSP exceed those for the NSP. However, taking into account the explosion of the feasible solution space for the GNSP compared to the NSP, the increase in computation time is rather small. We can conclude that column generation is an excellent technique for solving the GNSP.

If we compare the flexible scenario with the hard constrained scenario, a couple of things attract our attention. First of all, observe that for the NSP the computation times for the flexible scenario surpass those for the hard constrained scenario, whereas for the GNSP the computation times for the hard constrained scenario exceed those for the flexible scenario. For the NSP this difference is statistically significant (extremely small p-value for a two-tailed paired T-test) and easy to explain. In the flexible scenario, much more legal roster lines exist and hence much more roster lines with negative reduced cost are found during the search process (on average 207.25 versus 106.07). Moreover, the time needed to price out a new roster line is also larger since the feasible state space contains more legal states.

For the GNSP the difference in computation time is not statistically significant at the 5% level (p-value of 0.113 for a two-tailed paired T-test). As again the number of generated roster lines is significantly smaller (very small p-value for a two-tailed paired T-test), the higher computation times for the constrained scenario must be produced by the higher number of generated workload patterns and the higher number of nodes in the branch-and-bound tree. The differences in number of generated workload patterns and in nodes in the branch-and-bound tree are found to be significant (very small p-values for two-tailed paired T-tests). This can easily be explained as follows. In the flexible scenario, it is unlikely that an extra workload pattern improves the overall solution. Thanks to the flexibility in the roster lines, an already very good solution can be found using a limited set of workload patterns. In the hard constrained case on the other hand, the inflexibility of the roster lines

might obstruct the detection of a good solution. In this case, it is far more likely that adding a new workload pattern improves the overall solution. We can conclude that the GNSP is easier to solve if the collective agreement requirements are less strict, whereas the reverse is true for the NSP.

As a final remark we note that a large part of the computation time goes to the calculation of an overall feasible solution in order to detect an upper bound after each move in the branch-and-bound tree in the GNSP and at the end of the column generation process in the NSP.

Table 6.1: Lower and upper bounds for the NSP and the GNSP

		Flexible scenario				Hard constrained scenario			
		NSP		GNSP		NSP		GNSP	
Nr.	Problem	lb	ub	lb	ub	lb	ub	lb	ub
1	d00000_0	15	17	13	15	19	19	16	17
2	d00000_1	26	28	25	27	34	35	31	31
3	d00000_2	25	27	23	25	32	32	28	29
4	d00001_0	40	42	39	41	49	50	47	48
5	d00001_1	45	47	44	46	54	54	52	53
6	d00001_2	94	96	92	94	112	113	109	110
7	d00010_0	34	36	32	35	43	43	40	40
8	d00010_1	40	42	38	40	49	50	47	47
9	d00010_2	28	30	26	27	34	35	32	33
...	...	...	...	...	...	...	...	...	...
88	d11101_0	96	98	94	96	114	115	112	113
89	d11101_1	99	102	97	99	119	120	116	116
90	d11101_2	122	125	119	121	145	146	142	143
91	d11110_0	83	85	80	82	101	102	96	96
92	d11110_1	111	113	109	111	138	139	132	132
93	d11110_2	58	60	56	58	73	74	67	68
94	d11111_0	252	254	249	252	303	304	296	297
95	d11111_1	119	122	116	119	143	144	139	140
96	d11111_2	135	137	131	133	162	163	156	157
	Average	70.18	72.43	68.33	70.44	86.07	86.73	81.91	82.61

Table 6.2: Interpretation of the savings

		Flexible scenario			Hard constrained scenario		
Nr.	Problem	Total waste	Waste due to workforce surplus per shift	Waste due to inflexibility of roster lines	Total waste	Waste due to workforce surplus per shift	Waste due to inflexibility of roster lines
1	d00000_0	2	1.2	0.8	2	1.2	0.8
2	d00000_1	1	1.2	-0.2	4	1.4	2.6
3	d00000_2	1	2	-1	3	1	2
4	d00001_0	1	1.2	-0.2	2	0.6	1.4
5	d00001_1	2	1	1	1	0.2	0.8
6	d00001_2	2	1.6	0.4	3	0	3
7	d00010_0	1	1.4	-0.4	3	1	2
8	d00010_1	1	1.6	-0.6	3	1.6	1.4
9	d00010_2	1	1.8	-0.8	2	-0.6	2.6
...	...	...	...	...	...	...	...
88	d11101_0	2	1.4	0.6	2	0.6	1.4
89	d11101_1	2	1.8	0.2	4	0.2	3.8
90	d11101_2	1	2.2	-1.2	3	0.2	2.8
91	d11110_0	2	1.6	0.4	6	0.8	5.2
92	d11110_1	2	0.8	1.2	7	0.6	6.4
93	d11110_2	1	2	-1	6	1.8	4.2
94	d11111_0	2	1.2	0.8	7	0.2	6.8
95	d11111_1	2	1.8	0.2	4	-0.6	4.6
96	d11111_2	1	2	-1	6	0.6	5.4
	Average	1.58	1.43	0.16	4.11	0.28	3.84

Table 6.3: Computational results for the flexible scenario

		NSP		GNSP			
			Roster		Roster	Workload	
Nr.	Problem	Time (s)	lines	Time (s)	lines	patterns	Nodes
1	d00000_0	43484	150	44422	183	2	0
2	d00000_1	44063	174	51000	196	2	0
3	d00000_2	46423	235	45438	213	2	0
4	d00001_0	44078	173	46000	221	2	0
5	d00001_1	43829	167	45172	190	2	0
6	d00001_2	44844	212	48829	238	3	0
7	d00010_0	45266	211	70359	274	2	0
8	d00010_1	46311	237	185623	535	17	8
9	d00010_2	44594	208	166892	640	32	13
...	...	...	...	...	...	...	...
88	d11101_0	44390	213	47984	243	2	0
89	d11101_1	44953	228	52031	257	2	0
90	d11101_2	44734	230	56438	280	2	0
91	d11110_0	46203	252	358811	555	30	15
92	d11110_1	45265	238	1765257	815	128	59
93	d11110_2	47359	200	423125	507	28	14
94	d11111_0	46360	347	69266	381	2	0
95	d11111_1	45719	243	59063	319	2	0
96	d11111_2	45048	237	251970	512	14	6
	Average	44146.04	207.25	99008.57	310.31	5.93	1.95

Table 6.4: Computational results for the hard constrained scenario

		NSP		GNSP			
			Roster		Roster	Workload	
Nr.	Problem	Time (s)	lines	Time (s)	lines	patterns	Nodes
1	d00000_0	453	46	66953	263	8	4
2	d00000_1	500	70	55359	304	18	6
3	d00000_2	422	64	11781	111	2	0
4	d00001_0	468	77	609	81	2	0
5	d00001_1	453	74	687	95	3	0
6	d00001_2	672	120	782	127	2	0
7	d00010_0	4250	113	216064	470	79	43
8	d00010_1	953	113	323236	448	129	47
9	d00010_2	750	80	201970	459	102	39
...	...	...	...	...	...	...	...
88	d11101_0	2125	122	1656	130	2	0
89	d11101_1	1531	126	2625	146	2	0
90	d11101_2	1610	149	2109	159	2	0
91	d11110_0	1938	123	456191	439	58	17
92	d11110_1	1500	152	1228851	508	92	45
93	d11110_2	5438	101	102470	310	10	1
94	d11111_0	8000	251	12265	264	2	0
95	d11111_1	4859	143	19359	185	2	0
96	d11111_2	4922	153	1809557	600	221	83
	Average	1215.52	106.07	153927.85	226.05	28.08	10.81

6.8 Conclusions and further research

This chapter has presented an integrated approach for building nurse and surgery schedules. It has been shown how the column generation technique, often employed for solving nurse scheduling problems, can easily be extended to cope with this integrated approach. The approach involves the solution of two types of pricing problems, the first one is solved with a standard dynamic programming recursion, the second one by aims of a state-of-the-art mixed integer programming optimizer. A constraint branching scheme has been proposed to drive the solution into integrality with respect to the workload patterns while the integrality of the roster lines was left out of the scope of this chapter. Finally, some techniques were presented that helped to improve the computational efficiency of the branch-and-price algorithm.

Our computational results indicate that considerable savings could be achieved by using this approach to build nurse and surgery schedules. We simulated problems for a large range of surgery scheduling instances and distinguished between a flexible and a hard constrained scenario with respect to the collective agreement requirements. Our conclusions can be summarized as follows. First of all, column generation is a good technique to deal with the extra problem dimension of modifying surgery schedules. Second, the obtained gains originate from two sources of waste: waste due to the workforce surplus per shift and waste due to the inflexibility of roster lines. Third, unlike the NSP, the GNSP turns out to become harder to solve when the collective agreement requirements are more strict.

Obviously, in real-life hospital environments it is not so easy to modify the master surgery schedule. As the surgery schedule can be considered to be the main engine of the hospital, it not only has an impact on the workload distribution for nurses, but also on several other resources throughout the hospital. Think for instance about anaesthetists, equipment, radiology, laboratory tests and consultation. This observation yields a negative as well as a positive note for the reasoning in this chapter. The negative note is that the possible savings obtained through integrating the nurse and the surgery scheduling process are in real-life probably much smaller, due to the smaller flexibility with which surgery schedules can be modified. The positive note is that not only savings in nurse staffing costs are possible, but also in other related resource types, by integrating the scheduling of these resources

with the surgery scheduling process. This is probably the main contribution of this chapter. This work clearly shows the benefits of integrating scheduling processes in health care environments and moreover proposes a methodology for implementing the heart of a supporting ICT infrastructure.

Possible topics for further research include the application of this approach in a real-world environment involving a detailed report on the experienced merits and pitfalls. From a theoretical point of view, it would be interesting to develop similar techniques for one or more of the other resource types stated above.

Chapter 7

Conclusions and future directions

This last chapter gives some general conclusions on the material that has been presented in this thesis, states our main contributions and brings forward some ideas for future research. Section 7.1 reviews our study on the trainee scheduling problem that was presented in Chapter 2. Section 7.2 concludes our work concerning the development of master surgery schedules reviewing the material that was presented in Chapters 3, 4 and 5. The third section elaborates a little bit further on the integration of different scheduling areas, that has been illustrated by the integrated approach for nurse and operating room scheduling in Chapter 6. In each of these first three sections, we briefly discuss some of the future challenges that we believe represent promising directions for future work in these areas. The fourth and last section states some general reflections on important issues for further research in health care scheduling.

7.1 Trainee scheduling

Chapter 2 has described a new decomposition approach for a staff scheduling problem that, although frequently encountered in almost every hospital, has been neglected in the literature so far. The problem consists of building a long term schedule for advanced medical students, called trainees. Building such a schedule is often

a complicated task: the schedule has to satisfy all the coverage constraints and formation requirements, must assign the activities in a consecutive way and must take as much as possible the individual preferences for having weeks-off into account. The problem distinguishes from the classic nurse scheduling problem mainly in the presence of the formation requirements, the need for consecutiveness of performing the different activities and the longer time horizon for which schedules are built. Based on what we have called activity patterns, we have proposed a new formulation for this problem. This new formulation made it possible to develop an efficient approach that decomposes the problem on the activities. The comparison with a more traditional approach that decomposes the problem on the staff members instead of on the activities, has yielded some interesting insights on the computational efficiency as well as on the modeling power of the new approach. Finally, we have presented some heuristic extensions which made it possible to deal with more general problem formulations and larger problem instances.

We have considered only one type of specialization for which the trainees usually can carry out the complete internship within one particular hospital department. Other specializations, however, require that the trainees perform internships at different hospital departments. Think, for instance, of anaesthetists, who need to be trained to provide assistance at different medical disciplines. Consequently, the scheduling of anaesthetist trainees contains an extra level of hierarchy. Indeed, before building annual schedules for each discipline, it has to be decided which trainees will serve which disciplines. This task essentially comes down to an assignment problem in which the trainees' history record and individual preferences play a major role. The problem is complicated by the fact that the different disciplines compete for the higher skilled (typically higher years) trainees and hence also some equity measures have to be taken into account.

Additionally, it might be interesting to investigate whether an integrated approach for the higher level assignment problem and the lower level scheduling problem leads to an improvement of the developed trainee schedules. We think of a decomposition approach in which the master problem determines the composition of the trainee sets for each discipline and the subproblem consists of building the trainee schedules given these input sets. Within a general scientific framework, it would again be interesting to see which problem properties complicate the problem and how far we can go in solving the problem to optimality. If real-life problem dimen-

sions turn out to be to large to tackle with an exact approach, one may come up with (meta)heuristic approaches or hybridized exact/heuristic methods.

For which concerns general nurse scheduling, we consider problem decomposition, the exploitation of problem-specific information and the hybridization of exact and heuristic methodologies as important directions for further research.

Throughout this dissertation we have shown that problem decomposition, i.e., intelligently breaking up larger problems into smaller, easier to handle subproblems, is a very promising approach to solve difficult problems. In general, real-life nurse scheduling problems can be very large and consequently we believe that problem decomposition is an important direction for further research into these kinds of problems.

Many constraints and requirements have not been addressed (explicitly) in the nurse scheduling literature so far. Nevertheless, specific models and algorithms that use problem-specific information can dramatically increase the efficiency of solution methodologies. Consider, for instance, the trainee scheduling problem of Chapter 2 in which each trainee has to perform each activity exactly once. We have used this problem-specific information to develop an alternative, more efficient solution approach. Since this constraint is often encountered in the development of educational schedules, we believe that the exploitation of this observation may turn out to be advantageous for other applications as well.

Throughout this thesis, we have often started from an exact method that could solve small instances of the problem and hybridized our approach with heuristic methodologies in order to cope with larger instances (see, for instance, Section 2.7.3). Moreover, we do not believe that one method or technique is going to increase the uptake of health care scheduling on its own. Therefore, we have experimented with several methodologies trying to combine the strong points of different techniques. No need to argue that we strongly believe that the hybridization of different search techniques includes an important scope for further research.

7.2 Operating room scheduling

Chapters 3, 4 and 5 have dealt with master surgery scheduling. Our study has focussed on the impact of the cyclic master surgery schedule on the load of various resources throughout the hospital. Chapter 3 has introduced a visualization model to assist in the development of the master surgery schedule. Subsequently, Chapters 4 and 5 have concentrated on the bed occupancy as a function of the master surgery schedule. The proposed models aim at leveling the bed occupancy in order to avoid the occurrence of contra-productive peaks.

When we carried out the case studies, we have noticed that an enormous amount of data is being recorded inside hospitals, in particular with respect to the operating room. Not only start and end times of surgery, also the usage of various resources like the nursing personnel, the anaesthetist, the use of specialized equipment, the preceding tests, the need for blood analysis, the succeeding tests, the hospitalization bed, etc., are all registered. These data are saved in a central database that represents a treasure of information. Nowadays, this database already serves multiple purposes. It is not only used to measure the efficiency with which the operating room and various resources are used. The supply and inventory management information system and the human resource management information system are linked to this central database. Also, modern cost accounting systems like activity based costing rely heavily on this database. For more details on cost accounting in hospitals, in particular with respect to activity based costing, we refer the reader to Upda (1996) and Cardinaels et al. (2004). Often, important decisions like the purchase of expensive, specialized equipment or the opening of a new operating room are preceded by a profound analysis of historical data.

Unfortunately, up to the present, a lot of opportunities have still remained unexplored. The information extracted from the central database is currently too much focussed on what we call the 'a posteriori' analysis. The replenishment of inventories, the wages administration, the evaluation of personnel, the cost accounting and the calculation of the efficiency with which resources are used, are all carried out afterwards and do not directly contribute to an improvement of the current system. It is our conviction that data from the central database also can be transformed to information that can be used to actively amend the current practice. Consequently, the information is not only used to 'a posteriori' respond to what has happened, but

also to 'a priori' decision making in order to better streamline the current practice. Chapter 5 has illustrated this point by extracting from the database the probabilistic distributions of the patients' length of stay and the number of operated patients. We have shown how this information can be used for developing a better surgery schedule with respect to the bed occupancy.

We believe that there are many other opportunities left unexplored. Increasingly more data are recorded in the central database system. The continuous progress in database technology enables us to extract a maximum of information out of these data. The introduction of PC terminals at each corner of the hospital, even within the operating room, has made it possible to track the patient from the first arrival until the moment the patients leaves the hospital. This has led to a patient-oriented view on health care management instead of a hospital-oriented view. An important contribution to this process lies in the development of clinical pathways, also called patient care pathways or integrated care pathways. Integrated care pathways are multidisciplinary care plans that detail the essential steps in the care of patients with a specific clinical problem and describe the expected progress of the patient. They facilitate the introduction into clinical practice of clinical guidelines and systematic, continuing audit into clinical practice. They help in communication with patients by giving them access to a clearly written summary of their expected care plan and progress over time (Campbell et al., 1998). Clinical Pathways were introduced in the early 1990s in the UK and the USA, and are being increasingly used throughout the developed world, including Belgium (see, e.g., Sermeus et al., 2001; Vanhaecht et al., 2002; Vanhaecht and Sermeus, 2003, Vanherck et al., 2004; De Bleser et al., 2004). Because of their focus on the complete process rather than on the individual steps, clinical pathways are naturally compatible with activity based costing systems (see, e.g., Asadi and Baltz, 1996).

All this information can be used to develop models and algorithms that improve the current scheduling practices and, as such, optimize the resource management inside hospitals. In this dissertation, we have only dealt with the operating room, which we have considered the heart of the system. It can be argued that the preceding and succeeding steps in the clinical pathway are taken into account by considering the resource consumption patterns for each type of surgery (see Section 3.2), however, a more detailed integration with these pathways needs to be addressed. Moreover, we have only coped with long term master surgery scheduling, i.e., the

allocation of operating room blocks to surgeons or surgical groups in the operating room theatre. It must be clear that there is considerable scope of research left with respect to more detailed scheduling, that is scheduling on patient level. Think, for instance, of determining the order in which the individual patients are operated and the impact this order has, not only on the operating room efficiency, but also on the resources used throughout the rest of the hospital.

7.3 Integrating different scheduling areas

Chapter 6 has presented a model and algorithm for integrating the nurse and surgery scheduling. We have obtained very promising results concerning the computational efficiency of this approach as well as for the gains that could be achieved by coupling both scheduling areas. The study was however merely theoretic. The developed model still needs to be applied on a real-life data set in order to be able to draw more profound conclusions on the value of the integration approach. Undoubtedly, this will raise difficulties we have not foreseen yet. First of all, usually different people are involved in building the staff and surgery schedules. Often, the nurse and operating room scheduling at itself is already a fairly complex task and it is questionable whether the people that build the schedules nowadays accept an extra factor to be taken into account. It would take a clear communication of the possible profits and a large power of persuasion to convince them that, on the contrary, integration leads to more flexibility and hence less constraints. A second, practical difficulty lies in the fact that the software systems that support both scheduling practices are currently often not coupled which complicates the implementation of our algorithm.

We are convinced that many other scheduling areas within hospitals could be integrated in order to obtain overall better solutions. Too often, a scheduler must take a number of constraints for granted as they follow from decisions that have been made by other units within the same hospital. Good communication is a first necessary condition. However, well-thought-out algorithms that can manage the increased flexibility are a key issue for successful integration of different scheduling practices. We strongly believe that there is significant scope for further research in this direction. A specific idea is the integrated vacation scheduling of all hospital staff who are, in one way or another, connected to each other in the sense that

their work content is determined by the presence or absence of the other. Think, for instance, of the nurses, the surgeons and the anaesthetists who all have to be present in order to perform surgery. Sometimes, operating rooms are closed for a number of days as a response to sparse personnel occupancy in general vacation periods. The decisions concerning the timing of operating room close downs could be integrated with the staff vacation planning so that the overall schedule better meets the personnel's preferences for having a week-off.

7.4 General reflections on further research

7.4.1 Robustness

The ability to cope with unforseen events is a key issue for scheduling in the uncertain world of health care. Indeed, we wish to avoid situations where, for instance, one person calling in sick or one patient not showing up causes a chain reaction of disruptions throughout the hospital. Robust schedules are schedules that are protected against these kinds of events, for instance, by considering the expertise of staff on beforehand in order to make sure that people can easily be replaced at each moment of time. It could be argued that the operating room scheduling models that have been presented in Chapters 3, 4 and 5 address, in a particular sense, the issue of robustness. Indeed, a leveled bed occupancy for the elective cases decreases the probability of a bed shortage (due to an unexpected peak in the urgent cases) which at its turn protects the proper execution of the schedule. Nevertheless, compared to project scheduling, where robustness has recently become an important issue (see, e.g., Leus, 2003), robustness has received little attention in health care scheduling so far. We see a lot of opportunities for further research with respect to this issue.

7.4.2 Persuading all people involved

A key issue in the exploitation of all the available information in order to use it for streamlining the entire process of health care delivery is the cooperation of all people involved. Nowadays, physicians in particular are not always convinced of the benefits of registering all kinds of data in the central database, for they merely consider it to be an extra administrative load. If this information is only used to evaluate the physicians, they will not likely to change their mind on this

issue. Physicians often not realize that, after the patients, they gain in the first place by improving the scheduling and management of resources. Therefore, it is important that the benefits of the data collection are clearly communicated to all people involved.

7.4.3 Graphical user interface

We consider the development of the graphical user interfaces that hide the technical details of the algorithms from the end user as an important enrichment for the research that has been presented in this dissertation. Academic research on scheduling is often only concerned with developing algorithms to solve different versions of a theoretic problem. The focus lies too much on finding 'better' solutions in less computational effort. Nevertheless, the ease of use is a far more crucial issue for the acceptance of a decision support system for scheduling than the solution optimality and the required computation time. Moreover, a user-friendly, graphical user interface is just a necessity in order to apply the algorithms in a real-life case and to obtain useful feedback from the people involved.

Jeroen Beliën
October 2005

Appendices

The appendices refer to the work carried out in Chapter 4. The notation is defined in Sections 4.2 and 4.3.

APPENDIX A

In this Appendix it is shown that the variance of the Z_i's varies linearly in the decision variables. In the derivation that follows, the next two rules are frequently applied:

$$var(a_0 + \sum_{i=1}^{n} a_i x_i) = \sum_{i=1}^{n} a_i^2 var(x_i) + \sum_{i=1}^{n} \sum_{j=1}^{i-1} 2a_i a_j cov(x_i; x_j) \qquad \text{(A-1)}$$

$$\text{If } x_i \text{ and } x_j \text{ are independent, then } cov(x_i; x_j) = 0 \qquad \text{(A-2)}$$

For the derivation it is important to keep in mind that the number of patients staying on the same day in the hospital but having 'entered' it via different blocks are completely independent of each other. There is only dependency between patient numbers coming from one and the same block in one and the same cycle. Let us now start the derivation:

$$var(Z_i) = var(\sum_{s \in S} \sum_{j \in A} U_{ijs}) \qquad \text{(A-3)}$$

Applying (A-1) and knowing that the covariances between the different U_{ijs}'s are all zero (the number of patients occupying a bed operated in different operating room blocks are independent of each other) gives:

$$var(Z_i) = \sum_{s\in S}\sum_{j\in A} var(U_{ijs}) \tag{A-4}$$

$$= \sum_{s\in S}\sum_{j\in A} var\Big(\sum_{f=0}^{\lfloor \frac{m_s - dist(i,j)}{l} \rfloor} \sum_{g=1}^{x_{js}} \sum_{d=dist(i,j)+fl}^{m_s} D_{sd}\Big) \tag{A-5}$$

Recall that D_{sd} is a stochastic variable that stands for the number of patients who stay exactly d days in the hospital after one block of surgery by surgeon s. The first and third summations divide the D_{sd} variables into their cycles, i.e., the number of patients staying in the hospital on day i after surgery by surgeon s on day j can be divided according to the cycle in which they entered the system. For $f = 0$ all patients entered in the current cycle ($= 0$) are added, for $f = 1$ all patients entered in the previous cycle are added, etc. The second summation indicates the number of blocks (x_{js}) for which patients are added. Writing this in full gives:

$$\begin{aligned} var(Z_i) = \sum_{s\in S}\sum_{j\in A} var\Big(& \sum_{d=dist(i,j)}^{m_s} D_{sd} + \sum_{d=dist(i,j)}^{m_s} D_{sd} + \sum_{d=dist(i,j)}^{m_s} D_{sd} + \dots \\ & + \sum_{d=dist(i,j)+l}^{m_s} D_{sd} + \sum_{d=dist(i,j)+l}^{m_s} D_{sd} + \sum_{d=dist(i,j)+l}^{m_s} D_{sd} + \dots \\ & + \sum_{d=dist(i,j)+2l}^{m_s} D_{sd} + \sum_{d=dist(i,j)+2l}^{m_s} D_{sd} + \sum_{d=dist(i,j)+2l}^{m_s} D_{sd} + \dots \\ & + \dots \Big) \end{aligned} \tag{A-6}$$

The first line indicates all patients entered in the current cycle. The different terms in this line indicate the different blocks that 'produce' patients. The second line indicates the numbers entered in the previous cycle, etc. The number of patients occupying a bed on a particular day i having undergone surgery more than 1 cycle ago is of course completely independent of the new patients entered in the current cycle. In general, the number of patients operated in the same block, but in different cycles, are independent of each other. Hence, application of again (A-1) and (A-2) gives:

$$var(Z_i) = \sum_{s\in S}\sum_{j\in A}\Big[var\Big(\sum_{d=dist(i,j)}^{m_s} D_{sd} + \sum_{d=dist(i,j)}^{m_s} D_{sd} + \sum_{d=dist(i,j)}^{m_s} D_{sd} + \dots\Big) + var\Big(\sum_{d=dist(i,j)+l}^{m_s} D_{sd} + \sum_{d=dist(i,j)+l}^{m_s} D_{sd} + \sum_{d=dist(i,j)+l}^{m_s} D_{sd} + \dots\Big) + var\Big(\sum_{d=dist(i,j)+2l}^{m_s} D_{sd} + \sum_{d=dist(i,j)+2l}^{m_s} D_{sd} + \sum_{d=dist(i,j)+2l}^{m_s} D_{sd} + \dots\Big) + \dots\Big] \tag{A-7}$$

Within each cycle the number of patients coming from one block on a particular day assigned to surgeon s is independent from the number coming from another block on the same day assigned to the same surgeon s. Applying again (A-1) and (A-2) gives:

$$var(Z_i) = \sum_{s\in S}\sum_{j\in A}\Big[var(\sum_{d=dist(i,j)}^{m_s} D_{sd}) + var(\sum_{d=dist(i,j)}^{m_s} D_{sd}) + var(\sum_{d=dist(i,j)}^{m_s} D_{sd}) + \dots + var(\sum_{d=dist(i,j)+l}^{m_s} D_{sd}) + var(\sum_{d=dist(i,j)+l}^{m_s} D_{sd}) + var(\sum_{d=dist(i,j)+l}^{m_s} D_{sd}) + \dots + var(\sum_{d=dist(i,j)+2l}^{m_s} D_{sd}) + var(\sum_{d=dist(i,j)+2l}^{m_s} D_{sd}) + var(\sum_{d=dist(i,j)+2l}^{m_s} D_{sd}) + \dots + \dots\Big] \tag{A-8}$$

Rewriting it in the shorter summation notation:

$$var(Z_i) = \sum_{s\in S}\sum_{j\in A}\sum_{f=0}^{\lfloor\frac{m_s-dist(i,j)}{l}\rfloor}\sum_{g=1}^{x_{js}} var(\sum_{d=dist(i,j)+fl}^{m_s} D_{sd}) \tag{A-9}$$

Applying (A-1) gives:

$$var(Z_i) = \sum_{s\in S}\sum_{j\in A}\sum_{f=0}^{\lfloor\frac{m_s-dist(i,j)}{l}\rfloor}\sum_{g=1}^{x_{js}}\Big(\sum_{d=dist(i,j)+fl}^{m_s} var(D_{sd}) + \sum_{d_1=dist(i,j)+fl}^{m_s}\sum_{d_2=dist(i,j)+fl}^{d_1-1} 2cov(D_{sd_1}; D_{sd_2})\Big) \tag{A-10}$$

The covariances between the D_{sd} variables coming from the same block are of course not zero, but negative. Intuitively this can be seen as follows. The more patients that stay, e.g., exactly 1 day, the less patients will stay exactly 2, 3, etc., days and vice versa. The total is always n_s. The variance and covariance formulas for the individual variables of a multinomial distribution are as follows:

$$var(D_{sd}) = p_{sd}(1 - p_{sd})n_s \tag{A-11}$$

$$cov(D_{sd_1}; D_{sd_2}) = -p_{sd_1}p_{sd_2}n_s \tag{A-12}$$

Alternatively, these formulas could be obtained by observing that the individual variables of a multinomial distribution are binomial processes with probability of 'success' p_{sd} and n_s trials. Applying these formulas gives:

$$\begin{aligned} var(Z_i) = \sum_{s\in S}\sum_{j\in A} \sum_{f=0}^{\lfloor \frac{m_s - dist(i,j)}{l} \rfloor} \sum_{g=1}^{x_{js}} \Big(& \sum_{d=dist(i,j)+fl}^{m_s} p_{sd}(1-p_{sd})n_s \\ & - \sum_{d_1=dist(i,j)+fl}^{m_s} \sum_{d_2=dist(i,j)+fl}^{d_1-1} 2p_{sd_1}p_{sd_2}n_s \Big) \end{aligned} \tag{A-13}$$

Since g is merely a summation index and hence does not influence the calculation, summing up from 1 to x_{js} is the same as multiplying by x_{js}:

$$\begin{aligned} var(Z_i) = \sum_{s\in S}\sum_{j\in A} \sum_{f=0}^{\lfloor \frac{m_s - dist(i,j)}{l} \rfloor} \Big(& \sum_{d=dist(i,j)+fl}^{m_s} p_{sd}(1-p_{sd})n_s \\ & - \sum_{d_1=dist(i,j)+fl}^{m_s} \sum_{d_2=dist(i,j)+fl}^{d_1-1} 2p_{sd_1}p_{sd_2}n_s \Big) x_{js} \end{aligned} \tag{A-14}$$

This expression can be further simplified by observing that also the summation over f can be turned into a multiplication. The summation is replaced by respectively the factor $\lceil d/l \rceil$ and $\lceil d_2/l \rceil$ indicating how many cycle times the D_{sd} variables contribute to respectively the variance and the covariance:

$$var(Z_i) = \sum_{s\in S}\sum_{j\in A}\Big(\sum_{d=dist(i,j)}^{m_s} p_{sd}(1-p_{sd})n_s\lceil d/l\rceil - \sum_{d_1=dist(i,j)}^{m_s}\sum_{d_2=dist(i,j)}^{d_1-1} 2p_{sd_1}p_{sd_2}n_s\lceil d_2/l\rceil\Big)x_{js} \qquad \text{(A-15)}$$

In conclusion, the variance of each Z_i varies linearly in the decision variables.

APPENDIX B

In this Appendix it is shown how the expressions for both mean and variance of the daily bed occupancy are extended such that they incorporate stochastic patient arrivals. For the mean, we make use of the following theorem on conditional means:

$$E(Y) = E[E(Y|X)] \tag{B-1}$$

In words, the overall mean equals the mean of the conditional means. Applied to our problem: let N_s be a stochastic variable representing the number of patients for surgeon s. $k = 1, ..., q_s$ are the different (discrete) states of this variable with h_{sk} being the probability and n_{sk} the corresponding number of patients in state k for surgeon s.

$$E(U_{ijs}) = E[E(U_{ijs}|N_s)] \tag{B-2}$$

$$= \sum_{k=1}^{q_s} h_{sk} \Big(\sum_{d=dist(i,j)}^{m_s} p_{sd} n_{sk} \lceil d/l \rceil \Big) \tag{B-3}$$

For the variance, we make use of the following theorem on conditional variances:

$$var(Y) = E[var(Y|X)] + var[E(Y|X)] \tag{B-4}$$

In words, the overall variance equals the sum of (1) the mean of the conditional variances and (2) the variance of the conditional means. Applied to our problem:

$$var(U_{ijs}) = E[var(U_{ijs}|N_s)] + var[E(U_{ijs}|N_s)] \tag{B-5}$$

Elaborating the first term gives:

$$E[var(U_{ijs}|N_s)] = \sum_{k=1}^{q_s} h_{sk} \Big(\sum_{d=dist(i,j)}^{m_s} p_{sd}(1 - p_{sd}) n_{sk} \lceil d/l \rceil - \sum_{d_1=dist(i,j)}^{m_s} \sum_{d_2=dist(i,j)}^{d_1-1} 2 p_{sd_1} p_{sd_2} n_{sk} \lceil d_2/l \rceil \Big) \tag{B-6}$$

Elaborating the second term gives:

$$
\begin{aligned}
var[E(U_{ijs}|N_s)] &= \sum_{k=1}^{q_s} h_{sk}\Big(E(U_{ijs}|N_s) - E(U_{ijs})\Big)^2 \\
&= \sum_{k=1}^{q_s} h_{sk}\Big(\sum_{d=dist(i,j)}^{m_s} p_{sd} n_{sk} \lceil d/l \rceil - \sum_{k=1}^{q_s} h_{sk}\Big(\sum_{d=dist(i,j)}^{m_s} p_{sd} n_{sq} \lceil d/l \rceil\Big)\Big)^2
\end{aligned}
\tag{B-7}
$$

Combining all this gives us the variance of U_{ijs}:

$$
\begin{aligned}
var(U_{ijs}) = &\sum_{k=1}^{q_s} h_{sk}\Big(\sum_{d=dist(i,j)}^{m_s} p_{sd}(1-p_{sd}) n_{sk} \lceil d/l \rceil \\
&- \sum_{d_1=dist(i,j)}^{m_s} \sum_{d_2=dist(i,j)}^{d_1-1} 2 p_{sd_1} p_{sd_2} n_{sk} \lceil d_2/l \rceil\Big) \\
&+ \sum_{k=1}^{q_s} h_{sk}\Big(\sum_{d=dist(i,j)}^{m_s} p_{sd} n_{sk} \lceil d/l \rceil - \sum_{k=1}^{q_s} h_{sk}\Big(\sum_{d=dist(i,j)}^{m_s} p_{sd} n_{sq} \lceil d/l \rceil\Big)\Big)^2
\end{aligned}
\tag{B-8}
$$

In conclusion, incorporating numbers of patients following a multinomial discrete probability distribution preserves the linearity of both mean and variance of the daily bed occupancy.

List of Figures

List of Tables

Bibliography

Abdennadher, S. & Schlenker, H. (1999). INTERDIP - An interactive constraint based nurse scheduler, *Proceedings of The First International Conference and Exhibition on The Practical Application of Constraint Technologies and Logic Programming (PACLP99), London.*

Abernathy, W. J., Baloff, N., Hershey, J. & Wandel, S. (1973). A three-stage manpower planning and scheduling model - A service-sector example, *Operations Research* **21**(3): 693–711.

Agin, N. (1966). Optimum seeking with branch and bound, *Management Science* **13**: 176–185.

Aickelin, U. & Dowsland, K. A. (2000). Exploiting problem structure in a genetic algorithms approach to a nurse rostering problem, *Journal of Scheduling* **31**: 139–153.

Aickelin, U. & Dowsland, K. A. (2004). An indirect genetic algorithm for a nurse-scheduling problem, *Computers and Operations Research* **31**(5): 761–778.

Aickelin, U. & White, P. (2004). Building better nurse scheduling algorithms, *Annals of Operations Research* **128**: 159–177.

Alfares, H. & Bailey, J. (1997). Integrated project task and manpower scheduling, *IIE Transactions* **29**: 711–718.

Alfares, H., Bailey, J. & Lin, W. (1999). Integrating project operations and personnel scheduling with multiple labor classes, *Production Planning & Control* **10**: 570–578.

Anzai, M. & Miura, Y. (1987). Computer program for quick work scheduling of nursing staff, *Medical Informatics* **12**: 43–52.

Arthur, J. L. & Ravindran, A. (1981). A multiple objective nurse scheduling model, *AIIE Transactions* **13**: 55–60.

Asadi, M. J. & Baltz, W. A. (1996). Activity-based costing for clinical paths. An example to improve clinical cost & efficiency, *Journal of the Society for Health Systems* **5**(2): 1–7.

Assuralia (2005). De nationale uitgaven in de gezondheidszorg. Assur info to the point, January 2005.
*www.assuralia.be

Azaiez, M. N. & Al Sharif, S. S. (2005). A 0-1 goal programming model for nurse scheduling, *Computers and Operations Research* **32**: 491–507.

Bailey, J. & Field, J. (1985). Personnel scheduling with flexshift models, *Journal of Operations Management* **5**: 327–338.

Bard, J. F., Binici, C. & deSilva, A. H. (2003). Staff scheduling at the United States Postal Service, *Computers and Operations Research* **30**: 745–771.

Bard, J. F. & Purnomo, H. W. (2005a). A column generation-based approach to solve the preference scheduling problem for nurses with downgrading, *Socio-Economic Planning Sciences* **39**: 193–213.

Bard, J. F. & Purnomo, H. W. (2005b). Preference scheduling for nurses using column generation, *European Journal of Operational Research* **164**: 510–534.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P. & Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs, *Operations Research* **46**: 316–329.

Barr, A., Cohen, P. R. & Feigenbaum, E. A. (eds) (1989). *The Handbook of Artificial Intelligence*, Vol. 4, Addison-Wesley, Reading, Massachusetts.

Beaumont, N. (1997). Scheduling staff using mixed integer programming, *European Journal of Operational Research* **98**: 473–484.

Bell, P., Hay, G. & Liang, Y. (1986). A visual interactive decision support system for workforce (nurse) scheduling, *Information Systems and Operational Research* **24**(2): 133–144.

Bellanti, F., Carello, G., Della Croce, F. & Tadei, R. (2004). A greedy-based neighborhood search approach to a nurse rostering problem, *European Journal of Operational Research* **153**: 28–40.

Bellman, R. (1957). *Dynamic Programming*, Princeton University Press.

Berrada, I. (1993). *Planification d'horaires du personnel infirmier dans un établissement hospitalier*, Ph.D. dissertation, Université de Montréal, Canada.

Berrada, I., Ferland, J. & Michelon, P. (1996). A multi-objective approach to nursescheduling with both hard and soft constraints, *Socio-Economic Planning Science* **30**: 183–193.

Blake, J. T. & Carter, M. W. (2002). A goal programming approach to strategic resource allocation in acute care hospitals, *European Journal of Operational Research* **140**: 541–561.

Blake, J. T. & Carter, M. W. (2003). Physician and hospital funding options in a public system with decreasing resources, *Socio-Economic Planning Sciences* **37**: 45–68.

Blake, J. T., Dexter, F. & Donald, J. (2002). Operating room manager's use of integer programming for assigning block time to surgical groups: A case study, *Anesthesia and Analgesia* **94**: 143–148.

Blake, J. T. & Donald, J. (2002). Mount Sinai hospital uses integer programming to allocate operating room time, *Interfaces* **32**: 63–73.

Blau, R. (1985). Multishift personnel scheduling with a microcomputer, *Personnel Administrator* **20**(7): 43–58.

Blau, R. & Sear, A. (1983). Nurse scheduling with a microcomputer, *Journal of Ambulance Care Management* **6**: 1–13.

Blöchliger, I. (2004). Modeling staff scheduling problems. A tutorial, *European Journal of Operational Research* **158**: 533–542.

Bosi, F. & Milano, M. (2001). Enhancing constraint logic programming branch and bound techniques for scheduling problems, *Software Practice and Experience* **31**: 17–42.

Bowers, J. & Mould, G. (2004). Managing uncertainty in orthopaedic trauma theatres, *European Journal of Operational Research* **154**: 599–608.

Bradley, D. & Martin, J. (1990). Continuous personnel scheduling algorithms: A literature review, *Journal of the Society of Health Systems* **2**(2): 8–23.

Brandeau, M. L., Sainfort, F. & Pierskalla, W. P. (2004). *Operations research and health care: A handbook of methods and applications*, Kluwer Academic Publishers.

Brusco, M. J. & Jacobs, L. W. (1993). A simulated annealing approach to the cyclic staff-scheduling problem, *Naval Research Logistics* **40**: 69–84.

Buhaug, H. (2002). Long waiting lists in hospitals - Operational research needs to be used more often and may provide answers, *British Medical Journal* **324**: 252–253.

Burke, E. K., Cowling, P. I., De Causmaecker, P. & Vanden Berghe, G. (2001). A memetic approach to the nurse rostering problem, *Applied Intelligence* **15**: 199–214.

Burke, E. K., De Causmaecker, P., Petrovic, S. & Vanden Berghe, G. (2001a). Fitness evaluation for nurse scheduling problems, *Proceedings of the Congress on Evolutionary Computation (CEC), Seoul, Korea, May 27-30*, Vol. 2, pp. 1139–1146.

Burke, E. K., De Causmaecker, P., Petrovic, S. & Vanden Berghe, G. (2001b). Variable neighbourhood search for nurse rostering problems, *Proceedings of the 4th Metaheuristics International Conference (MIC 2001), Porto, Portugal, July 16-20*, Vol. 2, pp. 755–760.

Burke, E. K., De Causmaecker, P., Petrovic, S. & Vanden Berghe, G. (2002). A multi-criteria metaheuristic approach to nurse rostering, *Proceedings of the 2002 Congress on Evolutionary Computation (CEC2002), Honolulu, Hawaii, USA, May 12-17*, pp. 1197–1202.

Burke, E. K., De Causmaecker, P., Petrovic, S. & Vanden Berghe, G. (2003). Chapter 7: Variable neighborhood search for nurse rostering problems, *in* M. G. C. Resende & J. P. de Sousa (eds), *METAHEURISTICS: Computer Decision-Making*, Kluwer (Combinatorial Optimization Book Series), pp. 153–172.

Burke, E. K., De Causmaecker, P., Petrovic, S. & Vanden Berghe, G. (2006). Metaheuristics for handling time interval coverage constraints in nurse scheduling. To appear in *Applied Artificial Intelligence*.

Burke, E. K., De Causmaecker, P. & Vanden Berghe, G. (1998). A hybrid tabu search algorithm for the nurse rostering problem, *Selected Papers from the 2nd Asia Pacific Conference on Simulated Evolution and Learning*, pp. 187–194.

Burke, E. K., De Causmaecker, P. & Vanden Berghe, G. (1999). A hybrid tabu search algorithm for the nurse rostering problem, *Simulated Evolution and Learning*, Vol. 1585, Springer, pp. 187–194.

Burke, E. K., De Causmaecker, P. & Vanden Berghe, G. (2004). Chapter 44: Novel meta-heuristic approaches to nurse rostering problems in Belgian hospitals, *in* J. Leung (ed.), *The Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, CRC Press, pp. 1–18.

Burke, E. K., De Causmaecker, P., Vanden Berghe, G. & Van Landeghem, H. (2004). The state of the art of nurse rostering, *The Journal of Scheduling* **7**: 441–499.

Burke, E. K. & Kendall, G. (eds) (2005). *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, Springer.

Campbell, H., Hotchkiss, R., Bradshaw, N. & Porteous, M. (1998). Integrated care pathways, *British Medical Journal* **316**(7125): 133–137.

Cappanera, P. & Gallo, G. (2001). A multi-commodity flow approach to the crew rostering problem, *Technical Report TR-01-08*, Dip. di Informatica, Univ. di Pisa.

Caprara, A., Monaci, M. & Toth, P. (2003). Models and algorithms for a staff scheduling problem, *Mathematical Programming* **98**: 445–476.

Cardinaels, E., Roodhooft, F. & Van Herck, G. (2004). Drivers of cost system development in hospitals: Results of a survey, *Health Policy* **69**(2): 239–252.

Carter, J. (2000). Timing is everything in the OR, *Health Management Technology* **21**: 80–81.

Carter, M. (2002). Health care: Mismanagement of resources, *ORMS Today* **19**: 26–32.

Carter, M. & Lapierre, S. (1999). Scheduling emergency room physicians, *Report 99-23*, University of Toronto, Canada.

Cheang, B., Li, H., Lim, A. & Rodrigues, B. (2003). Nurse rostering problems - A bibliographic survey, *European Journal of Operational Research* **151**: 447–460.

Chen, J. G. & Yeung, T. W. (1993). Hybrid expert-system approach to nurse scheduling, *Computers in Nursing* **11**(4): 183–190.

Cheng, B., Lee, J. & Wu, J. (1997). A nurse rostering system using constraint programming and redundant modeling, *IEEE Transactions on Information Technology in Biomedicine* **1**(1): 44–54.

Chvátal, V. (1983). *Linear Programming*, W.H. Freeman and Co, New York.

Clerkin, D., Fos, P. J. & Petry, F. E. (1995). A decision-support system for hospital bed assignment, *Hospital and Health Services Administration* **40**: 386–400.

Dantzig, G. B. & Wolfe, P. (1960). Decomposition principle for linear programs, *Operations Research* **8**: 101–111.

Darmoni, S. J., Fajner, A., Mahe, N., Leforestier, A., Vondracek, M., O., S. & Baldenweck, M. (1995). Horoplan: computer-assisted nurse scheduling using constraint-based programming, *Journal of the Society for Health Systems* **5**(1): 41–54.

De Bleser, L., Vlayen, J., Vanhaecht, K. & Sermeus, W. (2004). Classifying clinical pathways, *Studies in Health Technology and Informatics* **110**: 9–14.

De Causmaecker, P. & Vanden Berghe, G. (2003). Relaxation of coverage constraints in hospital personnel rostering, *Practice and Theory of Automated Timetabling, Fourth International Conference, Gent*, Vol. 2740, Springer, pp. 129–147.

Demeulemeester, E. & Herroelen, W. S. (1992). A branch-and-bound procedure for the multiple resource-constrained project scheduling problem, *Management Science* **38**: 1803–1818.

Demeulemeester, E. & Herroelen, W. S. (2002). *Project scheduling - A research handbook*, Kluwer Academic Publishers, Boston.

Desrochers, M., Desrosiers, J. & Solomon, M. M. (1992). A new optimization algorithm for the vehicle routing problem with time windows, *Operations Research* **40**: 342–354.

Desrosiers, J., Soumis, F. & Desrochers, M. (1984). Routing with time windows by column generation, *Networks* **14**: 545–565.

Dexter, F., Macario, A. & O'Neill, L. (2000). Scheduling surgical cases into overflow block time - Computer simulation of the effects of scheduling strategies on operating room labor costs, *Anesthesia and Analgesia* **90**: 980–988.

Dexter, F., Macario, A. & Traub, R. D. (1999). Which algorithm for scheduling add-on elective cases maximizes operating room utilization?, *Anesthesiology* **91**: 1491–1500.

Dexter, F. & Traub, R. D. (2000). Determining staffing requirements for a second shift of anesthetists by graphical analysis of data from operating room information systems, *Anesthesia and Analgesia* **68**: 31–36.

Dexter, F. & Traub, R. D. (2002). How to schedule elective surgical cases into specific operating rooms to maximize the efficiency of use of operating room time, *Anesthesia and Analgesia* **94**: 933–942.

Dexter, F., Traub, R. D. & Lebowitz, P. (2001). Scheduling a delay between different surgeons' cases in the same operating room on the same day using upper prediction bounds for case durations, *Anesthesia and Analgesia* **92**: 943–946.

Dowsland, K. (1998). Nurse scheduling with tabu search and strategic oscillation, *European Journal of Operational Research* **106**: 393–407.

Dowsland, K. & Thompson, J. M. (2000). Solving a nurse scheduling problem with knapsacks, networks and tabu search, *Journal of the Operational Research Society* **51**: 825–833.

Dreyfus, S. E. & Law, A. M. (1977). *The art and theory of dynamic programming*, Academic Press, New York, NY.

Dumas, M. (1984). Simulation modeling for hospital bed planning, *Simulation* **8**: 69–78.

Dumas, M. (1985). Hospital bed utilization: An implemented simulation approach to adjusting and maintaining levels, *Health Services Research* **20**: 43–61.

Duraiswamy, N., Welton, R. & Reisman, A. (1981). Using computer simulation to predict ICU staffing needs, *Journal of Nursing Administration* **11**: 39–44.

Easton, F. & Mansour, N. (1993). A distributed genetic algorithm for employee staffing and scheduling problems, *Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo*, Morgan Kaufmann Publishers, pp. 360–367.

Ernst, A., Jiang, H., Krishnamoorthy, M. & Sier, D. (2004). Staff scheduling and rostering: A review of applications, methods and models, *European Journal of Operational Research* **153**: 3–27.

Folkard, S. & Tucker, P. (2003). Shift work, safety and productivity, *Occupational Medicine* **53**: 95–101.

Franz, L. S., Baker, H. M., Leong, G. K. & Rakes, T. R. (1989). A mathematical model for scheduling and staffing multiclinic health regions, *European Journal of Operational Research* **41**(3): 277–289.

Fries, B. (1976). Bibliography of operations research in health-care systems, *Operations Research* **24**: 801–804.

Galassi, M., Davies, J., Theiler, J., Gough, B., Jungman, G., Booth, M. & Rossi, F. (2003). *GNU Scientific Library Reference Manual Edition 1.3*, Network Theory LTd.

Gamache, M., Soumis, M., Marquis, G. & Desrosiers, J. (1999). A column generation approach for large scale aircrew rostering problems, *Operations Research* **47**: 247–262.

Gerchak, Y., Guptar, D. & Henig, M. (1996). Reservation planning for elective surgery under uncertain demand for emergency surgery, *Management Science* **42**: 321–334.

Gill, P. E., Murray, W. & Wright, M. H. (1982). *Practical Optimization*, Academic Press, New York.

Gilmore, P. C. & Gomory, R. E. (1961). A linear programming approach to the cutting stock problem, *Operations Research* **9**: 849–859.

Gorunescu, F., McClean, S. I. & Millard, P. H. (2002). A queueing model for bed-occupancy management and planning of hospitals, *Journal of the Operational Research Society* **53**: 19–24.

Griffiths, J. D., Price-Lloyd, N., Smithies, M. & Williams, J. E. (2005). Modelling the requirement for supplementary nurses in an intensive care unit, *Journal of the Operational Research Society* **56**: 126–133.

Guinet, A. & Chaabane, S. (2003). Operating theatre planning, *International Journal of Production Economics* **85**: 69–81.

Hamilton, D. M. & Breslawski, S. (1994). Operating room scheduling: Factors to consider, *Association of Operating Room Nurses Journal* **59**: 665–680.

Hans, E. W. (2001). *Resource loading by branch-and-price techniques*, Ph.D. dissertation, Twente University Press, Enschede, The Netherlands.

Hans, E. W., Wullink, G., van Houdenhoven, M. & Kazemier, G. (2005). Robust surgery loading, *Technical Report Beta-wp141*, dep. Operational Methods for Production and Logistics, University of Twente.

Harris, R. A. (1985). Hospital bed requirements planning, *European Journal of Operational Research* **25**: 121–136.

Huang, M. D., Romeo, F. & Sangiovanni-Vincentelli, A. (1986). An efficient general cooling schedule for simulated annealing, *IEEE International Conference on Computer-Aided Design*, pp. 381–384.

Hughes, W. L. & Soliman, S. Y. (1985). Short-term case mix management with linear programming, *Hospital and Health Services Administration* **30**: 52–60.

Ikegami, A. & Niwa, A. (2003). A subproblem-centric model and approach to the nurse scheduling problem, *Mathematical Programming* **97**(3): 517–541.

ILOG (2002). *ILOG CPLEX 8.1 User's Manual*.

Isken, M. (2004). An implicit tour scheduling model with applications in healthcare, *Annals of Operations Research* **128**: 91–109.

Isken, M. & Hancock, W. (1991). A heuristic approach to nurse scheduling in hospital units with non-stationary, urgent demand, and a fixed staff size, *Journal of the Society for Health Systems* **2**(2): 24–41.

Jan, A., Yamamoto, M. & Ohuchi, A. (2000). Evolutionary algorithms for nurse scheduling problem, *Proceedings of the 2000 Congress on Evolutionary Computation (CEC2000), San Diego*, pp. 196–203.

Jan, A., Yamamoto, M. & Ohuchi, A. (2002). Search algorithms for nurse scheduling with genetic algorithms, *Operations Research/Management Science at Work, the International Series in Operations Research & Management Science*, Vol. 43, Kluwer Academic Publishers, pp. 149–161.

Jans, R. (2002). *Capacitated lot sizing problems: New applications, formulations and algorithms*, Ph.D. dissertation, Faculteit Economische en Toegepaste Economische Wetenschappen, Katholieke Universiteit Leuven, Belgium.

Jaszkiewicz, A. (1997). A metaheuristic approach to multiple objective nurse scheduling, *Foundations of Computing and Decision Sciences* **22**(3): 169–184.

Jaumard, B., Semet, F. & Vovor, T. (1998). A generalized linear programming model for nurse scheduling, *European Journal of Operational Research* **107**: 1–18.

Jebali, A., Alouane, A. B. H. & Ladet, P. (2006). Operating rooms scheduling, *International Journal of Production Economics* **99**: 52–62.

Jiménez, V. M. & Marzal, A. (1999). Computing the K shortest paths: A new algorithm and an experimental comparison, *in* J. S. Vitter & C. D. Zaroliagis (eds), *Lecture Notes in Computer Science Series*, Spinger-Verlag, pp. 15–29.

Johnson, E. L. (1989). Modeling and strong linear programs for mixed integer programming, *in* S. W. Walace (ed.), *Algorithms and Model Formulations in Mathematical Programming*, NATO ASI Series, pp. 1–41.

Johnson, E. L., Nemhauser, G. L. & Savelsbergh, M. W. P. (2000). Progress in linear programming based branch-and-bound algorithms: An exposition, *INFORMS Journal on Computing* **12**: 1–48.

Jun, J. B., Jacobson, S. H. & Swisher, J. R. (1999). Applications of discrete event simulation in health care clinics: A survey, *Journal of the Operational Research Society* **50**: 109–123.

Kawanaka, H., Yamamoto, K., Yoshikawa, T., Shinogi, T. & Tsuruoka, S. (2001). Genetic algorithm with the constraints for nurse scheduling problem, *Proceedings of the 2000 Congress on Evolutionary Computation (CEC2001), Seoul*, pp. 1123–1130.

Kellerer, H., Pferschy, U. & Pisinger, D. (2004). *Knapsack Problems*, Springer-Verlag.

Kim, S.-C. & Horowitz, I. (2002). Scheduling hospital services: The efficacy of elective-surgery quotas, *Omega - the International Journal of Management Science* **30**: 335–346.

Kim, S.-C., Horowitz, I. & Buckley, T. A. (2000). Flexible bed allocation and performance in the intensive care unit, *Journal of Operations Management* **18**: 427–443.

Kirkpatrick, S., Gerlatt, C. D. J. & Vecchi, M. P. (1983). Optimization by simulated annealing, *Science* **220**: 671–680.

Klein, R. W., Dittus, R. S., Roberts, S. D. & Wilson, J. R. (1993). Simulation modeling and health-care decision making, *Medical Decision Making* **13**: 347–354.

Kostreva, M. & Jennings, K. (1991). Nurse scheduling on a microcomputer, *Computers and Operations Research* **18**: 731–739.

Kumar, A. & Ozdamar, L. (2004). International comparison of health care systems, *International Journal of the Computer, the Internet and Management* **12**: 81–95.

Lapierre, S. D., Batson, C. & McCaskey, S. (1999). Improving on-time performance in health care organizations: A case study, *Health Care Management Science* **2**: 27–34.

Leus, R. (2003). *The generation of stable project plans: Complexity and exact algorithms*, Ph.D. dissertation, Faculteit Economische en Toegepaste Economische Wetenschappen, Katholieke Universiteit Leuven, Belgium.

Li, H., Lim, A. & Rodrigues, B. (2003). A hybrid AI approach for nurse rostering problem, *Proceedings of the 2003 SAC symposium on Applied Computing*, pp. 730–735.

Litvak, E. & Long, M. C. (2000). Cost and quality under managed care: Irreconcilable differences?, *The American Journal of Managed Care* **6**: 305–312.

Marcon, E., Kharraja, S. & Simonnet, G. (2003). The operating theatre planning by the follow-up of the risk of no realization, *International Journal of Production Economics* **85**: 83–90.

Markowitz, H. M. (1959). *Portfolio Selection: Efficient Diversification of Investments*, John Wiley & Sons, New York.

Mason, A. J. & Smith, M. C. (1998). A nested column generator for solving rostering problems with integer programming, *International Conference on Optimisation: Techniques and Applications*, pp. 827–834.

McManus, M. L., Long, M. C., Cooper, A. & Litvak, E. (2004). Queuing theory accurately models the need for critical care resources, *Anesthesiology* **100**: 1271–1276.

Mehrotra, A., Murphy, K. E. & Trick, M. A. (2000). Optimal shift scheduling: A branch-and-price approach, *Naval Research Logistics* **47**: 185–200.

Meisels, A., Gudes, E. & Solotorevsky, G. (1996). Employee timetabling, constraint networks and knowledge-based rules: a mixed approach, *Practice and Theory of Automated Timetabling, First International Conference, Edinburgh*, Vol. 1153, Springer, pp. 93–105.

Meisels, A. & Lusternik, N. (1998). Experiments in networks of employee timetableing problems, *Practice and Theory of Automated Timetabling, Second International Conference, Toronto*, Vol. 1408, Springer, pp. 130–141.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. (1958). Equations of state calculations by fast computing machines, *Journal of Chemical Physics* **21**: 1087–1092.

Meyer auf'm Hofe, H. (1997). ConPlan/SIEDAplan: Personnel assignment as a problem of hierarchical constraint satisfaction, *Proceedings on the 3rd International Conference on Practical Applications of Constraint Technologies, London*, pp. 257–272.

Meyer auf'm Hofe, H. (2001). Solving rostering tasks as constraint optimization, *Selected Papers from the 3rd international conference on Practice and Theory of Automated Timetabling (PATAT-2000), Konstanz*, Vol. 2079, Springer Verlag, pp. 191–212.

Millar, H. H. & Kiragu, M. (1998). Cyclic and non-cyclic scheduling of 12 h shift nurses by network programming, *European Journal of Operational Research* **104**: 582–592.

Miller, H. E., Pierskalla, W. P. & Rath, G. J. (1976). Nurse scheduling using mathematical programming, *Operations Research* **24**: 857–870.

Moz, M. & Pato, M. V. (2003). An integer multicommodity flow model applied to the rerostering of nurse schedules, *Annals of Operations Research* **17**: 285–301.

Mueller, C. W. & McCloskey, J. C. (1990). Nurses' job satisfaction: A proposed measure, *Nursing Research* **39**: 113–117.

Mullen, P. M. (2003). Prioritising waiting lists: How and why?, *European Journal of Operational Research* **150**: 32–45.

Musliu, N., Gärtner, J. & Slany, W. (2000). Efficient generation of rotating workforce schedules, *Proceedings of the 3rd international conference on the practice and theory of automated timetabling (PATAT 2000), Konstanz*, pp. 314–332.

Nemhauser, G. L. & Wolsey, L. A. (1999). *Integer and Combinatorial Optimization*, John Wiley & Sons, New York.

Nilsson, N. J. (1980). *Principles of Artificial Intelligence*, Morgan Kaufmann, San Mateo, California.

Nutt, P. C. (1984). Decision-modeling methods used to design decision support systems for staffing, *Medical Care* **22**(11): 1002–1013.

OECD (2005). OECD health data 2005: Statistics and indicators for 30 countries. *www.oecd.org

Okada, M. (1992). An approach to the generalized nurse scheduling problem–generation of a declarative program to represent institution-specific knowledge, *Computers and Biomedical Research* **25**(5): 417–434.

Okada, M. & Okada, M. (1988). Prolog-based system for nursing staff scheduling implemented on a personal computer, *Computers and Biomedical Research* **21**(1): 53–63.

Oldenkamp, J. H. (1992). Investigating reasoning in maternity care scheduling, *Knowledge and Policy* **5**: 67–76.

Ozkarahan, I. (1989). Flexible nurse scheduling support systems, *Computer Methods and Programs in Biomedicine* **30**: 145–153.

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

Ozkarahan, I. (2000). Allocation of surgeries to operating rooms using goal programming, *Journal of Medical Systems* **24**(6): 339–378.

Ozkarahan, I. & Bailey, J. (1988). Goal programming model subsystem of a flexible nurse scheduling support system, *IIE Transactions* **20**(3): 306–316.

Peeters, M. (2002). *One dimensional cutting and packing: New problems and algorithms*, Ph.D. dissertation, Faculteit Economische en Toegepaste Economische Wetenschappen, Katholieke Universiteit Leuven, Belgium.

Petrovic, S., Beddoe, G. & Vanden Berghe, G. (2003). Storing and adapting repair experiences in personnel rostering, *Practice and Theory of Automated Timetabling, Fourth International Conference, Gent*, Vol. 2740, Springer, pp. 185–186.

Pincus, M. (1970). A monte carlo method for the approximate solution of certain types of constrained optimization problems, *Operations Research* **18**: 1225–1228.

Rifai, A. K. & Pecenka, J. O. (1989). An application of goal programming in healthcare planning, *International Journal of Production Management* **10**: 28–37.

RIZIV (2005). Evolutie van de uitgaven voor geneeskundige verzorging. *www.riziv.be

Robbins, W. A. & Tuntiwongbiboon, N. (1989). Linear programming is a useful tool in case-mix management, *Healthcare Financial Management* **43**: 114–116.

Rosenbloom, E. S. & Goertzen, N. F. (1987). Cyclic nurse scheduling, *European Journal of Operational Research* **31**: 19–23.

Ross, S. M. (1983). *Introduction to Stochastic Dynamic Programming*, Academic Press, New York, NY.

Ryan, D. M. & Foster, B. A. (1981). An integer programming approach to scheduling, *in* A. Weren (ed.), *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, North-Holland, Amsterdam, pp. 269–280.

Santibanez, P., Begen, M. & Atkins, D. (2005). Managing surgical waitlists for a British Columbia health authority, *Research report*, Centre for Operations Excellence, Sauder School of Business, University of British Columbia, Canada.

Schaerf, A. & Meisels, A. (2000). Solving employee timetabling problems by generalized local search, *AI*IA '99: Proceedings of the 6th Congress of the Italian Association for Artificial Intelligence on Advances in Artificial Intelligence*, Springer-Verlag, London, UK, pp. 380–389.

Scott, S. & Simpson, R. M. (1998). Case-bases incorporating scheduling constraint dimensions: Experiences in nurse rostering, *Advances in Case-Based Reasoning*, Vol. 1488, Springer, pp. 392–401.

Sedgewick, R. (1998). *Algorithms in C++*, Addisson-Wesley Publishing Company, Inc.

Sennott, L. I. (1999). *Stochastic Dynamic Programming and the Control of Queueing Systems*, John Wiley & Sons, New York, NY.

Sermeus, W., Vanhaecht, K. & Vleugels, A. (2001). The Belgian-Dutch clinical pathway network, *Journal of Integrated Care Pathways* **5**: 10–14.

Siferd, S. P. & Benton, W. C. (1992). Workforce staffing and scheduling: Hospital nursing specific models, *European Journal of Operational Research* **60**: 233–246.

Silvestro, R. & Silvestro, C. (2000). An evaluation of nurse rostering practices in the national health service, *Journal of Advanced Nursing* **32**(3): 525–535.

Sitompul, D. & Randhawa, S. (1990). Nurse scheduling models: A state-of-the-art review, *Journal of the Society of Health Systems* **2**(1): 62–72.

Slowinski, R. & Hapke, M. (eds) (2000). *Scheduling Under Fuzziness*, Vol. 37 of *Studies in Fuzziness and Soft Computing*, Springer-Verlag.

Slowinski, R. & Teghem, J. (eds) (1990). *Stochastic versus Fuzzy Approaches to Multiobjective Mathematical Programming under Uncertainty*, Kluwer Academic Publishers, Dordrecht.

Smith, L. D. (1976). The application of an interactive algorithm to develop cyclical rotational schedulies for nursing personnel, *Information Systems and Operational Research* **14**: 53–70.

Smith, L. D., Bird, D. & Wiggins, A. (1979). A computerised system to schedule nurses that recognises staff preferences, *Hospital & Health Service Administration* **24**: 19–35.

Smith, L. D. & Wiggins, A. (1977). A computerbased nurse scheduling system, *Computers and Operations Research* **4**(3): 195–212.

Standridge, C. R. (1999). A tutorial on simulation in health care: Applications and issues, *WSC '99: Proceedings of the 31st conference on Winter simulation*, ACM Press, New York, NY, USA, pp. 49–55.

Strum, D. P., Vargas, L. G. & May, J. H. (1997). Resource coordination systems for surgical services using distributed communications, *Journal of the American Medical Informatics Association* **4**: 125–135.

Swisher, J. R., Jacobson, S. H., Jun, J. B. & Balci, O. (2001). Modeling and analyzing a physician clinic environment using discrete-event (visual) simulation, *Computers and Operations Research* **28**: 105–125.

Tanomaru, J. (1995). Staff scheduling by a genetic algorithm with heuristic operators, *Proceedings of the IEEE Conference on Evolutionary Computation, New York*, pp. 456–461.

Tien, J. & Kamiyama, A. (1982). On manpower scheduling algorithms, *Society for Industrial and Applied Mathematics* **24**: 275–287.

Trivedi, V. M. & Warner, D. M. (1976). A branch and bound algorithm for optimum allocation of float nurses, *Management Science* **22**(9): 972–981.

Udpa, S. (1996). Activity-based costing for hospitals, *Health Care Management Review* **21**(3): 83–96.

USDHHS (2002). *Projected supply, demand and shortages of registered nurses: 2000-2020*, National Center for Health Workforce Analysis. US Department of health and Human Services, Rockville, MD.

Valouxis, C. & Housos, E. (2000). Hybrid optimization techniques for the workshift and rest assignment of nursing personnel, *Artificial Intelligence in Medicine* **20**(2): 155–175.

Van den Akker, M., Hoogeveen, H. & van de Velde, S. L. (2002). Combining column generation and lagrangian relaxation to solve a single-machine common due date problem, *INFORMS Journal on Computing* **14**: 37–51.

Van Laarhoven, P. J. M. & Aarts, E. H. L. (1988). *Simulated annealing: Theory and applications*, kluwer, Dordrecht.

Vance, P. H., Barnhart, C., Johnson, E. L. & Nemhauser, G. L. (1997). Airline crew scheduling: A new formulation and decomposition algorithm, *Operations Research* **45**: 188–200.

Vanderbeck, F. (2000). On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm, *Operations Research* **48**: 111–128.

Vanderbeck, F. & Wolsey, L. A. (1996). An exact algorithm for IP column generation, *Operations Research Letters* **19**: 151–159.

Vanhaecht, K. & Sermeus, W. (2003). The Leuven clinical pathway compass, *Journal of Integrated Care Pathways* **7**: 2–7.

Vanhaecht, K., Sermeus, W., Vleugels, A. & Peeters, G. (2002). Ontwikkeling en gebruik van klinische paden (clinical pathways) in de gezondheidszorg, *Tijdschrift voor Geneeskunde* **58**(23): 1442–1451.

Vanherck, P., Vanhaecht, K. & Sermeus, W. (2004). Effects of clinical pathways: Do they work, *Journal of Integrated Care Pathways* **8**: 95–105.

Vanhoucke, M. (2001). *Exact Algorithms for various Types of Project Scheduling Problems. Nonregular Objectives and time/cost Trade-offs.*, Ph.D. dissertation, Faculteit Economische en Toegepaste Economische Wetenschappen, Katholieke Universiteit Leuven, Belgium.

Venkataraman, R. & Brusco, M. J. (1996). An integrated analysis of nurse staffing and scheduling policies, *Omega* **24**: 57–71.

Vissers, J. M. H., Bertrand, J. & de Vries, G. (2001). A framework for production control in healthcare organisations, *Production Planning and Control* **12**(6): 591–604.

Vissers, J. M., Van Der Bij, J. D. & Kusters, R. J. (2001). Towards decision support for waiting lists: An operations management view, *Health Care Management Science* **4**: 133–142.

Warner, D. M. (1976a). Nurse staffing, scheduling, and reallocation in the hospital, *Hospital & Health Services Administration* **21**: 77–90.

Warner, D. M. (1976b). Scheduling nursing personnel according to nursing preferences: A mathematical programming approach, *Operations Research* **24**: 842–856.

Warner, D. M., Keller, B. & Martel, S. (1991). Automated nurse scheduling, *Journal of the Society of Health Systems* **2**(2): 66–80.

Warner, D. M. & Prawda, J. (1972). A mathematical programming model for scheduling nursing personnel in a hospital, *Management Science* **19**: 411–422.

Weil, G., Heus, K., Francois, P. & Poujade, M. (1995). Constraint programming for nurse scheduling, *Engineering in Medicine and Biology Magazine, IEEE* **14**: 417–422.

Weiss, E. N. (1990). Models for determining estimated start times and case orderings in hospital operating rooms, *IIE Transactions* **22**: 143–150.

Wiers, V. C. (1997). A review of the applicability of or and ai scheduling techniques in practice, *Omega - the International Journal of Management Science* **25**(2): 145–153.

Wilkinson, R. & Allison, S. (1989). Alertness of night nurses: Two shift systems compared, *Ergonomics* **32**: 281–292.

Williams, H. P. (1999). *Model Building in Mathematical Programming*, John Wiley & Sons, New York.

Winston, W. L. (1993). *Operations research: Applications and algorithms*, Duxbury Press, Belmont, CA.

Wolsey, L. A. (1998). *Integer Programming*, John Wiley & Sons, Chichester.

Wright, M. B. (1987). The application of a surgical bed simulation model, *European Journal of Operational Research* **32**: 26–32.

Doctoral Dissertations from the Faculty of Economic and Applied Economic Sciences

From August 1, 1971.

1. GEPTS Stefaan (1971)
   Stability and efficiency of resource allocation processes in discrete commodity spaces. Leuven, KUL, 1971. 86 pp.

2. PEETERS Theo (1971)
   Determinanten van de internationale handel in fabrikaten. Leuven, Acco, 1971. 290 pp.

3. VAN LOOY Wim (1971)
   Personeelsopleiding: een onderzoek naar investeringsaspekten van opleiding. Hasselt, Vereniging voor wetenschappelijk onderzoek in Limburg, 1971. VII, 238 pp.

4. THARAKAN Mathew (1972)
   Indian exports to the European community: problems and prospects. Leuven, Faculty of economics and applied economics, 1972. X,343 pp.

5. HERROELEN Willy (1972)
   Heuristische programmatie: methodologische benadering en praktische toepassing op complexe combinatorische problemen. Leuven, Aurelia scientifica, 1972. X, 367 pp.

6. VANDENBULCKE Jacques (1973)
   De studie en de evaluatie van data-organisatiemethodes en data-zoekmethodes. Leuven, s.n., 1973. 3 V.

7. PENNYCUICK Roy A. (1973)
   The economics of the ecological syndrome. Leuven, Acco, 1973. XII, 177 pp.

8. KAWATA T. Bualum (1973)
Formation du capital d'origine belge, dette publique et stratégie du développement au Zaire. Leuven, KUL, 1973. V, 342 pp.

9. DONCKELS Rik (1974)
Doelmatige oriëntering van de sectorale subsidiepolitiek in België: een theoretisch onderzoek met empirische toetsing. Leuven, K.U.Leuven, 1974. VII, 156 pp.

10. VERHELST Maurice (1974)
Contribution to the analysis of organizational information systems and their financial benefits. Leuven, K.U.Leuven, 1974. 2 V.

11. CLEMEUR Hugo (1974)
Enkele verzekeringstechnische vraagstukken in het licht van de nutstheorie. Leuven, Aurelia scientifica, 1974. 193 pp.

12. HEYVAERT Edward (1975)
De ontwikkeling van de moderne bank- en krediettechniek tijdens de zestiende en zeventiende eeuw in Europa en te Amsterdam in het bijzonder. Leuven, K.U.Leuven, 1975. 186 pp.

13. VERTONGHEN Robert (1975)
Investeringscriteria voor publieke investeringen: het uitwerken van een operationele theorie met een toepassing op de verkeersinfrastructuur. Leuven, Acco, 1975. 254 pp.

14. Niet toegekend.

15. VANOVERBEKE Lieven (1975)
Microeconomisch onderzoek van de sectoriële arbeidsmobiliteit. Leuven, Acco, 1975. 205 pp.

16. DAEMS Herman (1975)
The holding company: essays on financial intermediation, concentration and capital market imperfections in the Belgian economy. Leuven, K.U.Leuven, 1975. XII, 268 pp.

17. VAN ROMPUY Eric (1975)
Groot-Brittannië en de Europese monetaire integratie: een onderzoek naar de gevolgen van de Britse toetreding op de geplande Europese monetaire unie. Leuven, Acco, 1975. XIII, 222 pp.

18. MOESEN Wim (1975)
Het beheer van de staatsschuld en de termijnstructuur van de intrestvoeten met een toepassing voor België. Leuven, Vander, 1975. XVI, 250 pp.

19. LAMBRECHT Marc (1976)
Capacity constrained multi-facility dynamic lot-size problem. Leuven, KUL, 1976. 165 pp.

20. RAYMAECKERS Erik (1976)
De mens in de onderneming en de theorie van het producenten-gedrag: een bijdrage tot transdisciplinaire analyse. Leuven, Acco, 1976. XIII, 538 pp.

21. TEJANO Albert (1976)
Econometric and input-output models in development planning: the case of the Philippines. Leuven, KUL, 1976. XX, 297 pp.

22. MARTENS Bernard (1977)
Prijsbeleid en inflatie met een toepassing op België. Leuven, KUL, 1977. IV, 253 pp.

23. VERHEIRSTRAETEN Albert (1977)
Geld, krediet en intrest in de Belgische financiële sector. Leuven, Acco, 1977. XXII, 377 pp.

24. GHEYSSENS Lieven (1977)
International diversification through the government bond market: a risk-return analysis. Leuven, s.n., 1977. 188 pp.

25. LEFEBVRE Chris (1977)
Boekhoudkundige verwerking en financiële verslaggeving van huurkooptransacties en verkopen op afbetaling bij ondernemingen die aan consumenten verkopen. Leuven, KUL, 1977. 228 pp.

26. KESENNE Stefan (1978)
Tijdsallocatie en vrijetijdsbesteding: een econometrisch onderzoek. Leuven, s.n., 1978. 163 pp.

27. VAN HERCK Gustaaf (1978)
Aspecten van optimaal bedrijfsbeleid volgens het marktwaardecriterium: een risico-rendementsanalyse. Leuven, KUL, 1978. IV, 163 pp.

28. VAN POECK Andre (1979)
World price trends and price and wage development in Belgium: an investigation into the relevance of the Scandinavian model of inflation for Belgium. Leuven, s.n., 1979. XIV, 260 pp.

29. VOS Herman (1978)
De industriële technologieverwerving in Brazilië: een analyse. Leuven, s.n., 1978. onregelmatig gepagineerd.

30. DOMBRECHT Michel (1979)
Financial markets, employment and prices in open economies. Leuven, KUL, 1979. 182 pp.

31. DE PRIL Nelson (1979)
Bijdrage tot de actuariële studie van het bonus-malussysteem. Brussel, OAB, 1979. 112 pp.

32. CARRIN Guy (1979)
Economic aspects of social security: a public economics approach. Leuven, KUL, 1979. onregelmatig gepagineerd

33. REGIDOR Baldomero (1979)
An empirical investigation of the distribution of stock-market prices and weak-form efficiency of the Brussels stock exchange. Leuven, KUL, 1979. 214 pp.

34. DE GROOT Roger (1979)
Ongelijkheden voor stop loss premies gebaseerd op E.T. systemen in het kader van de veralgemeende convexe analyse. Leuven, KUL, 1979. 155 pp.

35. CEYSSENS Martin (1979)
On the peak load problem in the presence of rationizing by waiting. Leuven, KUL, 1979. IX, 217 pp.

36. ABDUL RAZK Abdul (1979)
Mixed enterprise in Malaysia: the case study of joint venture between Malysian public corporations and foreign enterprises. Leuven, KUL, 1979. 324 pp.

37. DE BRUYNE Guido (1980)
Coordination of economic policy: a game-theoretic approach. Leuven, KUL, 1980. 106 pp.

38. KELLES Gerard (1980)
Demand, supply, price change and trading volume on financial markets of the matching-order type. = Vraag, aanbod, koersontwikkeling en omzet op financiële markten van het Europese type. Leuven, KUL, 1980. 222 pp.

39. VAN EECKHOUDT Marc (1980)
De invloed van de looptijd, de coupon en de verwachte inflatie op het opbrengstverloop van vastrentende finaciële activa. Leuven, KUL, 1980. 294 pp.

40. SERCU Piet (1981)
Mean-variance asset pricing with deviations from purchasing power parity. Leuven, s.n., 1981. XIV, 273 pp.

41. DEQUAE Marie-Gemma (1981)
Inflatie, belastingsysteem en waarde van de onderneming. Leuven, KUL, 1981. 436 pp.

42. BRENNAN John (1982)
An empirical investigation of Belgian price regulation by prior notification: 1975 - 1979 - 1982. Leuven, KUL, 1982. XIII, 386 pp.

43. COLLA Annie (1982)
Een econometrische analyse van ziekenhuiszorgen. Leuven, KUL, 1982. 319 pp.

44. Niet toegekend.

45. SCHOKKAERT Eric (1982)
Modelling consumer preference formation. Leuven, KUL, 1982. VIII, 287 pp.

46. DEGADT Jan (1982)
Specificatie van een econometrisch model voor vervuilingsproblemen met proeven van toepassing op de waterverontreiniging in België. Leuven, s.n., 1982. 2 V.

47. LANJONG Mohammad Nasir (1983)
A study of market efficiency and risk-return relationships in the Malaysian capital market. s.l., s.n., 1983. XVI, 287 pp.

48. PROOST Stef (1983)
De allocatie van lokale publieke goederen in een economie met een centrale overheid en lokale overheden. Leuven, s.n., 1983. onregelmatig gepagineerd.

49. VAN HULLE Cynthia (1983)
Shareholders' unanimity and optimal corporate decision making in imperfect capital markets. s.l., s.n., 1983. 147 pp. + appendix.

50. VAN WOUWE Martine (2/12/83)
Ordening van risico's met toepassing op de berekening van ultieme ruïnekansen. Leuven, s.n., 1983. 109 pp.

51. D'ALCANTARA Gonzague (15/12/83)
SERENA: a macroeconomic sectoral regional and national account econometric model for the Belgian economy. Leuven, KUL, 1983. 595 pp.

52. D'HAVE Piet (24/02/84)
De vraag naar geld in België. Leuven, KUL, 1984. XI, 318 pp.

53. MAES Ivo (16/03/84)
The contribution of J.R. Hicks to macro-economic and monetary theory. Leuven, KUL, 1984. V, 224 pp.

54. SUBIANTO Bambang (13/09/84)
A study of the effects of specific taxes and subsidies on a firms' R&D investment plan. s.l., s.n., 1984. V, 284 pp.

55. SLEUWAEGEN Leo (26/10/84)
Location and investment decisions by multinational enterprises in Belgium and Europe. Leuven, KUL, 1984. XII, 247 pp.

56. GEYSKENS Erik (27/03/85)
Produktietheorie en dualiteit. Leuven, s.n., 1985. VII, 392 pp.

57. COLE Frank (26/06/85)
Some algorithms for geometric programming. Leuven, KUL, 1985. 166 pp.

58. STANDAERT Stan (26/09/86)
A study in the economics of repressed consumption. Leuven, KUL, 1986. X, 380 pp.

59. DELBEKE Jos (03/11/86)
Trendperioden in de geldhoeveelheid van België 1877-1983: een theoretische en empirische analyse van de "Banking school"hypothese. Leuven, KUL, 1986. XII, 430 pp.

60. VANTHIENEN Jan (08/12/86)
Automatiseringsaspecten van de specificatie, constructie en manipulatie van beslissingstabellen. Leuven, s.n., 1986. XIV, 378 pp.

61. LUYTEN Robert (30/04/87)
A systems-based approach for multi-echelon production/inventory systems. s.l., s.n., 1987. 3V.

62. MERCKEN Roger (27/04/87)
De invloed van de data base benadering op de interne controle. Leuven, s.n., 1987. XIII, 346 pp.

63. VAN CAYSEELE Patrick (20/05/87)
Regulation and international innovative activities in the pharmaceutical industry. s.l., s.n., 1987. XI, 169 pp.

64. FRANCOIS Pierre (21/09/87)
De empirische relevantie van de independence from irrelevant alternatives. Assumptie indiscrete keuzemodellen. Leuven, s.n., 1987. IX, 379 pp.

65. DECOSTER André (23/09/88)
Family size, welfare and public policy. Leuven, KUL. Faculteit Economische en Toegepaste Economische Wetenschappen, 1988. XIII, 444 pp.

66. HEIJNEN Bart (09/09/88)
Risicowijziging onder invloed van vrijstellingen en herverzekeringen: een theoretische analyse van optimaliteit en premiebepaling. Leuven, KUL. Faculteit Economische en Toegepaste Economische Wetenschappen, 1988. onregelmatig gepagineerd.

67. GEEROMS Hans (14/10/88)
Belastingvermijding. Theoretische analyse van de determinanten van de belastingontduiking en de belastingontwijking met empirische verificaties. Leuven, s.n., 1988. XIII, 409, 5 pp.

68. PUT Ferdi (19/12/88)
Introducing dynamic and temporal aspects in a conceptual (database) schema. Leuven, KUL. Faculteit Economische en Toegepaste Economische Wetenschappen, 1988. XVIII, 415 pp.

69. VAN ROMPUY Guido (13/01/89)
A supply-side approach to tax reform programs. Theory and empirical evidence for Belgium. Leuven, KUL. Faculteit Economische en Toegepaste Economische Wetenschappen, 1989. XVI, 189, 6 pp.

70. PEETERS Ludo (19/06/89)
Een ruimtelijk evenwichtsmodel van de graanmarkten in de E.G.: empirische specificatie en beleidstoepassingen. Leuven, K.U.Leuven. Faculteit Economische en Toegepaste Economische Wetenschappen, 1989. XVI, 412 pp.

71. PACOLET Jozef (10/11/89)
Marktstructuur en operationele efficiëntie in de Belgische financiële sector. Leuven, K.U.Leuven. Faculteit Economische en Toegepaste Economische Wetenschappen, 1989. XXII, 547 pp.

72. VANDEBROEK Martina (13/12/89)
Optimalisatie van verzekeringscontracten en premieberekeningsprincipes. Leuven, K.U.Leuven. Faculteit Economische en Toegepaste Economische Wetenschappen, 1989. 95 pp.

73. WILLEKENS Francois (1990)
Determinance of government growth in industrialized countries with applications to Belgium. Leuven, K.U.Leuven. Faculteit Economische en Toegepaste Economische Wetenschappen, 1990. VI, 332 pp.

74. VEUGELERS Reinhilde (02/04/90)
Scope decisions of multinational enterprises. Leuven, K.U.Leuven. Faculteit Economische en Toegepaste Economische Wetenschappen, 1990. V, 221 pp.

75. KESTELOOT Katrien (18/06/90)
Essays on performance diagnosis and tacit cooperation in international oligopolies. Leuven, K.U.Leuven. Faculteit Economische en Toegepaste Economische Wetenschappen, 1990. 227 pp.

76. WU Changqi (23/10/90) Strategic aspects of oligopolistic vertical integration. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1990. VIII, 222 pp.

77. ZHANG Zhaoyong (08/07/91)
A disequilibrium model of China's foreign trade behaviour. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1991. XII, 256 pp.

78. DHAENE Jan (25/11/91)
Verdelingsfuncties, benaderingen en foutengrenzen van stochastische grootheden geassocieerd aan verzekeringspolissen en -portefeuilles. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1991. 146 pp.

79. BAUWELINCKX Thierry (07/01/92)
Hierarchical credibility techniques. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1992. 130 pp.

80. DEMEULEMEESTER Erik (23/3/92)
Optimal algorithms for various classes of multiple resource-constrained project scheduling problems. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1992. 180 pp.

81. STEENACKERS Anna (1/10/92)
Risk analysis with the classical actuarial risk model: theoretical extensions and applications to Reinsurance. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1992. 139 pp.

82. COCKX Bart (24/09/92)
The minimum income guarantee. Some views from a dynamic perspective. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1992. XVII, 401 pp.

83. MEYERMANS Eric (06/11/92)
Econometric allocation systems for the foreign exchange market: Specification, estimation and testing of transmission mechanisms under currency substitution. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1992. XVIII, 343 pp.

84. CHEN Guoqing (04/12/92)
Design of fuzzy relational databases based on fuzzy functional dependency. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1992. 176 pp.

85. CLAEYS Christel (18/02/93)
Vertical and horizontal category structures in consumer decision making: The nature of product hierarchies and the effect of brand typicality. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1993. 348 pp.

86. CHEN Shaoxiang (25/03/93)
The optimal monitoring policies for some stochastic and dynamic production processes. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1993. 170 pp.

87. OVERWEG Dirk (23/04/93)
Approximate parametric analysis and study of cost capacity management of computer configurations. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1993. 270 pp.

88. DEWACHTER Hans (22/06/93)
Nonlinearities in speculative prices: The existence and persistence of nonlinearity in foreign exchange rates. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1993. 151 pp.

89. LIN Liangqi (05/07/93)
Economic determinants of voluntary accounting choices for R & D expenditures in Belgium.

Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1993. 192 pp.

90. DHAENE Geert (09/07/93)
Encompassing: formulation, properties and testing. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1993. 117 pp.

91. LAGAE Wim (20/09/93)
Marktconforme verlichting van soevereine buitenlandse schuld door private crediteuren: een neo-institutionele analyse. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1993. 241 pp.

92. VAN DE GAER Dirk (27/09/93)
Equality of opportunity and investment in human capital. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1993. 172 pp.

93. SCHROYEN Alfred (28/02/94)
Essays on redistributive taxation when monitoring is costly. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1994. 203 pp. + V.

94. STEURS Geert (15/07/94)
Spillovers and cooperation in research and development. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1994. 266 pp.

95. BARAS Johan (15/09/94)
The small sample distribution of the Wald, Lagrange multiplier and likelihood ratio tests for homogeneity and symmetry in demand analysis: a Monte Carlo study. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1994. 169 pp.

96. GAEREMYNCK Ann (08/09/94)
The use of depreciation in accounting as a signalling device. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1994. 232 pp.

97. BETTENDORF Leon (22/09/94)
A dynamic applied general equilibrium model for a small open economy. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1994. 149 pp.

98. TEUNEN Marleen (10/11/94)
Evaluation of interest randomness in actuarial quantities. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1994. 214 pp.

99. VAN OOTEGEM Luc (17/01/95)
An economic theory of private donations. Leuven. K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1995. 236 pp.

100. DE SCHEPPER Ann (20/03/95)
Stochastic interest rates and the probabilistic behaviour of actuarial functions. Leuven,

K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1995. 211 pp.

101. LAUWERS Luc (13/06/95)
Social choice with infinite populations. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1995. 79 pp.

102. WU Guang (27/06/95)
A systematic approach to object-oriented business modeling. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1995. 248 pp.

103. WU Xueping (21/08/95)
Term structures in the Belgian market: model estimation and pricing error analysis. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1995. 133 pp.

104. PEPERMANS Guido (30/08/95)
Four essays on retirement from the labor force. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1995. 128 pp.

105. ALGOED Koen (11/09/95)
Essays on insurance: a view from a dynamic perspective. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1995. 136 pp.

106. DEGRYSE Hans (10/10/95)
Essays on financial intermediation, product differentiation, and market structure. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1995. 218 pp.

107. MEIR Jos (05/12/95)
Het strategisch groepsconcept toegepast op de Belgische financiële sector. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1995. 257 pp.

108. WIJAYA Miryam Lilian (08/01/96)
Voluntary reciprocity as an informal social insurance mechanism: a game theoretic approach. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1996. 124 pp.

109. VANDAELE Nico (12/02/96)
The impact of lot sizing on queueing delays: multi product, multi machine models. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1996. 243 pp.

110. GIELENS Geert (27/02/96)
Some essays on discrete time target zones and their tails. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1996. 131 pp.

111. GUILLAUME Dominique (20/03/96)
Chaos, randomness and order in the foreign exchange markets. Essays on the modelling of the markets. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1996. 171 pp.

112. DEWIT Gerda (03/06/96)
Essays on export insurance subsidization. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1996. 186 pp.

113. VAN DEN ACKER Carine (08/07/96)
Belief-function theory and its application to the modeling of uncertainty in financial statement auditing. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1996. 147 pp.

114. IMAM Mahmood Osman (31/07/96)
Choice of IPO Flotation Methods in Belgium in an Asymmetric Information Framework and Pricing of IPO's in the Long-Run. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1996. 221 pp.

115. NICAISE Ides (06/09/96)
Poverty and Human Capital. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1996. 209 pp.

116. EYCKMANS Johan (18/09/97)
On the Incentives of Nations to Join International Environmental Agreements. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1997. XV + 348 pp.

117. CRISOLOGO-MENDOZA Lorelei (16/10/97)
Essays on Decision Making in Rural Households: a study of three villages in the Cordillera Region of the Philippines. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1997. 256 pp.

118. DE REYCK Bert (26/01/98)
Scheduling Projects with Generalized Precedence Relations: Exact and Heuristic Procedures. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1998. XXIV + 337 pp.

119. VANDEMAELE Sigrid (30/04/98)
Determinants of Issue Procedure Choice within the Context of the French IPO Market: Analysis within an Asymmetric Information Framework. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1998. 241 pp.

120. VERGAUWEN Filip (30/04/98)
Firm Efficiency and Compensation Schemes for the Management of Innovative Activities and Knowledge Transfers. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1998. VIII + 175 pp.

121. LEEMANS Herlinde (29/05/98)
The Two-Class Two-Server Queueing Model with Nonpreemptive Heterogeneous Priority Structures. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1998. 211 pp.

122. GEYSKENS Inge (4/09/98)
Trust, Satisfaction, and Equity in Marketing Channel Relationships. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1998. 202 pp.

123. SWEENEY John (19/10/98)
Why Hold a Job ? The Labour Market Choice of the Low-Skilled. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1998. 278 pp.

124. GOEDHUYS Micheline (17/03/99)
Industrial Organisation in Developing Countries, Evidence from Côte d'Ivoire. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1999. 251 pp.

125. POELS Geert (16/04/99)
On the Formal Aspects of the Measurement of Object-Oriented Software Specifications. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1999. 507 pp.

126. MAYERES Inge (25/05/99)
The Control of Transport Externalities: A General Equilibrium Analysis. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1999. XIV + 294 pp.

127. LEMAHIEU Wilfried (5/07/99)
Improved Navigation and Maintenance through an Object-Oriented Approach to Hypermedia Modelling. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1999. 284 pp.

128. VAN PUYENBROECK Tom (8/07/99)
Informational Aspects of Fiscal Federalism. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1999. 192 pp.

129. VAN DEN POEL Dirk (5/08/99)
Response Modeling for Database Marketing Using Binary Classification. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1999. 342 pp.

130. GIELENS Katrijn (27/08/99)
International Entry Decisions in the Retailing Industry: Antecedents and Performance Consequences. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1999. 336 pp.

131. PEETERS Anneleen (16/12/99)
Labour Turnover Costs, Employment and Temporary Work. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1999. 207 pp.

132. VANHOENACKER Jurgen (17/12/99)
Formalizing a Knowledge Management Architecture Meta-Model for Integrated Business Process Management. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 1999. 252 pp.

133. NUNES Paulo (20/03/2000)
Contingent Valuation of the Benefits of Natural Areas and its Warmglow Component. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2000. XXI + 282 pp.

134. VAN DEN CRUYCE Bart (7/04/2000)
Statistische discriminatie van allochtonen op jobmarkten met rigide lonen. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2000. XXIII + 441 pp.

135. REPKINE Alexandre (15/03/2000)
Industrial restructuring in countries of Central and Eastern Europe: Combining branch-, firm- and product-level data for a better understanding of Enterprises' behaviour during transition towards market economy. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2000. VI + 147 pp.

136. AKSOY, Yunus (21/06/2000)
Essays on international price rigidities and exchange rates. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2000. IX + 236 pp.

137. RIYANTO, Yohanes Eko (22/06/2000)
Essays on the internal and external delegation of authority in firms. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2000. VIII + 280 pp.

138. HUYGHEBAERT, Nancy (20/12/2000)
The Capital Structure of Business Start-ups. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2000. VIII + 332 pp.

139. FRANCKX Laurent (22/01/2001)
Ambient Inspections and Commitment in Environmental Enforcement. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. VIII + 286 pp.

140. VANDILLE Guy (16/02/2001)
Essays on the Impact of Income Redistribution on Trade. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. VIII + 176 pp.

141. MARQUERING Wessel (27/04/2001)
Modeling and Forecasting Stock Market Returns and Volatility. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. V + 267 pp.

142. FAGGIO Giulia (07/05/2001)
Labor Market Adjustment and Enterprise Behavior in Transition. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. 150 pp.

143. GOOS Peter (30/05/2001)
The Optimal Design of Blocked and Split-plot experiments. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. X + 224 pp.

144. LABRO Eva (01/06/2001)
Total Cost of Ownership Supplier Selection based on Activity Based Costing and Mathematical Programming. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. 217 pp.

145. VANHOUCKE Mario (07/06/2001)
Exact Algorithms for various Types of Project Scheduling Problems. Nonregular Objectives and time/cost Trade-offs. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. 316 pp.

146. BILSEN Valentijn (28/08/2001)
Entrepreneurship and Private Sector Development in Central European Transition Countries. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. XVI + 188 pp.

147. NIJS Vincent (10/08/2001)
Essays on the dynamic Category-level Impact of Price promotions. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001.

148. CHERCHYE Laurens (24/09/2001)
Topics in Non-parametric Production and Efficiency Analysis. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. VII + 169 pp.

149. VAN DENDER Kurt (15/10/2001)
Aspects of Congestion Pricing for Urban Transport. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. VII + 203 pp.

150. CAPEAU Bart (26/10/2001)
In defence of the excess demand approach to poor peasants' economic behaviour. Theory and Empirics of non-recursive agricultural household modelling. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. XIII + 286 pp.

151. CALTHROP Edward (09/11/2001)
Essays in urban transport economics. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001.

152. VANDER BAUWHEDE Heidi (03/12/2001)
Earnings management in an Non-Anglo-Saxon environment. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2001. 408 pp.

153. DE BACKER Koenraad (22/01/2002)
Multinational firms and industry dynamics in host countries : the case of Belgium. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. VII + 165 pp.

154. BOUWEN Jan (08/02/2002)
Transactive memory in operational workgroups. Concept elaboration and case study. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. 319 pp. + appendix 102 pp.

155. VAN DEN BRANDE Inge (13/03/2002)
The psychological contract between employer and employee : a survey among Flemish employees. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. VIII + 470 pp.

156. VEESTRAETEN Dirk (19/04/2002)
Asset Price Dynamics under Announced Policy Switching. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. 176 pp.

157. PEETERS Marc (16/05/2002)
One Dimensional Cutting and Packing : New Problems and Algorithms. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. IX + 247 pp.

158. SKUDELNY Frauke (21/05/2002)
Essays on The Economic Consequences of the European Monetary Union. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002.

159. DE WEERDT Joachim (07/06/2002)
Social Networks, Transfers and Insurance in Developing countries. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. VI + 129 pp.

160. TACK Lieven (25/06/2002)
Optimal Run Orders in Design of Experiments. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. XXXI + 344 pp.

161. POELMANS Stephan (10/07/2002)
Making Workflow Systems work. An investigation into the Importance of Task-appropriation fit, End-user Support and other Technological Characteristics. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. 237 pp.

162. JANS Raf (26/09/2002)
Capacitated Lot Sizing Problems : New Applications, Formulations and Algorithms. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002.

163. VIAENE Stijn (25/10/2002)
Learning to Detect Fraud from enriched Insurance Claims Data (Context, Theory and Applications). Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. 315 pp.

164. AYALEW Tekabe (08/11/2002)
Inequality and Capital Investment in a Subsistence Economy.Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. V + 148 pp.

165. MUES Christophe (12/11/2002)
On the Use of Decision Tables and Diagrams in Knowledge Modeling and Verification. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. 222 pp.

166. BROCK Ellen (13/03/2003)
The Impact of International Trade on European Labour Markets. K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002.

167. VERMEULEN Frederic (29/11/2002)
Essays on the collective Approach to Household Labour Supply. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. XIV + 203 pp.

168. CLUDTS Stephan (11/12/2002)
Combining participation in decision-making with financial participation : theoretical and empirical perspectives. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2002. XIV + 247 pp.

169. WARZYNSKI Frederic (09/01/2003)
The dynamic effect of competition on price cost margins and innovation. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

170. VERWIMP Philip (14/01/2003)
Development and genocide in Rwanda ; a political economy analysis of peasants and power under the Habyarimana regime. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

171. BIGANO Andrea (25/02/2003)
Environmental regulation of the electricity sector in a European Market Framework. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003. XX + 310 pp.

172. MAES Konstantijn (24/03/2003)
Modeling the Term Structure of Interest Rates Across Countries. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003. V+246 pp.

173. VINAIMONT Tom (26/02/2003)
The performance of One- versus Two-Factor Models of the Term Structure of Interest Rates. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen. 2003.

174. OOGHE Erwin (15/04/2003)
Essays in multi-dimensional social choice. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003. VIII+108 pp.

175. FORRIER Anneleen (25/04/2003)
Temporary employment, employability and training. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

176. CARDINAELS Eddy (28/04/2003)
The role of cost system accuracy in managerial decision making. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003. 144 pp.

177. DE GOEIJ Peter (02/07/2003)
Modeling Time-Varying Volatility and Interest Rates. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003. VII+225 pp.

178. LEUS Roel (19/09/2003)
The generation of stable project plans. Complexity and exact algorithms. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

179. MARINHEIRO Carlos (23/09/2003)
EMU and fiscal stabilisation policy : the case of small countries. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

180. BAESSENS Bart (24/09/2003)
Developing intelligent systems for credit scoring using machine learning techniques. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

181. KOCZY Laszlo (18/09/2003)
Solution concepts and outsider behaviour in coalition formation games. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

182. ALTOMONTE Carlo (25/09/2003)
Essays on Foreign Direct Investment in transition countries : learning from the evidence. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

183. DRIES Liesbeth (10/11/2003)
Transition, Globalisation and Sectoral Restructuring: Theory and Evidence from the Polish Agri-Food Sector. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

184. DEVOOGHT Kurt (18/11/2003)
Essays On Responsibility-Sensitive Egalitarianism and the Measurement of Income Inequality. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

185. DELEERSNYDER Barbara (28/11/2003)
Marketing in Turbulent Times. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

186. ALI Daniel (19/12/2003)
Essays on Household Consumption and Production Decisions under Uncertainty in Rural Ethiopia. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2003.

187. WILLEMS Bert (14/01/2004)
Electricity networks and generation market power. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

188. JANSSENS Gust (30/01/2004)
Advanced Modelling of Conditional Volatility and Correlation in Financial Markets. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

189. THOEN Vincent (19/01/2004)
On the valuation and disclosure practices implemented by venture capital providers. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

190. MARTENS Jurgen (16/02/2004)
A fuzzy set and stochastic system theoretic technique to validate simulation models. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

191. ALTAVILLA Carlo (21/05/2004)
Monetary policy implementation and transmission mechanisms in the Euro area. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

192. DE BRUYNE Karolien (07/06/2004)
Essays in the location of economic activity. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

193. ADEM Jan (25/06/2004)
Mathematical programming approaches for the supervised classification problem. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

194. LEROUGE Davy (08/07/2004)
Predicting Product Preferences : the effect of internal and external cues. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

195. VANDENBROECK Katleen (16/07/2004)
Essays on output growth, social learning and land allocation in agriculture : micro-evidence from Ethiopia and Tanzania. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

196. GRIMALDI Maria (03/09/2004)
The exchange rate, heterogeneity of agents and bounded rationality. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

197. SMEDTS Kristien (26/10/2004)
Financial integration in EMU in the framework of the no-arbitrage theory. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

198. KOEVOETS Wim (12/11/2004)
Essays on Unions, Wages and Employment. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

199. CALLENS Marc (22/11/2004)
Essays on multilevel logistic Regression. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

200. RUGGOO Arvind (13/12/2004)
Two stage designs robust to model uncertainty. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2004.

201. HOORELBEKE Dirk (28/01/2005)
Bootstrap and Pivoting Techniques for Testing Multiple Hypotheses. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

202. ROUSSEAU Sandra (17/02/2005)
Selecting Environmental Policy Instruments in the Presence of Incomplete Compiance. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

203. VAN DER MEULEN Sofie (17/02/2005)
Quality of Financial Statements : Impact of the external auditor and applied accounting standards. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

204. DIMOVA Ralitza (21/02/2005)
Winners and Losers during Structural Reform and Crisis : the Bulgarian Labour Market Perspective. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

205. DARKIEWICZ Grzegorz (28/02/2005)
Value-at-risk in Insurance and Finance : the Comonotonicity Approach. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

206. DE MOOR Lieven (20/05/2005)
The Structure of International Stock Returns : Size, Country and Sector Effects in Capital Asset Pricing. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

207. EVERAERT Greetje (27/06/2005)
Soft Budget Constraints and Trade Policies : The Role of Institutional and External Constraints. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

208. SIMON Steven (06/07/2005)
The Modeling and Valuation of complex Derivatives : The Impact of the Choice of the term structure model. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

209. MOONEN Linda (23/09/2005)
Algorithms for some Graph-Theoretical Optimization Problems. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

210. COUCKE Kristien (21/09/2005)
Firm and industry adjustment under de-industrialisation and globalization of the Belgian economy. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

211. DECAMPS Marc (21/10/2005)
Some actuarial and financial applications of generalized diffusion processes. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

212. KIM Helena (29/11/2005)
Escalation games: an instrument to analyze conflicts. The strategic approach to the bargaining problem. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2005.

213. GERMENJI Etleva (06/01/2006)
Essays on the Economics of Emigration from Albania. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.

214. BELIEN Jeroen (18/01/2006)
Exact and Heuristic Methodologies for Scheduling in Hospitals: Problems, Formulations and Algorithms. Leuven, K.U.Leuven, Faculteit Economische en Toegepaste Economische Wetenschappen, 2006.