KATHOLIEKE UNIVERSITEIT
LEUVEN

Arenberg Doctoral School of Science, Engineering & Technology
Faculty of Science
Department of Computer Science

Negotiation at the short and mid term level supported by analysis of nurse rostering problems

Stefaan HASPESLAGH

Dissertation presented in partial
fulfillment of the requirements for
the degree of Doctor
in Science

September 2012

Negotiation at the short and mid term level supported by analysis of nurse rostering problems

Stefaan HASPESLAGH

Jury:
Prof. dr. P. Igodt, chair
Prof. dr. P. De Causmaecker, supervisor
Prof. dr. ir. T. Holvoet
Prof. dr. D. De Schreye
Prof. dr. ir. G. Vanden Berghe
Prof. dr. M.-A. Guerry
(Vrije Universiteit Brussel)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Science

September 2012

D/2012/10.705/74
ISBN 978-90-8649-557-3

Acknowledgements

For obtaining a PhD, a self-interested agent is required to complete a set of tasks. The agent is unable to complete its tasks without negotiating and cooperating with other agents in its environment. I would like to thank the people who have made it possible for the agent to successfully complete its tasks.

First of all, I would like to thank my promotor, prof. Patrick De Causmaecker. His experience in many different research domains, his guidelines, the numerous discussions providing new insights and his continuous support during my PhD were indispensable for the realisation of this work.

I would like to thank prof. Paul Igodt for introducing me to my promotor and giving me the chance to start a PhD studentship at KU Leuven KULAK.

I would like to thank prof. Greet Vanden Berghe, an expert in the field of nurse rostering, who was always willing to discuss my work. Her opinion and remarks improved the quality of this research project.

The members of my examination committee, prof. Igodt (chair), prof. De Schreye, prof. Holvoet, prof. Vanden Berghe and prof. Guerry, reviewed this work from many different angles. Their remarks and suggestions considerably improved the quality of this manuscript.

Finally, I would like to thank my family, especially my parents, Christel and Jef, who have always encouraged me during my education and in life, and my wife, Caroline, for their never-ending support. And last but not least I would like to thank my son, Siebe, for being the most important person in my life. Whenever things were not going as supposed, his smile is the source of my perseverance to reach my goals.

Abstract

The nurse rostering problem is an NP-hard combinatorial optimisation problem for the assignment of shifts to nurses. The rosters need to comply to an extensive set of constraints stemming from legal concerns, organisational concerns and personal preferences of the nurses. In most cases, the rostering is performed at the level of one ward in a hospital.

There is a wide set of different models for nurse rostering problems. The problems differ from country to country, from hospital to hospital and even wards within the same hospital may have different models. To improve the transferability of solution methods, designed for the different models, common and standardised formulations are necessary. We show how numberings, designed for a fast evaluation of nurse rostering problem solutions can be used to generate a formal, unambiguous representation of nurse rostering problem instances. The use of numberings allows for an efficient translation of instances to other problem domains. We give a proof of concept in the form of translation schemes to Satisfiability Problems and Mixed Integer Programs. The schemes can be used for the fast development of prototype solvers for new problems, or variations of known ones for which no specific solver is available.

As is commonly known, there is a gap between the nurse rostering problem models developed for research purposes and those used in practice. We organised the "First International Nurse Rostering Competition 2010". The main objectives of the competition were to generate new approaches to the associated problem by attracting researchers from different areas of research, to reduce the gap between research and practice in this important area of operations research and to further stimulate debate within the widening rostering research community.

Wards operate within a larger whole. At the individual level of a ward, (sudden) nurse shortages may arise. As a solution, wards in a hospital can cooperate to resolve shortages and to increase their level of efficiency. This kind of

interoperability is naturally modelled using agents. The autonomous agents are responsible for the optimisation of their rosters and communicate in order to tackle the aforementioned issues. In this thesis, we apply a number of state-of-the-art negotiation protocols and compare computation times, solution quality and communication times (network load) of the aforementioned approaches. We compare with a centralised approach.

The above stated problems are at the short term level of personnel management. The problems at the different personnel management levels, short, mid and long term, are interconnected and information from lower level problems should serve as input for the higher level decision problems. In this thesis we consider the mid term level problem of dividing workload (shifts) and personnel (nurses) among the different wards in a hospital. We model this problem as a multi-issue multi-agent negotiation problem with workload and personnel as issues. Information on the short term level problem is aggregated into operational performance level curves and serves as input for this negotiation problem. We study the problem in both an environment with complete and incomplete information. For the incomplete information setting, we design a negotiation protocol and study theoretical and qualitative properties of the protocol such as Pareto optimality, solution quality, convergence, ...

In the first part of the thesis, we elaborate on the nurse rostering problem under study. Besides an informal problem description, we present a formal representation using numberings and we elaborate on a mathematical model. We show how the use of numberings allows for an automated translation into Satisfiability Problems and Mixed Integer Programs. Next, we present the "First International Nurse Rostering Competition 2010". We elaborate on the organisation of the competition, its spirit and its rules, and we discuss the results. Negotiation is the main subject of the second part of the thesis. We study the application of five state-of-the-art negotiation protocols to the short term level problem of resolving nurse shortages and increasing the efficiency levels of wards in a hospital. Next, we perform an in-depth study of short term nurse rostering problem instances, leading to the introduction of the operational performance platforms of a ward. Finally, we present and discuss the multi-issue multi-agent negotiation protocol we developed for the division of workload and personnel among the wards in a hospital.

Beknopte samenvatting

Het personeelsplanningsprobleem in een ziekenhuis voor het opstellen van uurroosters voor verpleegkundig personeel is een NP-hard combinatorisch optimalisatieprobleem. Essentieel bestaat het probleem erin om het verpleegkundig personeel toe te wijzen aan tijdsloten, shifts genoemd. De uurroosters moeten voldoen aan een uitgebreide verzameling beperkingen. Beperkingen komen voort uit wettelijke voorschriften, uit specifieke belangen van het ziekenhuis en uit de persoonlijke voorkeuren van het personeel. Meestal wordt per afdeling van het ziekenhuis een uurrooster opgesteld.

Er is een grote diversiteit aan modellen voor personeelsplanning in een ziekenhuis. De modellen verschillen van land tot land, van hospitaal tot hospitaal en zelfs binnen hetzelfde hospitaal hanteren niet alle afdelingen hetzelfde model. Een oplossingsmethode is meestal ontworpen voor één specifiek model. Het is niet vanzelfsprekend om de oplossingsmethoden op verschillende modellen toe te passen. Een gemeenschappelijke, standaard voorstelling van personeelsplanningsproblemen is nodig om de overdraagbaarheid van methodes te faciliteren. We gebruiken nummeringen, ontworpen om efficiënt de kwaliteit van uurroosters te controleren, om problemen op een formele, eenduidige manier voor te stellen. Deze werkwijze laat toe om de problemen automatisch en efficiënt te vertalen naar andere probleemdomeinen. Concreet stellen we vertaalschema's op naar "Satisfiability" problemen (SAT) en "Mixed Integer" programma's (MIP). Dit laat ons toe om snel een prototype oplossingsmethode te ontwikkelen voor nieuwe problemen of voor varianten van gekende problemen waarvoor nog geen specifieke oplossingsmethode bestaat.

Zoals algemeen bekend, is er een kloof tussen de modellen gebruikt voor onderzoeksdoeleinden en modellen die in de praktijk toepasbaar zijn. De organisatie van de "First International Nurse Rostering Competitie 2010" is een poging om nieuwe oplossingsmethoden in het domein van personeelsplanning te vinden, om onderzoekers uit andere domeinen aan te sporen onderzoek te verrichten in dit domein en om de hierboven vernoemde kloof te verkleinen.

Afdelingen in een ziekenhuis zijn operationeel in een groter geheel. Op het niveau van een individuele afdeling kunnen er plotselinge personeelstekorten optreden. Als mogelijke oplossing kunnen afdelingen samenwerken om tekorten weg te werken en om een hoger efficiëntie-niveau na te streven. Dergelijke samenwerkingsverbanden worden vaak gemodelleerd met agenten. De autonome agenten in een systeem zijn elk verantwoordelijk voor het opstellen van hun eigen uurroosters. De agenten communiceren om de hiervoor opgesomde problemen aan te pakken. In deze thesis passen we een aantal "state-of-the-art" onderhandelingstechnieken toe. We bestuderen rekentijd, oplossingskwaliteit en communicatietijd. We vergelijken de technieken met een centrale oplossingsmethode.

De bovengenoemde problemen bevinden zich op het korte termijn niveau van personeelsplanning. We onderscheiden problemen op korte, middellange en lange termijn. De problemen zijn met elkaar verbonden en informatie moet uitgewisseld worden tussen de verschillende niveaus. In deze thesis bestuderen we het middellange termijn probleem om werklast (shifts) en personeel te verdelen onder de verschillende afdelingen in een ziekenhuis. We beschouwen dit probleem als een "multi-issue multi-agent" onderhandelingsprobleem. De shifts en het personeel vormen het onderwerp van de onderhandeling. Informatie van het korte termijn niveau wordt samengevat in curves die het operationeel performantieniveau van een afdeling voorstellen. We bestuderen onderhandeling voor zowel omgevingen waar informatie verborgen gehouden wordt voor anderen als voor omgevingen waar alle informatie gedeeld is tussen de verschillende partijen. Voor het eerste geval ontwerpen we een onderhandelingsprotocol en bestuderen we zowel theoretische als kwalitatieve eigenschappen van het protocol, zoals Pareto-optimaliteit, oplossingskwaliteit, convergentie, ...

In het eerste deel van de thesis bespreken we uitvoerig het model van het beschouwde personeelsplanningsprobleem. Naast een informele omschrijving van het probleem presenteren we een formele voorstelling gebaseerd op nummeringen en stellen we een mathematisch model op. We stellen vertaalschema's naar SAT en MIP op. We beschrijven de organisatie en de resultaten van de "First International Nurse Rostering Competition 2010". Onderhandeling is het onderwerp van het tweede thesisdeel. We passen vijf "state-of-the-art" onderhandelingstechnieken toe die toelaten personeelstekorten aan te pakken en het efficiëntie-niveau van afdelingen te verhogen. Vervolgens leidt een uitvoerige studie van korte termijn personeelsproblemen tot het opstellen van curves om het operationeel performantieniveau van afdelingen voor te stellen. Tenslotte presenteren en evalueren we een onderhandelingsprotocol voor het "multi-issue multi-agent" onderhandelingsprobleem dat bestaat uit het verdelen van werklast en personeel tussen de afdelingen in een ziekenhuis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Objective

Worldwide, personnel shortages arise in the health care sector (Vandeurzen, 2010). Due to a higher life expectancy, health needs are increasing and becoming more complex. At the same time, a rising number of ageing health care workers sets on to retire, putting even more stress on the personnel requirements. Because the operation of the health care system highly influences social welfare, governments are willing to ease social regulations for jobs in the health care sector. Health care organisations are willing to pay higher wages than average and are offering an increased flexibility towards the work schedules to suit the living and family situation of their employees. With these long-term measures for addressing the imminent staff shortages, students are encouraged to consider a career in health care and current health care providers are encouraged to remain active for a longer period of time. In addition, more instantly actions are required to deploy current health care personnel more efficiently. The focus of this thesis is situated in the context of the latter problem, more specifically for the employment of nurses in a hospital.

De Causmaecker and Vanden Berghe (2012) distinguish three levels in personnel management in hospitals: short term, mid term and long term. The authors stress there is still a gap between research at the different levels. At the long term level, strategic options are decided such as the existence of a ward or the discontinuation of an activity. The main decisions to be taken at the mid term level are tactical such as tuning the allocation of staff to expected performance and productivity. Short term decision making starts from a given staffing

and demand to produce feasible rosters that meet both the demand and the constraints of the individual working rosters. The decision making problems at these three levels are interconnected. Information needs to be fed to higher levels. Only part of the information is relevant. Not all details at a lower level are of practical use at a higher level.

The short term level problem, called nurse rostering problem, is essentially the problem of assigning *shifts* to *nurses*, mostly situated at the level of one ward. It is a problem of practical relevance, known to be a difficult combinatorial optimisation problem that in most cases is NP-hard (Karp, 1972; Osogami and Imai, 2000). In reality, the rostering is mostly performed manually by a head nurse, who is responsible for the good operation of the ward. Rosters must respect legal regulations, need to meet the demand and have to satisfy as much as possible nurses' personal preferences. Those preferences reflect the flexibility mentioned above. From experience, head nurses know what quality rosters are, i.e. rosters acceptable for nurses. The construction of the rosters is a time consuming task. The last decades, the nurse rostering problem received ample attention in literature with a vast set of models and algorithms (Burke et al., 2004; De Causmaecker and Vanden Berghe, 2011). Many contributions focus on particular, often simplified problems, defined by local hospitals in different countries. Problem definitions even differ from ward to ward within the same hospital (Dias et al., 2003). The solution methods presented are often only suited for those specific problems. Hence, transferring solution methods from one particular problems is hard and sometimes impossible.

Schaerf and Di Gaspero (2007) identify analogous problems within the field of timetabling research. Because authors describe their specific problems and ad hoc algorithms to tackle the problems, readers are left alone to judge the quality of contributions. Two important research qualities that often lack attention are measurability (comparability) and reproducibility. Several measures have been proposed that can be taken to improve contributions within timetabling (and rostering) research. Some of these include designing common formulations, providing benchmarks and solution validators, organising online competitions to further stimulate research and to attract researchers from other fields, defining the data format (to represent publicly available instances), ... One aim of this thesis is to contribute in this area and to give an onset to further close the gap between research and practice (McCollum, 2007).

A second aim of this thesis is to study negotiation within the context of nurse rostering. Suddenly personnel shortages may occur. Nurses can become ill or the staffing at a ward may abruptly be reduced due to the unexpected resignation of nurses. Some hospitals try to tackle this problem by introducing a 'mobile equipe', a pool of floating nurses to replace the absent nurses. In hospitals without a mobile equipe, agency nurses may be hired or rerostering

takes place. In the latter case, a ward tries to build a new roster with minimal changes with respect to the original roster. Nurses do not want their entire roster to be disturbed because of the absence of a colleague, as this would harm the promised flexibility of their rosters. If none of the aforementioned possibilities offer a solution, wards call in the help of other wards and try to exchange personnel to cover the problematic shifts. In this thesis we study a negotiation based approach for handling such shortages. The same method, exchanging personnel, is also applicable on a larger scheduling horizon, not only in case of absenteeism, to increase the level of efficiency of the wards in a hospital.

Wards of a hospital function within larger wholes. The number and types of wards a hospital houses is decided at the long term level. Wards often specialise in specific medical disciplines or patient populations. The wards need to agree on the allocation of nurses (of which the staffing is managed by the long term level) ensuring the good operation of their tasks. By experience, head nurses have built a good view on the amount of work that needs to be done to ensure the good care of their patients and the good operation of their ward. This information is key input for the allocation problem at the mid term level. In this thesis we collect this type of information by an in depth study of a state-of-the-art nurse rostering algorithm at the short term level. The mid term level problem is regarded as a negotiation problem. We model the wards as self interested agents negotiating distribution of personnel and workload. Essentially, the negotiation problem is a multi-issue multi-agent negotiation problem with the personnel and the workload as issues. Approaching the problem through negotiation will allow individual wards to hide part of their internals. However, this will not turn out to be an essential element.

1.2 Contributions

As mentioned in the previous section, the purpose of this research project is twofold. A first focus is on closing the gap between the nurse rostering problem models developed for research purposes and those used in practice. A conscientious modelling of the problems is essential. Because of the proliferation of different models, a common, formal and standardised formulation of rostering problems is indispensable and facilitates the comparison of models and solution methods. More concretely, researchers are required to provide structured formulations of the problem under study in a language such as XML, a clear and unambiguous description of the evaluation function and implementations of this evaluation function in the form of online and offline evaluators. As mentioned before, the organisation of a competition, taking the aforementioned

considerations into account, is an ideal way for and contributed substantially in achieving the goal of closing the gap between research and practice. As an additional advantage of the proposed modelling methodology, nurse rostering problem instances can be automatically converted into problems of different problem domains.

In the second part of the thesis, the focus is on negotiation techniques for short term and mid term nurse rostering problems. Early experiments for the short term negotiation problem on well known benchmark instances found in literature (Brucker et al., 2010), exposed some problems when using those benchmarks within the context of negotiation. As a result of the research in the first part of the thesis, a new set of benchmark instances based on the original model is developed, appropriate for studying negotiation in the context of nurse rostering.

Concretely, the main contributions of this thesis are:

- the introduction of a new **benchmark set** of nurse rostering problem instances based on a set of real world constraints. The set serves as a test bed to perform a formal comparison of the performance of various algorithms. The instances are described using numberings as defined by (Burke et al., 2001). The numberings allow for a formal and unambiguous representation of nurse rostering problem constraints and instances.

- the organisation of the **First International Nurse Rostering Competition 2010 (INRC2010)**. The above mentioned benchmark set was the subject of INRC2010. Similar to ITC2002 and ITC2007 (McCollum et al., 2009), two timetabling competitions, the main objectives of INRC2010 were to generate new approaches to the associated problem by attracting researchers from different areas of research, to reduce the gap between research and practice in this important area of operations research and to further stimulate debate within the widening rostering (and timetabling) research community.

- the design of **translation schemes** for automatically expressing nurse rostering problem instances as **Satisfiability Problem (SAT)** and as **Mixed Integer Programs (MIP)**. The first schema was used for hardness analysis of nurse rostering problems (Bilgin et al., 2009) to calculate SAT features for predicting the value of the objective function and the calculation time. The latter schema was used to study the difficulty level of the instances used for INRC2010. As we will show, the use of numberings for representing instances allows for compact and efficient translation schemes.

- a study of state-of-the-art **negotiation protocols for automated personnel exchange** between the wards in a hospital. We implemented and compared a Contract Net Protocol approach (Smith, 1980), an extension to the Contract Net Protocol (Aknine et al., 2004), a Simultaneous Ascending Auction (Milgrom, 2000), a Vickrey Auction (Vickrey, 1961) and an Ascending Proxy Auction (Ausubel and Milgrom, 2004). In particular, we studied the speed, quality and network load of the aforementioned approaches. We also perform a comparison of the protocols with a central approach.
- the introduction of **operational performance levels** as a means to aggregate relevant information of short term level nurse rostering problems for use at the mid term level of personnel management. These operational performance levels are used to narrow the gap between short and mid term nurse rostering.
- the design of a **Pareto optimal multi-agent multi-issue negotiation protocol**. The protocol is applied for the division of personnel (nurses) and workload (shifts) among wards in a hospital. We consider both situations where all information is publicly known in the hospital and when wards hide details of their operation towards the other wards in the hospital.

In summary, Figure 1.1 positions the above contributions respectively at the short term and the mid term level of personnel management in hospitals. The figure also highlights the contributions for which negotiation techniques are proposed as solution methods.

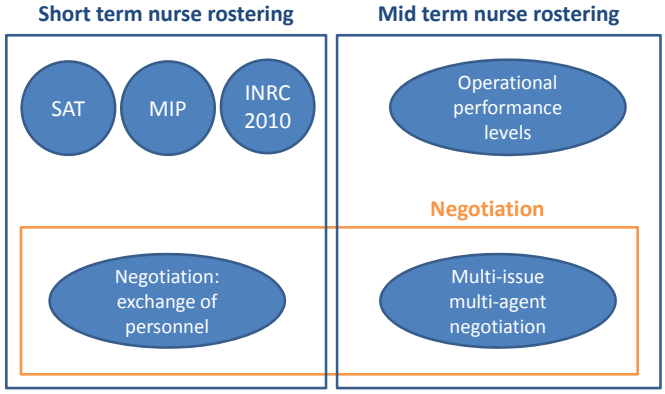

Figure 1.1: Overview of contributions

1.3 Structure of the Thesis

The structure of the thesis is as follows:

- The problem domain background is sketched by discussing literature on the modelling of nurse rostering problems, on state-of-the art negotiation protocols for the allocation of tasks and resources in a multi-agent setting and on multi-issue multi-agent negotiation protocols in Chapter 2.
- Chapter 3 presents the data model of the nurse rostering problem considered within this dissertation. We give a formal and unambiguous representation of the constraints using numberings (Burke et al., 2001).
- In Chapter 4, we present the translation schemes for automatically converting nurse rostering problem instances to Satisfiability Problems and Mixed Integer Programs.
- The "First International Nurse Rostering Competition 2010", its organisation and its results, is the subject of Chapter 5.
- We study the application of five state-of-the-art negotiation protocols for short term nurse rostering problems in Chapter 6.
- Chapter 7 elaborates on the mid term nurse rostering problem of dividing workload (shifts) and personnel (nurses) among the wards in a hospital.
- Conclusions and directions for further research are given in Chapter 8.

Chapter 2

Literature

2.1 The Nurse Rostering Problem

'Nurse rostering' is the generally used term for staff scheduling in hospitals. For an overview of an extensive set of models and algorithms we refer to the following surveys (Burke et al., 2004; De Causmaecker and Vanden Berghe, 2011; Burke et al., 2008). Many models (and data sets) are drawn from local hospitals and hence many algorithms are tailored specifically for solving those models. As Ernst et al. (2004) conclude for staff scheduling in general, there is a need for common and more generic problem formulations. This statement is further supported by Schaerf and Di Gaspero (2007). As stated in the introduction, to improve contributions within the timetabling and rostering research area these authors propose measures to tackle the following problems:

- *common formulations*: The advantage of using common formulations is twofold. It increases the interchangeability of approaches between different problems. It also allows to test an algorithm designed for a particular problem, on several publicly available benchmark sets. Problem formulations must be described clearly and exhaustively.
- *data format*: Using ad hoc, text-only data formats increases the chance for parsing errors and data format misinterpretations. Researchers need to put effort into the development of error-prone parsers. Using a structured format, such as XML, increases the flexibility, extensibility and maintainability of the data format. Often, well-tested parsers are publicly available. In order to maximise accessibility to a wide audience

of researchers, benchmark sets should be available in both a text-only and a structured data format. Best practice is to provide conversion tools to translate 'text-only instances' to 'structures instances' and vice versa.

- *comparison of methods*: It is hard to fairly compare different algorithms. Next to specifying what instances are used, authors need to report on the features (value of the objective function, success rate, speed, ...) used for the comparison and on the hardware on which the experiments have been performed. Special attention needs to be paid to stochastic algorithms. To handle the randomness, a proper statistical evaluation is essential.
- *result validation*: Due to bugs in the code or misunderstandings in the problem formulation, claimed results may turn out to be wrong. Solution validators need to be provided allowing researchers to confirm the correctness of obtained results.

We give an overview of nurse rostering related contributions, implementing measures to some of the aforementioned problems.

Petrovic and Vanden Berghe (2012) propose seven criteria for the comparison of different approaches: expressive power, flexibility, algorithmic power, learning capabilities, maintenance, rescheduling capacities and parameter tuning. As an example, a case-based reasoning and meta-heuristic approach are compared against those criteria.

Brucker et al. (2010) provide a benchmark set of 11 problem instances collected from small departments at real hospitals and simplified. The instances are available in an XML format along with best found results for each instance. Researchers validate their solutions using the publicly available evaluator. The authors provide a graphical user interface, a parser and some examples of solvers. Researchers incorporate their solution methods within the framework, possibly reducing development time. A drawback is the system dependency, as the offered tools are only available on a Windows platform.

Vanhoucke and Maenhout (2009) implemented an instance generator for the nurse scheduling problem. A number of complexity indicators are introduced. Using the generator and the complexity indicators, an extensive set of artificial instances, called NSPLib, has been created which, due to its size, allows for proper statistical analysis and comparison of solutions techniques. The instances are available in a proprietary text-only data format.

An attempt towards a generic model, allowing to define a broad variety of problems, is made by Bilgin et al. (2012). The authors claim the proposed model to have an increased ability to reflect extra soft constraints in different hospitals, sectors and countries. The authors present a set of real world inspired

benchmarks. With each instance, a number of different scenarios are associated, reflecting issues that real life wards need to deal with. Examples of such scenarios are overload of work and unexpected absence of nurses.

The problem of unexpected absences of nurses was identified by Warner (1976). A branch-and-bound algorithm assigns nurses from a pool of "floating" nurses from different wards when shortages arise.

Weil et al. (1995) suggest a Constraint Programming based approach. Feasible rosters are generated successively. No optimisation takes place, leaving the decision on the quality of the proposed rosters to the user of the system (e.g. a head nurse).

Moz and Vaz Pato (2003) identify the nurse rerostering problem at a hospital in Lisbon. For handling shortages, the schedules of the other nurses are altered. The changes to the rosters should be minimal and, analogously to the original nurse rostering problem, may not be in conflict with legal, organisational and contractual regulations. The problem is formulated as an integer multi commodity flow problem with additional constraints. The authors applied a heuristic to solve the problem and solved an integer linear programming formulation of the model using CPLEX. In follow-up papers, a genetic algorithm approach (Moz and Vaz Pato, 2007) and a utopic Pareto genetic heuristic (Vaz Pato and Moz, 2008) are applied to the same problem.

Maenhout and Vanhoucke (2011) tackle the rerostering problem using an evolutionary approach. A benchmark dataset for the rerostering problem is introduced. The set is based on the artificial instances of NSPLib. Similarly to NSPLib, a set of schedule disruption characteristics is used to generate the rerostering problem instances.

Bard and Purnomo (2006) consider both altering the schedules of a subset of nurses and hiring 'traveling nurses', identified as the *nurse addition problem*, as a solution for shortages. They studied the problem for hospitals in the UK and Great Britain where the schedules of nurses are fixed by contract and therefore difficult to change. A column generation method is proposed for the 'alternation problem'. A branch and price algorithm is applied to the nurse addition problem.

2.2 Negotiation

Software systems involving multiple, often distributed parties are frequently conceptualised and implemented using a multi-agent system (Wooldridge, 2009). An agent is a computer system capable of performing autonomous

actions in an environment in order to meet certain objectives. In certain environments, agents need to interact in order to reach their goals. Agents interact with each other in a negotiation. Negotiation is an umbrella term for methods and techniques that allow agents to communicate in order to reach mutually beneficial agreements (Kraus, 2001; Raiffa, 1982).

According to Rosenschein and Zlotkin (1994) a negotiation model consists of the following four components:

- the negotiation protocol: The protocol is the specification of the 'rules of encounter' between the negotiating agents. Which deals can be made and which sequence of actions/offers is valid?
- the negotiation strategies: How does an agent decide which action to take next? Possible actions are for example making a bid or conceding. The outcome of the negotiation protocol is strongly influenced by the specific strategies chosen by the agents.
- the information state of the agents: This component describes the information opponents have about each other. Typically, the classification is made between complete and incomplete information (Von Neumann and Morgenstern, 1944; Jennings et al., 2001; Marsa-Maestre et al., 2008).
- the negotiation equilibrium: The outcome of a negotiation process should be stable. No agent should have the incentive to deviate from agreed strategies.

Automated negotiation (Jennings et al., 2001) is an important research area with contributions from fields like economics, game theory, computer science and artificial intelligence. In many applications agents need to reach agreements on the allocation of tasks (Cowling et al., 2003) or the division of resources among themselves (Lau et al., 2007; Hutzschenreuter et al., 2009). In general, the subjects of the negotiation are called issues. Agents typically describe their preference over issues through a utility function. During negotiation, the agents use these utility functions to evaluate bids on the issues and to identify proper counter proposals in case a bid is unsatisfiable.

Negotiation protocols are designed for managing either a single issue or multiple issues. The literature involving single issue negotiation is exhaustive. We restrict the discussion to a few contributions relevant to this thesis.

Smith (1980) designed the well known *Contract Net Protocol* (CNP). A set of *manager* agents negotiate a contract with a number of *contractor* agents that are able to perform a set of tasks.

Aknine et al. (2004) developed an extension to the CNP to circumvent some limitations. As multiple managers can concurrently negotiate with multiple contractors, the total length of the negotiation processes when applying the CNP in its original form may become excessive. When negotiating in parallel, agents decommit contracts already agreed on, when better offers are received. This decommission makes managers to re-embark some negotiation processes to find new contractors for their tasks. Finally, the original CNP does not account for the fact that agents may fail during the negotiation process. The most important extensions are to incorporate multiple stages in the negotiation process and the introduction of a deadline for the reception of proposals by the manager.

Auctions are another type of negotiation protocols, frequently applied for the distribution of task or allocation of resources. For example, the well known Vickrey auction (Vickrey, 1961) for single items, by design, enforces buyers to bid their true valuations of an item. The Vickrey-Clarke-Groves mechanism (Clarke, 1971; Groves, 1973) is a generalisation for multiple or divisible goods. In an auction with M goods, buyers make offers for one out of $2^M - 1$ possible packages.

The Simultaneous Ascending Auction, introduced to sell licenses for bands of radio spectrum in the United Stated, was studied by Milgrom (2000). In this type of auction, buyers bid on multiple objects simultaneously. The highest bid of a round is the minimal value of acceptable bids in the consecutive round. The auction stops if no new bids have been received. Items are assigned to the highest bidder. With a so called 'activity rule', putting lower bounds on bids, pressure is created on bidders to participate actively during the auction, therefore increasing the pace of the auction. Because of the accelerated pace, more information becomes available to the bidders as the auction continues.

In an Ascending Proxy Auction (Ausubel and Milgrom, 2004), in contrast to the previous auction, bids are not made for all items simultaneously, but for a limited number of items, bundles in a package. After each round, provisionally winners are announced. A bidder that was not announced as winner in a previous round is allowed to make a new bid. This new bid may either be a raised bid for the same package for which a bid was made in the previous round, or a bid for an alternative package. After each round, the seller determines the combination of non overlapping packages maximising its revenue. The auction stops after a predetermined number of rounds or if an alternative stopping criterion is met.

The previous auctioning techniques are closely related to mechanism theory based systems (Jackson, 2001). The main focus of mechanism theory/design is the development of interactive systems that satisfy certain objectives assuming

that participants will act strategically and may still keep information private. Examples of such objectives are forcing agents to reveal true valuations on items, discouraging agents to lie or cheat when negotiating, ... For example, Zlotkin and Rosenschein (1996) study the resilience against lying and cheating for automated negotiation in task oriented domains. Wellman et al. (2001) study the application of ascending auction, package (combinatorial) auctions and the generalised Vickrey auction for theoretical distributed scheduling problems. The focus is on theoretical aspects such as the existence of equilibrium prices and the quality of equilibrium solutions. Unfortunately, most of those type of approaches are unable to cope with the complexities that arise when facing 'real world' inspired problems.

In the remainder of this section, we focus on negotiation involving multiple issues. Negotiation in a multi-issue setting may be more complex when compared to single issue negotiation (Lai and Sycara, 2009).

First, the utility functions of the agents depend on all the issues. According to Raiffa (1982), when negotiating multiple issues, it is not the case anymore that one agent gets less utility when an other agent receives more issue. Both can get more by making trade-offs on issues or by conceding on the expected utility level. By making trade-offs, agents try to generate win-win situations in which they are mutually better off. However, this complicates the negotiation because identifying appropriate offers is usually non-trivial. Numerous combinations of issues may exist yielding an agent equal amounts of utility. It may be impossible to compute all of those combinations and therefore hard to find the optimal one.

The level of interdependency of issues has an immediate impact on the complexity of the utility function. Issues may be seen as independent. In this case, issues can still be interrelated, acquiring one issue does not constrain how the other issues are acquired. More complexity arises when issues are interdependent. We distinguish two categories. First, issues can be weakly interdependent. The interdependency creates opportunities for making trade-offs between issues, which may facilitate finding a suitable counter offer that maximizes the utility of the opponent. In this case, if an agent receives the full amount of all the issues, it also gains maximum utility. Second, issues can be strongly interrelated. In this case, fixing the amount of one issue puts limits on the amounts of the remaining issues that an agent is willing to acquire. Generally, a higher level of interdependency makes generating offers and counter offers computationally more difficult.

By conceding, agents lower their expected utility level in order to make reaching agreements feasible or to speed up the negotiation process. In many cases, it is straightforward to see that agents are unable to reach an agreement at

their maximum utility level. Concession strategies allow agents to determine the utility level(s) where agreement is possible. Also, in many applications, the utility of agents decreases with time and deadlines may exist after which reaching an agreement is useless (Fatima et al., 2006).

A second reason for higher complexity in multi-issue negotiation is incompleteness of information. Agents may not have a complete view on the valuation of the issues of their adversaries or the information may be variable over time. This lack of information further complicates the calculation of proposals, again making it impossible to find all possible trade-offs. Therefore the negotiation process may result in a suboptimal outcome.

Third and also following from the first two, Pareto optimality is harder to achieve. In a rational environment, there should not be 'wasted utility'. When a non Pareto optimal outcome has been achieved, this means another agent can possibly improve its utility without harming any of its adversaries. This also means that one of the agents possibly conceded too much. Often a mediating approach is used for helping the agents achieving Pareto optimality.

Lai et al. (2004) provide an overview of multi-issue negotiation contributions until 2004. The authors discuss approaches in the field of Economics and AI. In the field of Economics two branches are reviewed: non-cooperative and cooperative multi-attribute negotiation. Within AI the approaches are categorised according to negotiation framework, trade-off mechanism and searching method. Lai and Sycara (2009) mainly focus on incomplete information, Pareto optimality and tractability. Marsa-Maestre et al. (2011) discuss the literature on the complexity of the preference spaces.

We extend the taxonomy for automated negotiation, proposed by Buttner (2006), to enable the categorisation of multi-issue negotiation contributions and we classify them accordingly. Figure 2.1 shows the extensions to the taxonomy in bold. Sections 2.2.1 to 2.2.5 discuss the various components of the taxonomy. For each element, relevant literature is highlighted.

2.2.1 Multi-attribute versus multi-issue: definition and classification

As An et al. (2011) point out, there are two different definitions of a negotiation issue in the literature.

A first definition of an issue is a resource (good, product, task to perform, ...). For example, in a supply chain, a manufacturer typically has to acquire multiple resources from possibly multiple suppliers. In a second definition, an issue is

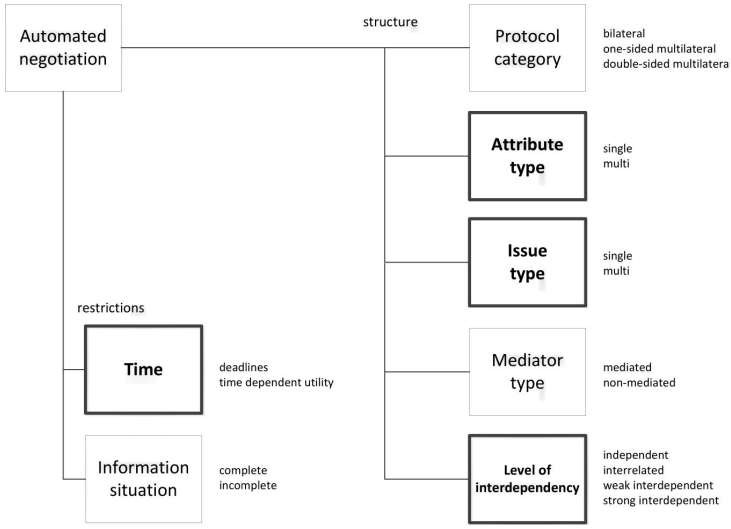

Figure 2.1: Taxonomy for automated multi-issue negotiation

considered an attribute of a resource. In many contributions, such issues are called attributes, to distinguish them from the first type of issues. According to Fatima et al. (2004), this type of negotiation settings can be regarded as multi-issue negotiation problems of the first type. The only limitation is that issues have to be negotiated simultaneously because they are associated with only one good, product or resource. For example, when buying a car, one typically discusses numerous possible options among which price, colour, engine and type of wheels.

These two meanings and the terms 'issue' and 'attribute' are used interchangeably in literature. As mentioned before, within the current contribution, the term issue designates subjects of negotiation. These subjects can be resources for manufacturing products or completing tasks. Goods in an auction or eCommerce setting (Guttman et al., 1998) may also be subjects. The former subjects are examples of tangible issues. Intangible properties of a subject such as the delivery time of a product or the price and amount of a resource under negotiation should also be considered issues within automated negotiation.

Equally important are local decision making parameters, often stemming from local optimisation problems. Those parameters, called attributes, influence the valuation of the negotiation issues. For example in Duan et al. (2011) the authors consider a negotiation setting with a supplier and a manufacturer. The agents negotiate the delivery schedule of issues such as timing, quantities

and price for various components and products. Thus, the local combinatorial optimisation problems of both need to be matched. The authors look at concessions in the attribute space, defined by the local optimisation problem, rather than concessions on the negotiation issues itself.

At the negotiation level, we distinguish single and multi-issue negotiation. Also, we can distinguish problems where local parameters influence the negotiation and problems where the local level does not play part in the negotiation. The former situation can be regarded as multi-attribute negotiation. Multiple attributes enable searching for trade-offs and making concessions in the attribute space. The latter setting can be regarded as single attribute negotiation. Except for conceding to a lower utility level, no more options exist at the local level to facilitate the negotiation.

Consequently, the literature can be classified according to four categories:

1. single-issue, single-attribute (SI,SA)
2. single-issue, multi-attribute (SI,MA)
3. multi-issue, single-attribute (MI,SA)
4. multi-issue, multi-attribute (MI,MA)

Table 2.1 shows an overview of relevant literature of the last decade, specifically on negotiation protocols, within the field of multi-issue negotiation. The first category (SI,SA) is not within the scope of this thesis and remains undiscussed. Most contributions fit into the third category, (MI,SA). We did not find contributions for (SI,MA). In our opinion, most techniques in (MI,SA) are applicable to (SI,MA). The previously mentioned mix-up between the terms issue and attribute supports this statement. One contribution handles a (MI,MA) negotiation problem. Note that Duan et al. (2011) is found in both (MI,MA) and (MI,SA). The authors design two algorithms, one for each category.

2.2.2 Classification by level of interdependency

As mentioned before, the level of interdependency directly influences the complexity of the negotiation. The higher the level of interdependency, the higher the complexity. We distinguish between independent (often interrelated), weakly and strongly interdependent issues in order of raising

Attribute type	Contributions
Single-attribute	Faratin et al. (2002)
	Luo et al. (2003)
	Fatima et al. (2004)
	Lau et al. (2004)
	Fatima et al. (2006)
	Jonker et al. (2007)
	Ito et al. (2008)
	Ragone et al. (2008)
	Fatima et al. (2009)
	Lopez-Carmona et al. (2010)
	Lai and Sycara (2009)
	Wu et al. (2009)
	Lopez-Carmona et al. (2011)
	Marsa-Maestre et al. (2011)
	Duan et al. (2011)
Multi-attribute	Duan et al. (2011)

Table 2.1: Categorisation of multi-issue contributions by attribute structure

complexity. Table 2.2 summarises the classification of literature within these three categories.

In (An et al., 2011), the authors explicitly state independent but interrelated issues are considered. In fact, the total utility for an agent is calculated as the sum of the utility the agent receives for the resources separately. Analogously, Fatima et al. (2004, 2006); Jonker et al. (2007); Lau et al. (2004); Ragone et al. (2008) regard the issues independent but interrelated. Faratin et al. (2002) discuss interrelated issues. The valuation on the issues is a weighted sum of the valuation of each issue separately. The weights represent the relative importance of an agent on the issues.

Wu et al. (2009) study two weakly interdependent issues. They introduce indifference curves representing the combination of utility of equal importance for an agent. Also, at a certain utility level, an indifference curve shows the minimal amount an agent wants for one issue if the amount of the other issue is fixed. However, the agent is still satisfied when gaining more than this minimal amount. Within (Lai and Sycara, 2009), the issues are also weakly interdependent. The authors allow non-linear utility functions but impose important restrictions on the utility function. Concerning the level of interdependency, especially the limitation to strict convexity of the utility function is important: "for any solution x, the set of solutions preferred over x by an agent is strictly convex". Thus, fixing all issues but one, the utility of the

agent is either monotonically increasing or decreasing when raising or lowering the value of the remaining issue.

Duan et al. (2011) omit the restriction to strictly convex utility functions. The above monotonicity is no longer present. Therefore these authors study negotiation for strongly interdependent issues. Ito et al. (2008), Lopez-Carmona et al. (2011), Lopez-Carmona et al. (2010) and Marsa-Maestre et al. (2011) do not put restrictions on the utility functions and therefore also consider strongly interdependent issues.

Interdependency	**Contributions**
independent	An et al. (2011) Fatima et al. (2004) Faratin et al. (2002) Jonker et al. (2007) Lau et al. (2004) Ragone et al. (2008)
weakly interdependent	Lai and Sycara (2009) Wu et al. (2009)
strongly interdependent	Duan et al. (2011) Ito et al. (2008) Lopez-Carmona et al. (2011) Lopez-Carmona et al. (2010) Marsa-Maestre et al. (2011)

Table 2.2: Categorisation on level of interdependency of issues

2.2.3 Classification by protocol category

We distinguish three different types of protocol category: bilateral, one-sided and double-sided multilateral negotiation. As an example, Figure 2.2 shows the different possibilities within the context of buyers and sellers. In a bilateral negotiation setting, negotiation is restricted to two negotiation parties. For example, one buyer and one seller are negotiating on selling issues. One-sided multilateral negotiation means either one buyer and multiple sellers or many buyers and one seller are negotiating. Double-sided multilateral negotiation is characterised by many buyers negotiating with many sellers.

In general, the more parties involved, the more complex the negotiation. In the buyers and sellers example above, this gives raise to a significant increase in the number of possible offers. Also making trade-offs becomes harder as the number of opponents (and their utility function in a complete information environment)

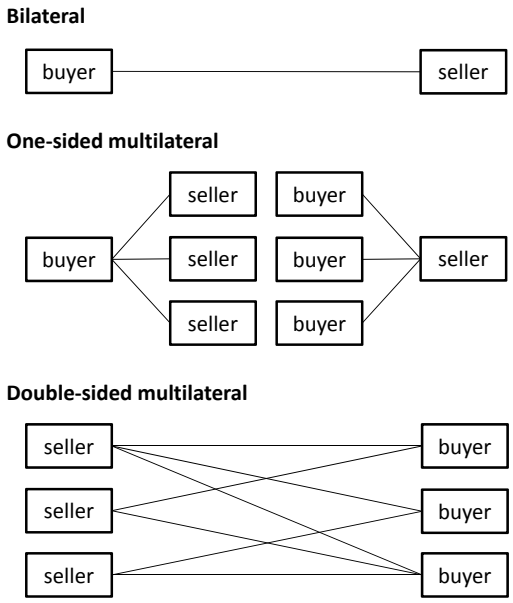

Figure 2.2: Protocol categories

to be taken into account increases. The structure of the negotiation between the parties further influences the complexity. A seller that has no opponent for his buyers to negotiate with, is better off than when multiple sellers offer the same product and vice versa.

Thus, bilateral negotiation is the protocol category with the lowest complexity, followed by one-sided multilateral negotiation. The double-sided multilateral protocol category resembles the highest complexity.

Table 2.3 shows the categorisation of literature by protocol category.

2.2.4 Categorisation by Information situation and Mediator type

In an environment with complete information, where the utility functions of the agents are common knowledge, it is possible to calculate the Pareto optimal frontier. However, in multi-issue negotiation, the calculation of the frontier may become intractable because of the increased complexity of the utility function and the mass of information to be taken into account.

Protocol category	Contributions
Bilateral	Marsa-Maestre et al. (2011) Ragone et al. (2008) Luo et al. (2003) Lopez-Carmona et al. (2011) Lau et al. (2004) Jonker et al. (2007) Faratin et al. (2002) Fatima et al. (2006) Fatima et al. (2004) Duan et al. (2011) Lai and Sycara (2009)
One-sided multilateral	An et al. (2011) Lopez-Carmona et al. (2010) Ito et al. (2008)
Double-sided multilateral	Wu et al. (2009)

Table 2.3: Categorisation by protocol category

Information may also be incomplete. Information may evolve over time or it may be impossible to collect all information beforehand. In an environment with self-interested agents, information is typically kept private. Agents do not willingly share their preferences with adversaries. The information may be misused and thus may lead to inferior solutions. In such a negotiation setting, agents try to minimise the amount of exchanged information in order to prevent opponents from learning their utility functions. As a consequence, it is impossible to calculate the Pareto optimal frontier and therefore more complicated to reach a Pareto optimal outcome (Klein et al., 2003). Frequently, a 'trusted' mediating party is added to the negotiation setting, helping the negotiators to find Pareto optimal solutions. In this way, agents are able to reveal more of their internal information without the risk of exposing themselves to their opponents.

From the information above, it is clear that calculating proper trade-offs in multi-issue negotiation is a non-trivial problem. The difficulty level of generating proposals and finding Pareto optimal solutions is related to the amount and nature of the information agents put at the disposal of their opponents and the mediator when present.

A first category is when only offers and counter-offers are exchanged. In this category, the least information is exposed. Generating trade-offs and obtaining Pareto optimality can be very hard within this category. Instead of proposing only one counter-offer, agents can submit a set of trade-offs that are of equal

importance or have a minimum utility. In this second category, opponents are able to choose the counter-offer from the set that maximises their own utility. This can speed up the negotiation process and improve the optimality of the outcome. Third, agents can choose to exchange regions within their preference space instead of explicitly making bids. For example, in (Marsa-Maestre et al., 2011) agents first recursively negotiate on overlapping regions of shrinking size within their preference space until agreement is found. Finally, a fourth category represents negotiation settings with complete information. For each category mentioned, agents may choose to also send along their valuations on the proposed offer.

Table 2.4 shows the categorisation of literature according to the aforementioned criteria and to the presence of a mediator within the negotiation setting. Some contributions discuss multiple "Information situations".

	Information					
Contribution	bids	set of bids	region	complete	valuation	mediator
Fatima et al. (2004)	•					
Lopez-Carmona et al. (2011)			•			
Marsa-Maestre et al. (2011)			•			
Lai and Sycara (2009)	•					•
An et al. (2011)		•				•
Fatima et al. (2006)	•			•		
Duan et al. (2011)		•			•	•
Jonker et al. (2007)	•					
Ragone et al. (2008)		•			•	•
Luo et al. (2003)	•				•	
Lopez-Carmona et al. (2010)		•			•	•
Ito et al. (2008)	•				•	•
Lau et al. (2004)	•					
Faratin et al. (2002)	•					
Wu et al. (2009)	•					

Table 2.4: Categorisation by Information situation and Mediator type

2.2.5 Categorisation by Time

Time may be taken into account in two ways. An agent can introduce a deadline, the latest moment to reach an agreement in the negotiation process. If the deadline has passed and no agreement has been reached, the negotiation ends without success. Alternatively, the utility an agent receives may decrease over time. This does not mean that contributions on protocols where concession takes place automatically deal with time dependent utility functions. Conceding may take place when no agreement at a certain utility level is possible or the moment of conceding may be time dependent, i.e. the agents concede when no agreement has been reached after a certain amount of time. The categorisation of literature by "Time" is shown in Table 2.5.

	deadlines	time dependent utility
An et al. (2011)	•	
Duan et al. (2011)	•	•
Marsa-Maestre et al. (2011)	•	
Lai and Sycara (2009)	•	•
Fatima et al. (2006)		•
Lopez-Carmona et al. (2011)	•	
Lopez-Carmona et al. (2010)	•	
Ito et al. (2008)	•	
Lau et al. (2004)	•	•
Fatima et al. (2004)	•	

Table 2.5: Categorisation by Time

Chapter 3

Formal models for nurse rostering problems

The nurse rostering problem involves constructing a roster for nurses in a ward of a hospital taking several constraints into account. The constraints stem from organisational requirements, legal regulations and personal requests from nurses. We define *shift types* as the time frames during which a ward wants a nurse with a specific *skill* to work[1]. A *shift* is a particular shift type on a specific day for which a ward requests a nurse. A solution to a nurse rostering problem instance is an assignment of shifts to nurses. We consider two levels of constraints: hard and soft constraints. A feasible solution satisfies all hard constraints. The quality of a solution is measured in soft constraint violations. By putting weights, the relative importance of the constraints is set. The instances we consider within this thesis can be classified as $AS2I|VNO|P$ according to the categorisation proposed by De Causmaecker and Vanden Berghe (2011):

- A: the nurse rostering under study consideres constraints related to minimum and maximum availibity of nurses.
- S: we consider constraints handling sequences: i.e series of a maximum or a minimum number of working shifts.
- 2: we consider two skill types

[1] In this thesis, we relate the required skill to the shift type. If for the same time frame different skills are required, different shift types are introduced.

- I: the problem under study considered individual skill definitions
- V: the coverage constraints are variable over time
- N: the number of shifts is variable
- O: shift types can overlap
- P: goal is to minimise the number of (personnel and coverage) constraints violations

We discuss the model of the nurse rostering problem that is the subject of this thesis. In Section 3.1 we give an informal description of the constraints considered. Section 3.2 gives a formal definition of the constraints by using numberings (Burke et al., 2001). Finally, Section 3.3 elaborates on an exact description of the problem using a mathematical model.

3.1 Description

A nurse rostering problem consists of the following:

- a planning horizon: The number of consecutive days during which the nurses need to be assigned. The planning horizon can be of arbitrary size.
- shift types: Each day is divided in a number of (possibly overlapping) shift types. A shift type represents a time frame for which a nurse with a certain skill set is required. Common shift types are for example early, late, day and night shift types.
- skills: some shift types need to be covered by nurses with specific skills. A nurse can have multiple skills. Having a certain skill type enables a nurse to work shifts that require this skill type.
- nurse pool: A set of nurses. Each nurse works following a contract describing their working regulations. Nurses have certain skills and have personal requests (not) to work certain days/shifts.

We consider the following constraints:

- Contract related constraints:

  - The maximum and minimum number of shifts that can be assigned to a nurse during the scheduling period.
  - The maximum and minimum number of consecutive days on which a shift can be assigned to a nurse.
  - The maximum and minimum number of consecutive days on which a nurse does not have a shift assigned.
  - The maximum of consecutive working weekends.
  - Whether a nurse works complete weekends (i.e., every day of the weekend).
  - Whether a nurse needs to work the same shift type on all days of a working weekend.
  - Alternative skill - whether or not nurses are entitled to work shift types for which they do not have the required skills.
  - Unwanted shift patterns - a sequence of consecutive assignments not wanted by a nurse: e.g., the employee should not work an early shift after a night shift. A more in detail description of unwanted shift patterns is given below.

- Requests: A nurse can request (not) to work a particular shift on a certain day, or even the complete day.
- Hard constraints:
  - Single assignment per day: Nurses should work at most 1 shift type per day.
  - Coverage constraints. These are the personnel requirements. For each shift type on each day, the number of required nurses is specified. The coverage constraints are often referred to as demand.

Unwanted shift patterns

An unwanted pattern is a sequence of assignments that a nurse does not want to work. We distinguish between patterns that are unwanted on specific days (e.g. a nurse does not want to work a night shift before a free weekend, a nurse wants to work on Friday before a working weekend, ...) and patterns that are unwanted throughout the entire planning period (e.g. a nurse should not work a late shift before an early shift, ...).

A pattern consists of a number of pattern entries X: $[X]_{1\ldots\ n}$. A pattern entry X can be one of the following:

- ST: a specific shift type
- W: any shift type on a day
- F: free (no shift type) on a day

A pattern entry X can occur on any day in the scheduling period or on a specific day. We introduce the following pattern types:

1. $\{W, ST\} - [F]_{2...n}$: Before a series of free days, it is prohibited to work any or a specific shift type. E.g. an employee may not work a night shift before a free weekend.
2. $F - [W, ST]_{2...n}$: No free day can occur before working any of a number of consecutive days or shift types. E.g. if an employee works a shift in a weekend, the employee should also work on Friday.
3. $[ST]_{2...n}$: Unwanted shift type successions. E.g. Late-Early-Late, Night-Early, ...

3.2 Formal representation of nurse rostering problems using numberings

The numbering method in (Burke et al., 2001) was originally designed for the efficient evaluation of constraint violations for nurse rostering problems. The numberings also serve as a formal representation of constraints. After we elaborate on the definition of numberings, we give an example numbering for each constraint mentioned in Section 3.1.

Numberings

We start with some elementary notation.

Definition 1. *A* time unit *is an elementary interval of time in which a nurse can be assigned a shift.*

In our case, shift types determine those intervals. So, each shift type has a corresponding time unit. In the case of the nurse rostering problems considered within this thesis, the number of time units equaled the number of shift types times the number of days in the planning period. Thus, supposing we have a

planning horizon of D days and for each day there are Sh shift types, we have a set T of $D * Sh$ time units. A solution to a nurse rostering problem is then an assignment of nurses to shifts on specific time units.

Numberings on the time units are defined as follows:

Definition 2. *A* numbering N_i *is a mapping of the set of time units onto a set of numbers extended with the symbol* U *i.e.* $N_i : T \rightarrow \{-M, -M + 1, \ldots, 0, 1, \ldots, M - 1, M, U\}$ *where* $i = 1, \ldots, I$ *and* I *is the total number of numberings.* M *is a positive integer and* U *(undefined) is a symbol introduced to represent the time units that are not mapped onto a number.*

The mapping does not need to be into or onto, nor does it need to preserve sequence.

An event is a time unit for which a nurse has a shift type assigned. The idea of the evaluation method (Burke et al., 2001) is to go trough the set of events for which the time units do not have U (undefined) as value. We call these the 'numbered events'. More formally:

Definition 3. *A* personal schedule S_p *for person* p *is a mapping* $S_p : T \rightarrow \{working, free\}$.

Definition 4. *For a given personal schedule* S_p *an event is a time unit* e *for which* $S_p(e) = working$.

Denote by T_{S_p} the set of all time units for which S_p maps to *working*. T_{S_p} is thus the set of time units induced by S_p, or in other words: the set of time units for which nurse p is assigned to work. Denote by T_{N_i} the set of time units for which the numbering N_i is defined.

Definition 5. *The event set* $T_{S_p,N_i} := T_{S_p} \cap T_{N_i}$ *is the set of time units induced by the schedule* S_p*, for which the numbering* N_i *is defined* $(\neq U)$.

Definition 6. *Two events* e_j *and* e_{j+1} *are consecutive with respect to numbering* N_i *if* $N_i(e_{j+1}) - N_i(e_j) = 1$

We can now express the following four groups of constraints:

- total number of assignments (*total*): for a numbering N_i, this constraint type limits the number of events e for which $N_i(e) \neq U$.
- total number of assignments of a certain type (*perType*): for a numbering N_i, this constraint type limits the number of events corresponding to a specific number j $(N_i(e) = j)$.

- consecutive assignments (*consecutiveness*): for a numbering N_i, this constraint limits the length a a sequence of consecutive events in T_{S_p,N_i}.
- gaps between consecutive sequences of assignments (*between*): for a numbering N_i this constraint type limits the maximum gap (e.g. free time) between two non-consecutive events e_j and e_{j+1} (the gap equals $N_i(e_{j+1}) - N_i(e_j)$).

For each constraint type, a maximum and minimum bound can be set, resulting in eight constraint types: *max_total*, *min_total*, *max_consecutive*, *min_consecutive*, *max_between*, *min_between*, *max_pert* and *min_pert*. Four counting variables are introduced:

- total: This counter represents the total number of events for the numbering.
- consecutive: This counter represents the number of consecutive events. If an interruption in a sequence is found, the counter is reset to 0.
- pert: This counter keeps track of the number of events per value in the numbering.
- last: This variable keeps track of the number of the last evaluated event.

With each numbering N_i a set of four of the above counters is associated.

An example is given in Table 3.1. A simple numbering N_i is given for a scheduling period of one week. There is only one shift type. The value of each counter after each event is given. By comparing the value of the counter with the values of the constraints, violations are detected. Per constraint type, a 'cost'-variable can be set, representing the weight of the constraint in the objective function. E.g. if $max\_total < total$ there is a violation with amount $cost\_max\_total.(total - max\_total)$. In the example, for $max\_total = 2$ and $cost\_max\_total = 3$, a penalty of 6 is raised.

We introduce two additional variables $prev\_nr$ and $future\_nr$, denoting how constraints should be evaluated at the borders of the scheduling period. $prev\_nr$ is the number of the last evaluated event from the previous scheduling period. The number of the first event after the planning horizon is $future\_nr$. The original evaluation method (Burke et al., 2001) required detailed information on the assignments in the previous planning period as input. An initialisation algorithm initialised the counters based on this information. As we do not consider history and future in this thesis[2], we

[2]We only want to denote how to evaluate the constraints at the borders of the planning horizon.

	Mo	Tu	We	Th	Fr	Sa	Su
N_i	1	2	3	4	5	6	7
Events	*	*	*				*
last	U	1	2	3	3	3	3
total	1	2	3	3	3	3	4
consecutive	1	2	3	3	3	3	1
pert[1]	1	1	1	1	1	1	1
pert[2]	0	1	1	1	1	1	1
pert[3]	0	0	1	1	1	1	1
pert[4]	0	0	0	0	0	0	0
pert[5]	0	0	0	0	0	0	0
pert[6]	0	0	0	0	0	0	0
pert[7]	0	0	0	0	0	0	1

Table 3.1: Example of counters

simplified the initialisation and the final evaluation algorithm for taking the two additional variables into account. Appendix A shows the changes to the original algorithms.

3.2.1 Description of constraints using numberings

The numberings defined in Section 3.2 can also be used for a formal description of the constraints presented in Section 3.1. We discuss in more detail an example numbering for some constraints (for a planning horizon of 1 week and 2 shift types: early (E) and late (L)) and show that some constraints need multiple numberings. Finally, for an example nurse rostering problem instance, we give numberings for all the constraints.

Example of numbering for 'Maximum and minimum number of assignments'

The time units have arbitrary numbers assigned. The variable max_total is set to the maximum number of assignments. The variable min_total is set to the minimum number of assignments. last_nr and future_nr are Undefined. An example numbering is given in Table 3.2.

	Mon		Tue		Wed		Thu		Fri		Sat		Sun	
Shift	E	L	E	L	E	L	E	L	E	L	E	L	E	L
Numbering	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3.2: Numbering for total number of assignments

Example of numbering for 'Maximum and minimum number of consecutive working days'

Each time unit on the same day has the same number. Consecutive days have consecutive numbers. Max_consecutive is set to the maximum number of consecutive working days. Min_consecutive is set to the minimum number of consecutive working days. last_nr and future_nr are Undefined. An example numbering is given in Table 3.3.

	Mon		Tue		Wed		Thu		Fri		Sat		Sun	
Shift	E	L	E	L	E	L	E	L	E	L	E	L	E	L
Numbering	0	0	1	1	2	2	3	3	4	4	5	5	6	6

Table 3.3: Numbering for the number of consecutive working days

Constraints with Multiple Numberings

Some constraints cannot be expressed using only one numbering. Mostly constraints that do not occur on fixed days, need multiple numberings. The number of numberings needed is equal to the length of the pattern. For example (see Table 3.4) consider the unwanted pattern, L-E-L, of length 3 and a scheduling period of 7 days. Since the pattern can start on any day, we need 3 numberings to achieve this.

	Mon		Tue		Wed		Thu		Fri		Sat		Sun	
	E	L	E	L	E	L	E	L	E	L	E	L	E	L
N_1	U	0	1	U	U	2	U	4	5	U	U	6	U	U
N_2	U	U	U	0	1	U	U	2	U	4	5	U	U	6
N_3	U	U	U	U	U	0	1	U	U	2	U	U	U	U

Table 3.4: Multiple numberings for unwanted pattern $L - E - L$

Example

We give a sample numbering for each constraint described in Section 3.1 in Table 3.5. More formal definitions of the numberings can be found in Appendix B. We consider a planning horizon of two weeks. There are three shift types: an early (E), late (L) and night (N) shift type. A weekend consists of three days: Friday, Saturday and Sunday. Table 3.6 shows the mapping between a numbering and the constraints it can represent. As stated before, note that one particular numbering can represent multiple constraints and that some constraints require multiple numberings.

Day	Thu			Fri			Sat			Sun			Mon			Tue			Wed		
Date	01/07			02/07			03/07			04/07			05/07			06/07			07/07		
ST	E	L	N	E	L	N	E	L	N	E	L	N	E	L	N	E	L	N	E	L	N
N_1	0	0	0	1	1	1	2	2	2	3	3	3	4	4	4	5	5	5	6	6	6
N_2	U	U	U	0	0	0	0	0	0	0	0	0	U	U	U	U	U	U	U	U	U
N_3	U	U	U	0	0	0	1	1	1	2	2	2	U	U	U	U	U	U	U	U	U
N_4	U	U	U	0	1	2	0	1	2	0	1	2	U	U	U	U	U	U	U	U	U
N_5	0	U	U	0	U	U	0	U	U	0	U	U	0	U	U	0	U	U	0	U	U
N_6	U	U	0	1	U	3	4	U	6	7	U	9	10	U	12	13	U	15	16	U	18
N_7	U	U	0	1	1	1	1	1	1	1	1	1	U	U	U	U	U	U	U	U	U
N_8	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
N_9	3	3	3	4	4	4	4	4	4	4	4	4	U	U	U	U	U	U	U	U	U
N_{10}	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
N_{11}	U	U	0	1	1	U	1	1	U	U	U	3	4	4	4	4	4	4	U	U	6
N_{12}	U	U	U	U	U	0	1	1	U	1	1	U	U	U	3	4	4	4	4	4	U
N_{13}	U	U	U	U	U	U	U	U	0	1	1	U	1	1	U	U	U	3	4	4	U
N_{14}	0	0	0	U	U	U	U	U	U	U	U	U	1	1	U	U	U	U	U	U	U
N_{15}	U	U	U	0	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Date	08/07			09/07			10/07			11/07			12/07			13/07			14/07		
N_1	7	7	7	8	8	8	9	9	9	10	10	10	11	11	11	12	12	12	13	13	13
N_2	U	U	U	1	1	1	1	1	1	1	1	1	U	U	U	U	U	U	U	U	U
N_3	U	U	U	4	4	4	5	5	5	6	6	6	U	U	U	U	U	U	U	U	U
N_4	U	U	U	3	4	5	3	4	5	3	4	5	U	U	U	U	U	U	U	U	U
N_5	0	U	U	0	U	U	0	U	U	0	U	U	0	U	U	0	U	U	0	U	U
N_6	19	U	21	22	U	24	25	U	27	28	U	30	31	U	33	34	U	36	37	U	U
N_7	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
N_8	U	U	0	1	1	1	1	1	1	1	1	1	U	U	U	U	U	U	U	U	U
N_9	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
N_{10}	3	3	3	4	4	4	4	4	4	4	4	4	U	U	U	U	U	U	U	U	U
N_{11}	7	7	7	U	U	7	9	U	9	10	10	10	U	U	9	10	U	10	10	U	U
N_{12}	U	U	6	7	7	U	7	7	U	U	U	9	10	10	U	10	10	U	U	U	U
N_{13}	4	4	U	U	U	6	7	7	U	7	7	U	U	U	9	10	10	U	10	10	U
N_{14}	U	U	U	U	U	U	2	2	2	U	U	U	U	U	U	U	U	U	U	U	U
N_{15}	U	U	U	U	U	U	U	1	U	U	U	2	U	U	3	U	U	U	U	U	U

Table 3.5: Example numberings for the constraints of the model

Numbering	Constraints	Constraint Type	Value
N_1	Maximum number of assignments	`max_total`	maximum number of shifts the employee can work during the planning horizon
	Minimum number of assignments	`min_total`	minimum number of shifts the employee can work during the planning horizon
	Maximum number of consecutive working days	`max_consecutive`	maximum number of consecutive days on which a nurse can be assigned a shift
	Minimum number of consecutive working days	`min_consecutive`	minimum number of consecutive days on which a nurse can be assigned a shift
	Maximum number of consecutive free days	`max_between`	maximum gap between two non-consecutive shifts
	Minimum number of consecutive free days	`min_between`	minimum gap between two non-consecutive shifts
	Single assignment per day	`max_pert`	1 for each value in the numbering
N_2	Maximum number of consecutive working weekends	`max_total`	maximum number of weekends a nurse can work during the planning horizon
N_3	Complete weekends	`min_consecutive`	the number of days in a weekend
N_4	Identical complete weekends	`min_pert`	the number of days in a weekend for each value in the numbering
N_5	Alternative skill	`max_pert`	0 for each value of the numbering
N_6	Unwanted pattern type 3: N-E	`max_consecutive`	the number of shift types in the pattern - 1
N_7,N_8	Unwanted pattern type 1	`min_consecutive`	2; `future_nr` is assigned 2
N_9, N_{10}	Unwanted pattern type 2	`max_between`	2 ; `last_nr` is assigned 0
N_{11}, N_{12}, N_{13}	Two free days after a night shift	`max_consecutive`	1
N_{14}	Requested day off	`max_pert`	0 for each value in the numbering
N_{15}	Requested shift off	`max_pert`	0 for each value in the numbering

Table 3.6: Mapping between numberings and constraints

3.3 Mathematical model

We present an exact description of the nurse rostering problem constraints presented in Section 3.1. The objective is to minimize the following sum (3.1) subject to the linear expressions in equations (3.2 - 3.33).

$$\begin{aligned}\text{MINIMIZE}\{\quad & V(T^{max}) + V(T^{min}) + V(CW^{max}) + V(CW^{min}) + \\ & V(CF^{max}) + V(CF^{min}) + V(CWW^{max}) + V(CWW^{min}) + \\ & V(CW) + V(CIW) + V(D^{off}) + V(S^{off}) + V(D^{on}) + \\ & V(S^{on}) + V(SK) + V(PAT^{cons}) + V(PAT^{free})\}\end{aligned} \tag{3.1}$$

Let N be the set of nurses, D the set of days in the scheduling period, W the set of all weekends in the scheduling period, and S the set of shift types. Equation (3.2) defines the binary decision variables $x_{n,d,s}$. These state whether nurse n is working shift type s on day d. A set of auxiliary variables $p_{n,d}$ is defined in expression (3.3). These variables indicate whether nurse n is working any shift type on day d. A similar set of auxiliary variables $q_{n,w}$ is defined in expression (3.4). These indicate whether nurse n is working any shift during weekend w. A weekend w is defined by its first day $d_{w,1}$ and its length k (which is equal for all weekends in the planning horizon)[3].

$$\forall n \in N, \forall d \in D, \forall s \in S : x_{n,d,s} = \begin{cases} 1 \text{ if nurse } n \text{ works shift type } s \text{ on day } d \\ 0 \text{ otherwise} \end{cases} \tag{3.2}$$

$$\forall n \in N, \forall d \in D : -|S| p_{n,d} + \sum_{s \in S} x_{n,d,s} \leq 0 \text{ and } -p_{n,d} + \sum_{s \in S} x_{n,d,s} \geq 0 \tag{3.3}$$

$$\forall n \in N, \forall w \in W : -k q_{n,w} + \sum_{i=0}^{k-1} p_{n,d_{w,1+i}} \leq 0 \text{ and } -q_{n,w} + \sum_{i=0}^{k-1} p_{n,d_{w,1+i}} \geq 0 \tag{3.4}$$

[3]for example: a weekend w of $k = 2$ days consists of the consecutive days $d_{w,1}$ and $d_{w,2}$.

The two hard constraints (single assignment per day and the coverage constraints) are enforced by expressions (3.5) and (3.6). Equation (3.5) defines the coverage requirement constraints. These constraints $C_{d,s}$ state the exact number of nurses required to work shift type s on day d. Inequality (3.6) defines the single assignment per day constraint. It states that nurses should work at most one shift type per day.

$$\forall d \in D, \forall s \in S : \sum_{n \in N} x_{n,d,s} = C_{d,s} \tag{3.5}$$

$$\forall n \in N, \forall d \in D : \sum_{s \in S} x_{n,d,s} \leq 1 \tag{3.6}$$

The soft constraints in the problem are described below. Each nurse n has a weight W_n^C associated with each constraint type C. When violated, the constraints contribute to the objective function proportionally to the degree of violation, multiplied by the individual weight assigned by the nurse to the constraint which causes the violation. The objective function is represented in equation (3.1). The goal is to minimize a sum of weighted constraint violations, which are defined by the expressions (3.7 - 3.33).

Constraints T_n^{max} and T_n^{min} on the total workload of nurses limit respectively the maximum and minimum number of shifts assigned per nurse n. The total number of weighted constraint violations $V(T^{max})$ and $V(T^{min})$ are counted in equations (3.7) and (3.8).

$$V(T^{max}) = \sum_{n \in N} W_n^{T^{max}} max\{(\sum_{d \in D} \sum_{s \in S} x_{n,d,s}) - T_n^{max}, 0\} \tag{3.7}$$

$$V(T^{min}) = \sum_{n \in N} W_n^{T^{min}} max\{T_n^{min} - (\sum_{d \in D} \sum_{s \in S} x_{n,d,s}), 0\} \tag{3.8}$$

The constraints CW_n^{max} and CW_n^{min} limit the maximum and minimum number of consecutive working days for each nurse n. The total number of weighted constraint violations $V(CW^{max})$ and $V(CW^{min})$ are counted in equations (3.9) and (3.10), subject to the inequalities in expressions (3.12) and (3.14). For $d > 0$, the auxiliary variables $t_{n,d,0}$ in inequalities (3.12) indicate whether nurse n works on day d while he or she is free on day $d - 1$. The auxiliary

variables $t_{n,d,i}$ in inequalities (3.14) indicate whether there is a consecutive row of working days (of length i) for nurse n since day d.

$$V(CW^{max}) = \sum_{n \in N} W_n^{CW^{max}} \sum_{d=0}^{|D|-1-CW_n^{max}} max\{(\sum_{i=0}^{CW_n^{max}} t_{n,d,i}) - CW_n^{max}, 0\} \tag{3.9}$$

$$V(CW^{min}) = \sum_{n \in N} W_n^{CW^{min}} \sum_{d \in D} (CW_n^{min} t_{n,d,0} - \sum_{i=0}^{CW_n^{min}-1} t_{n,d,i}) \tag{3.10}$$

$$t_{n,d,0} = p_{n,d} \text{ for } d = 0 \tag{3.11}$$

$$0 \leq t_{n,d,0} \leq 1 - p_{n,d-1} \text{ and } p_{n,d} - p_{n,d-1} \leq t_{n,d,0} \leq p_{n,d} \text{ for } d > 0 \tag{3.12}$$

$$0 \leq t_{n,d,i} \leq p_{n,d} \text{ and } t_{n,d,i} \leq t_{n,d,i-1} \text{ for } i > 0 \tag{3.13}$$

$$t_{n,d,i} = 0 \text{ for } d \notin D \tag{3.14}$$

Similarly, the constraints CF_n^{max} and CF_n^{min} limit the maximum and minimum number of consecutive free days for each nurse n. The total number of weighted constraint violations $V(CF^{max})$ and $V(CF^{min})$ are counted in equations (3.15) and (3.16), subject to the inequalities in (3.18) and (3.20). For $d > 0$, the auxiliary variables $t_{n,d,0}$ in the inequalities (3.18) indicate whether nurse n works on day $d-1$ while he or she is free on day d. The auxiliary variables $t_{n,d,i}$ in the inequalities (3.20) indicate whether there is a consecutive row of free days (of length i) for nurse n since day d.

$$V(CF^{max}) = \sum_{n \in N} W_n^{CF^{max}} \sum_{d=0}^{|D|-1-CF_n^{max}} max\{(\sum_{i=0}^{CF_n^{max}} t_{n,d,i}) - CF_n^{max}, 0\} \tag{3.15}$$

$$V(CF^{min}) = \sum_{n \in N} W_n^{CF^{min}} \sum_{d=0}^{|D|-CF_n^{min}} (CF_n^{min} t_{n,d,0} - \sum_{i=0}^{CF_n^{min}-1} t_{n,d,i}) \quad (3.16)$$

$$t_{n,d,0} = 1 - p_{n,d} \text{ for } d = 0 \quad (3.17)$$

$$0 \leq t_{n,d,0} \leq p_{n,d-1} \text{ and } p_{n,d-1} - p_{n,d} \leq t_{n,d,0} \leq 1 - p_{n,d} \text{ for } d > 0 \quad (3.18)$$

$$0 \leq t_{n,d,i} \leq 1 - p_{n,d} \text{ and } t_{n,d,i} \leq t_{n,d,i-1} \text{ for } i > 0 \quad (3.19)$$

$$t_{n,d,i} = 0 \text{ for } d \notin D \quad (3.20)$$

Consecutiveness constraint CWW_n^{max} limits the maximum number of consecutive working weekends per nurse n. The total number of weighted constraint violations $V(CWW^{max})$ is counted in equation (3.21).

$$V(CWW^{max}) = \sum_{n \in N} W_n^{CWW^{max}} \sum_{w=0}^{|W|-CWW_n^{max}} max\{ \sum_{i=0}^{CWW_n^{max}} q_{n,w+i} - CWW_n^{max}, 0\} \quad (3.21)$$

The complete weekend constraints $CW_n \in \{1, 0\}$ specify whether or not nurse n should work either all days or no days at all during a weekend[4]. The total number of weighted constraint violations $V(CW)$ is defined in equation (3.22).

$$V(CW) = \sum_{n \in N} W_n^{CW} \sum_{w \in W} CW_n (k q_{n,w} - \sum_{i=0}^{k-1} p_{n,d_{w,1+i}}) \quad (3.22)$$

Similarly, the complete identical weekend constraints $CIW_n \in \{1, 0\}$ specify whether or not nurse n should work the same shift type on all days in a (complete) weekend[5]. The total number of weighted constraint violations

[4] $CW_n = 1$ if nurse n needs to work complete weekends, $CW_n = 0$ if she or he can work an arbitrary number of days in a weekend.

[5] $CIW_n = 1$ if nurse n should work identical days during a weekend, $CIW_n = 0$ if he or she can work different shift types during the days of a weekend.

$V(CIW)$ is defined in equation (3.23), subject to the inequalities in (3.24). The auxiliary variables $t_{n,w,s}$ in the inequalities in (3.24) indicate whether nurse n is working shift type s during weekend w.

$$V(CIW) = \sum_{n \in N} W_n^{CIW} \sum_{w \in W} \sum_{s \in S} CIW_n (t_{n,w,s} k - \sum_{i=0}^{k-1} x_{n,d+i,s}) \tag{3.23}$$

$$-t_{n,w,s} k + \sum_{i=0}^{k-1} x_{n,d_{w,1+i},s} \leq 0 \text{ and } -t_{n,w,s} + \sum_{i=0}^{k-1} x_{n,d_{w,1+i},s} \geq 0 \tag{3.24}$$

Nurses can request to be free or to work on specific days or to have certain shifts assigned on a specific day. Let D^{off} be the set of requests $req_{n,d}^{off}$ denoting nurse n does not want to work on day d and D^{on} be the set of requests $req_{n,d}^{on}$ denoting nurse n prefers to work on day d. Let S^{off} be the set of requests $req_{n,d,s}^{off}$ denoting nurse n does not want to work shift type s on day d and S^{on} be the set of requests $req_{n,d,s}^{on}$ denoting nurse n wants to work shift type s on day d. The total number of weighted constraint violations $V(D^{off})$, $V(D^{on})$, $V(S^{off})$ and $V(S^{on})$ are counted in equations (3.25), (3.26), (3.27) and (3.28) respectively.

$$V(D^{off}) = \sum_{req_{n,d}^{off} \in D^{off}} W_n^{D^{off}} \sum_{s \in S} x_{n,d,s} \tag{3.25}$$

$$V(D^{on}) = \sum_{req_{n,d}^{on} \in D^{on}} W_n^{D^{on}} (1 - min\{1, (\sum_{s \in S} x_{n,d,s})\}) \tag{3.26}$$

$$V(S^{off}) = \sum_{req_{n,d,s}^{off} \in S^{off}} W_n^{S^{off}} x_{n,d,s} \tag{3.27}$$

$$V(S^{on}) = \sum_{req_{n,d,s}^{on} \in S^{on}} W_n^{S^{on}} (1 - x_{n,d,s}) \tag{3.28}$$

The alternative skill constraint $SK^{\text{alt}} \in \{0, 1\}$ is common to all nurses. It specifies whether or not nurses can be assigned shift types for which they do not have all the required skills[6]. The weighted number of constraint violations

[6] $SK^{\text{alt}} = 0$ if nurses are allowed to work shift types for which they do not have all the required skills, $SK^{\text{alt}} = 1$ otherwise.

$V(SK)$ is defined in equation (3.29), subject to the inequalities in (3.30) and (3.31). SK is the set of skills sk that are present in the problem, SK_s is the set of skills nurses are requried to have when they are assigned a shift of type s and SK_n is the set of skills of nurse n. The auxiliary variables $r_{n,s}$ in (3.31) denote whether nurse n has the required skills to work shift s. The variables $b_{n,sk}$ denote whether nurse n has skill sk.[7] Hence, the variables $t_{n,d,s}$ in the inequality (3.30) equal 1 if and only if nurse n works shift type s on day d, while not having all the required skills.

$$V(SK) = SK^{\text{alt}}.W^{SK} \sum_{n\in N} \sum_{d\in D} \sum_{s\in S} t_{n,d,s} \tag{3.29}$$

$$t_{n,d,s} \geq x_{n,d,s} - r_{n,s} \tag{3.30}$$

$$-r_{n,s}|SK_s| + \sum_{sk\in SK_s} b_{n,sk} \geq 0 \text{ and } -r_{n,s} - |SK_s| + 1 + \sum_{sk\in SK_s} b_{n,sk} \geq 0 \tag{3.31}$$

We distinguished three unwanted pattern types. The first pattern type avoids nurse to work a specific or any shift on a day before a series of free days. Each nurse has a set PAT_n^{work} of patterns pat that are unwanted. Each pattern pat consists of a starting day sd (i.e. the day on which a shift has to be assigned) and a certain length l. The second pattern type avoids nurses to have a free day before working one of the successive days. Similarly, each nurse has a set PAT_n^{free}. The starting day sd now denotes the day that should be free. The last pattern type is a series of consecutive shifts (starting on any day) that should be avoided. Again, each nurse has a set PAT_n^{cons} with patterns pat denoting an unwanted sequence of shift types s_i and a certain length, l. The total number of weighted violations $V(PAT^{work})$, $V(PAT^{free})$ and $V(PAT^{cons})$ are counted in equations (3.34) and (3.33) respectively.

$$V(PAT^{work}) = \sum_{n\in N} W_n^{PAT^{work}} \sum_{pat\in PAT_n^{work}} p_{n,sd} max\{1 - l - \sum_{i=1}^{l} (1 - p_{n,sd+i}), 0\} \tag{3.32}$$

[7] $b_{n,sk} = 1$ if $sk \in SK_n$ and $b_{n,sk} = 0$ otherwise.

$$V(PAT^{free}) = \sum_{n \in N} W_n^{PAT^{free}} \sum_{pat \in PAT_n^{free}} (1 - p_{n,sd}) max\{(\sum_{i=1}^{l} p_{n,sd+i}) - l + 1, 0\} \tag{3.33}$$

$$V(PAT^{cons}) = \sum_{d \in D} \sum_{n \in N} W_n^{PAT^{cons}} \sum_{pat \in PAT_n^{cons}} max\{\sum_{i=1}^{l} (x_{n,d+i,s_i}) - l + 1, 0\} \tag{3.34}$$

3.4 Conclusion

We gave a detailed description of the nurse rostering problem that is the subject of this thesis. We emphasised the need for a clear, formal and unambiguous representation of the problem under study. If no such description is given, problems may arise. As the model in this thesis is based on the nurse rostering problem considered by Brucker et al. (2010), we highlight a problem that may arise because the authors only gave an informal description of the studied nurse rostering problem (and the associated constraints). For some constraints, an evaluation for one particular simplified sample instance is given. The set of constraints is listed in a table and the violation measurement factor for each constraint is given. The description of some constraints is ambiguous. For example, the violation measurement factor for the constraint "An early shift after a day shift should be avoided" is "Number of early shifts after day shifts". It is unclear what the precise amount of violation is when an early shift is detected after a day shift. Is it the number of early shifts that follow that day shift? Or is it the number of occurrences of an early shift after a day shift? No formal specification on the evaluation of instances is given, leaving the reader to figure out the details of the objective function.

Good practice requires providing solution evaluators for benchmarking instances for validation of research results. Evaluators may not be suitable for incorporating into solution methods. The evaluator may be too slow or implemented in a different programming language. Some methods benefit from delta evaluation of the objective function. For example, some local search neighbourhoods only partially influence the objective function. Hence, only those parts need to be reevaluated. Despite the availability of an evaluator, researchers are still required to build own implementations. This is hard in case only informal descriptions are given. At best, the objective function is defined by a mathematical representation. But even then, ambiguities may arise, leading to misinterpretations and faulty implementations.

For instance, as Smet et al. (2012a) point out, some constraints can be interpreted differently at the borders of the scheduling period. The authors provide a detailed informal description of a rich generic model, extending the model provided by Bilgin et al. (2012), fitting the component where possible in the categorisation of De Causmaecker and Vanden Berghe (2011). Although the authors emphasize the need for a consistent evaluation procedure capable of handling problems arising when considering the previous and future planning horizon, a detailed unambiguous description of the evaluation of the constraint is not provided. For example, if a series of 5 consecutive assignments is required and only 3 consecutive shifts are detected, what is the size of violation? Although one can argue the violation naturally has size 2, this information

should be provided formally to avoid misinterpretations and ambiguities.

The aforementioned contributions all introduce a benchmark set based on the proposed models. The models are available for downloading from some specific websites. Unfortunately, the instances are often provided on personal websites of the researchers who developed the benchmark sets. Problematic about this way of publicly offering benchmark sets is the perishableness of certain websites. For example, the benchmark sets of Brucker et al. (2010) and Bilgin et al. (2012) are, at the time of writing, not available anymore at the website specified within the paper. Special attention should be paid so that data is not made available on transient websites such as personal pages of researchers and temporary project websites.

To overcome the above listed problems, besides an informal description of the nurse rostering problem (and its constraints) under study, we formally defined each constraint by using numberings and by a mathematical formulation. Both representation methods clearly and unambiguously provide details on the evaluation of solutions to problem instances.

The use of numberings allows the use of the efficient evaluation procedure of (Burke et al., 2001). The evaluation method does not need to be altered to incorporate new constraints. It is sufficient to formulate a numbering for the new constraints. It is not in all cases obvious and sometimes even impossible to find a suitable numbering. For example, fairness constraints such as balancing the number of constraints violations between the nurses cannot be expressed using numberings. Currently the roster of one nurse can raise the majority of constraint violations while the other nurses' roster have low costs assigned. Expressing some constraints using numberings may lead to an unnatural evaluation of the constraints. One example we encountered is the constraint requiring nurses to work complete weekends (see Appendix B for more details on the constraint). A natural cost associated with this constraint is the number of non working days in a working weekend. For example consider a weekend consisings of three days (Friday-Saturday-Sunday). If a nurse works on Friday and Sunday and is free on Saturday, naturally a violation of size 1 is detected. However, expressing the constraints using numberings raises a cost of 4.

The use of numberings allows for an automated translation of nurse rostering problem instances into other problem domains. Chapter 4 demonstrates the translation to satifiability and mixed integer problems. As we will show, and analogously to the evaluation mechanism, introducing new constraints does not require altering the translation schemes.

As we do not consider the previous and planning period, it was sufficient to introduce the variables $last\_nr$ and $future\_nr$ denoting how to evaluate the constraints at the borders of the planning period. The method should be extended to take history and future into account. Special attention needs to be paid to consistency. Using the method presented in this thesis for history and future may lead to an inconsistent evaluation of the rosters. Constraint violations at the borders can be counted double, once as a violation with respect to future and once with respect to history.

Chapter 4

Automated translation of nurse rostering problem instances to SAT and MIP models

In this chapter we present schemes for automatically translating nurse rostering problem instances, that can be expressed using the numberings discussed in Chapter 3, into satisfiability problem instances (SAT) and mixed integer programs (MIP).

A Satisfiability (SAT) problem consists of deciding whether a given Boolean formula in conjunctive normal form has an assignment that 'makes the formula true'. Cook (1971) showed that the problem is NP-complete. A formula consist of *literals*, i.e. variables or their negation. For example, x is a positive literal and $\neg x$ is a negative literal. Literals can be combined using $\vee$ and $\wedge$. A formula is in Conjunctive Normal Form (CNF) if it is a conjunction of one or more conjuncts, which are all disjunctions of literals, e.g. $(x_1 \vee x_2) \wedge (x_3 \vee x_4)$. A formula is often referred to as a clause.

The SAT-translation scheme was designed for a hardness study of nurse rostering problem instances (Bilgin et al., 2009). Some algorithms perform well on some specific instances while those same algorithms perform worse on others. Leyton-Brown et al. (2006) present an experimental approach for predicting the run time of algorithms designed to solve the winner determination problem for

combinatorial auctions. The previous strategy was applied to the more abstract problem of propositional satisfiability (Nudelman et al., 2004) introducing a set of 91 features for the runtime prediction of several algorithms that prove whether a certain problem instance is satisfiable or not. This work has led to the construction of SATzilla, a port-folio solver for SAT problems (Xu et al., 2008). This portfolio was very successful, winning several tracks of different SAT competitions[1]. The aforementioned SAT-features set is, combined with hand crafted features representing expert knowledge, used to predict the runtime of a certain algorithm and the value of the objective function obtained by the algorithm.

A second potential of the SAT translation scheme is applying SAT (or MaxSAT) solvers to nurse rostering problem instances. As a first attempt, Acharyya (2008) translated a simple nurse rostering problem with a small set of constraints into SAT and applied GSAT to solve it. The approach is limited to small instances as the number of clauses needed to represent an instance increases rapidly with the size of the instance. It is thus important to develop efficient encodings of the constraints.

Modeling optimisation problems involving constraints on sequences of decision variables to MIP (and SAT) can be very complex (Côté et al., 2011). Special attention is paid to the translation of counting constraints into SAT clauses. We present an efficient translation procedure generating $O(n^2)$ clauses and $O(nlogn)$ variables. Bailleux and Boufkhad (2003) developed a translation scheme with similar complexity. Sinz (2005) provides a $O(n.k)$ and Asín et al. (2009) a $O(nlog^2(k))$ scheme to translate the constraint $x_1 + \ldots + x_n \leq k$. All contributions however, only provide a one way translation, preserving arc consistency. As we do not solely focus on solving nurse rostering problems using SAT, a two way translation is required.

Cadoli and Schaerf (2005) present the automated translation of problem specifications expressed in NP-SEC into SAT. NP-SEC is a logic based language capable of expressing all problems belonging to complexity class NP. The authors tested the system on a few classical problems such as graph colouring and job-shop scheduling, NP-complete problems with a rather simple problem definition. The authors highlight the benefits of an automated approach. Although the performance of SAT solvers on the generated instances is often inferior to manual encodings, the authors stem that the system is a valuable tool for developing fast prototypes for new problems, or variations of known ones for which no specific solver is available.

[1]More information about these competitions can be found at `http://www.satcompetition.org`

4.1 Formal definitions of generic constraints

In this study, we only consider *monotonically ascending* (Definition 7) numberings. As shown in Appendix C, this does not limit the expression of the constraints of our model as introduced in Chapter 3.

Definition 7. *A numbering N_i is* monotonically ascending *if, for every two time units j_a and j_b for which N_i is defined ($\neq U$), $j_a < j_b \Rightarrow N_i(j_a) \leq N_i(j_b)$.*

The constraints on the numberings can be expressed using event sets. An important concept in the expression of constraints is the notion of event sequences, in particular sequences that increase at a steady pace.

Definition 8. *The event set E of a numbering N_i and a personal schedule S_p is the set T_{S_p,N_i}.*

With every event $e \in E$, a unique number ($\in N_i$) is associated. We refer to Section 3.2 for detailed information on the definition of S_p and T_{S_p,N_i}.

Definition 9. *An* event sequence *r is any sequence of events e_j from E, ($j = 0 \ldots m$), conserving the order of the time units corresponding to the events.*

Definition 10. *A sequence r is* contiguously ascending *if $\forall j \in \{1, \ldots, m\}$: $e_j - e_{j-1} = 1$.*

We can now give formal definitions for the constraints in Table 4.1. The constraints presented in Section 3.1 can all be represented by one of these eight constraint types.

4.2 Translation of generic constraints to SAT

We present a scheme to translate each of the eight generic constraints from Table 4.1 to CNF SAT clauses. A preprocessing step is performed before the translation of the constraint types.

4.2.1 Preprocessing

We introduce boolean decision variables $v_{p,j}$ indicating whether nurse p is working on time unit j. Since the expression of the sequence constraints is

Type	Constraint	Value	Definition
consecutiveness	*max_consecutive*	max_c	There is no contiguously ascending event sequence of length $max_c + 1$
consecutiveness	*min_consecutive*	min_c	There is no contiguously ascending event sequence of length l: $min_c > l \geq 1$, which is not part of another ascending event sequence of length at least min_c.
between	*max_between*	max_b	For any two consecutive events e_j and e_{j+1}, $N_i(e_{j+1}) - N_i(e_j) \leq max_b$
between	*min_between*	min_b	For any two consecutive events e_j and e_{j+1}, $N_i(e_{j+1}) - N_i(e_j) \leq 1$ *or* $N_i(e_{j+1}) - N_i(e_j) \geq min_b$
total	*max_total*	max_t	The event set E contains at most max_t events
total	*min_total*	min_t	The event set E contains at least min_t events
perType(k)	*max_perType*(k)	max_{pt}	The event set E contains at most max_{pt} events corrensponding to a number k
perType(k)	*min_perType*(k)	min_{pt}	If the event set E contains an event corresponding to a number k, then it contains at least min_{pt} events corresponding to k

Table 4.1: Formal definition of eight generic constraints

basically the same for all employees, we denote $v_{p,j}$ as v_j, in order to simplify the notation.

For some numberings, consecutive time units are assigned the same number, e.g. numbering N_1 in Table 4.2 does not distinguish between different shifts on the same day. Hence it is natural to introduce a variable t_i indicating for each

sequence of consecutive time units with equal numbers whether an employee is working on a time unit within that sequence. Generally, for such a sequence of time units starting at k and ending at l:

$$t_i \Leftrightarrow \bigvee_{j=k}^{l} v_j$$

In cnf, each variable definition translates into the following clauses:

$$\neg t_i \vee (\bigvee_{j=k}^{l} v_j) \text{ and } \bigwedge_{j=k}^{l} (t_i \vee \neg v_j)$$

This results in $2 + l - k$ clauses. For n numbers in the numbering we generate at most $\lceil n/2 \rceil$ variables and at most $1 + \lceil n/2 \rceil$ clauses. The preprocessing step thus results in $O(n)$ variables and $O(n)$ clauses. An example can be found in Table 4.2.

Preprocessing for between constraints

Translating 'between' constraints requires a slightly different preprocessing step. Where a numbering is interrupted, one or more implicit variables with value 'False' should be placed. An example is given in Table 4.3, both variables $t_3 \equiv False$ and $t_4 \equiv False$. Without the implicit variables, some event sequences would lack. Suppose we want a gap of at most 1 between two consecutive events and an event occurs for t_2 ($v_3 = True$ or $v_4 = True$) and t_5 ($v_5 = True$ or $v_6 = True$). Between the two events, there is a gap of 2. By omitting the implicit variables, the two event sequences detecting the violation would be missing.

Day	Mo		Tu		We		Th		Fr		Sa		Su	
Shift type	E	L	E	L	E	L	E	L	E	L	E	L	E	L
Time unit	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Variable	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}
Numbering N_1	1	1	2	2	3	3	4	4	5	5	6	6	7	7
Preprocessing	t_1		t_2		t_3		t_4		t_5		t_6		t_7	
Numbering N_2	1	1	1	U	U	2	3	3	3	U	U	4	4	4
Preprocessing	t_1		-		t_2		t_3		-		t_4			

Day	Mo		Tu		We		Th		Fr		Sa		Su	
Shift type	E	L	E	L	E	L	E	L	E	L	E	L	E	L
Time unit	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Variable	v_{15}	v_{16}	v_{17}	v_{18}	v_{19}	v_{20}	v_{21}	v_{22}	v_{23}	v_{24}	v_{25}	v_{26}	v_{27}	v_{28}
Numbering N_1	8	8	9	9	10	10	11	11	12	12	13	13	14	14
Preprocessing	t_8		t_9		t_{10}		t_{11}		t_{12}		t_{13}		t_{14}	
Numbering N_2	5	5	5	6	6	6	6	U	U	U	8	8	8	8
Preprocessing	t_5		t_6				—				t_7			

Table 4.2: Preprocessing

Time unit	1	2	3	4	-	-	5	6	7	8
Numbering	1	1	2	2	-	-	5	5	6	6
Variable	v_1	v_2	v_3	v_4	-	-	v_5	v_6	v_7	v_8
Preprocessing	t_1		t_2		t_3	t_4	t_5		t_6	

Table 4.3: Preprocessing for 'between' constraints

4.2.2 Translation of 'consecutive' and 'between' constraints

Maximum number of consecutive events and free time units between two events.

As stated in Table 4.1 there should not be a contiguously ascending event sequence of length $max_c + 1$. For every contiguously ascending sequence *cas* (cas_i is the index in the original sequence of the i^{th} variable within *cas*) we have the following clauses:

$$\neg \left(\bigwedge_{i=0}^{max_c} t_{cas_i} \right), \text{ in cnf: } \bigvee_{i=0}^{max_c} (\neg t_{cas_i})$$

Analogously, for a maximum gap between two events, we obtain:

$$\neg \left(\bigwedge_{i=0}^{max_b} \neg t_{cas_i} \right), \text{ in cnf: } \bigvee_{i=0}^{max_b} (t_{cas_i})$$

In general, translating the constraints for a monotonically ascending numbering with n numbers generates at most $(n - max_c)$, respectively $(n - max_b)$ clauses and no extra variables.

Minimum number of consecutive events and free time units between two events

Following Table 4.1, there should not be a contiguously ascending event sequence of length l ($min_c > l > 1$) which is not part of another contiguously ascending event sequence of length at least min_c. This implies that in any

consecutive sequence of variables t_i of length l ($min_c + 1 \geq l \geq 3$) the middle variables cannot be true without one of the border variables. For every contiguously ascending sequence *cas* of length l ($3 \leq l \leq min_c + 1$) this results in the following clauses:

$$\neg \left(\neg t_{cas_0} \wedge \neg t_{cas_{l-1}} \wedge \left(\bigwedge_{i=1}^{l-2} t_{cas_i}) \right) \right), \text{ in cnf: } t_{cas_0} \vee t_{cas_{l-1}} \vee \left(\bigvee_{i=1}^{l-2} \neg t_{cas_i} \right)$$

Analogously, for a minimum gap between two events, we obtain:

$$\neg \left(t_{cas_0} \wedge t_{cas_{l-1}} \wedge \left(\bigwedge_{i=1}^{l-2} \neg t_{cas_i}) \right) \right), \text{ in cnf: } \neg t_{cas_0} \vee \neg t_{cas_{l-1}} \vee \left(\bigvee_{i=1}^{l-2} t_{cas_i} \right)$$

In general, translating the constraints for a monotonically ascending numbering consisting of n numbers, generates at most $(n - min_c)$ clauses and no extra variables.

4.2.3 Translation of counting constraints

We developed an efficient general procedure for translating counting constraints into CNF clauses. First we elaborate on the procedure in general. Then we show how to translate *total* and *pert* constraint types using this procedure.

General procedure

This procedure uses an iterative process of introducing variables and clauses. We start by some elementary definitions.

Definition 11. *V is the set of variables v_i for which we want to count the number of variables that are assigned true.*

Definition 12. *$U_{\alpha,\beta}$ is the set of indices (of time units) k between α and β for which v_k in V is assigned true.*

Definition 13. *$F_{\alpha,\beta}$ is the set of indices (of time units) k between α and β for which v_k in V is assigned false.*

We then introduce the variables u_i, respectively f_i, denoting whether there are at least i elements in the set $U_{1,n}$, respectively $F_{1,n}$, with n the number of elements in V:

$$u_i \Leftrightarrow |U_{1,n}| \geq i \quad \text{for} \quad i \in \{1, \ldots, n\} \tag{4.1}$$

$$f_i \Leftrightarrow |F_{1,n}| \geq i \quad \text{for} \quad i \in \{1, \ldots, n\} \tag{4.2}$$

More generally, we introduce the variables $u_{x,y,z}$ ($f_{x,y,z}$) denoting whether there are at least z events in the set $U_{x,y}$ ($F_{x,y}$):

$$u_{x,y,z} \Leftrightarrow |U_{x,y}| \geq z \tag{4.3}$$

$$f_{x,y,z} \Leftrightarrow |F_{x,y}| \geq z \tag{4.4}$$

In the remainder of this section, we only consider equivalence (4.3). As $u_i = u_{1,n,i}$, for representing equivalence (4.1), we show we only need to translate:

$$u_{1,n,i} \Rightarrow |U_{1,n}| \geq i \tag{4.5}$$

By contradiction, the other direction of the equivalence

$$|U_{1,n}| \geq i \Rightarrow u_{1,n,i} \tag{4.6}$$

is equivalent with

$$\neg u_{1,n,i} \Rightarrow \neg(|U_{1,n}| \geq i) \tag{4.7}$$

If no i variables may be assigned $true$, at least $n - i + 1$ variables should be assigned $false$:

$$\neg u_i \Leftrightarrow (|U_{1,n}| < i) \Leftrightarrow (|F_{1,n}| \geq n - i + 1) \Leftrightarrow f_{n-i+1} \tag{4.8}$$

Using (4.8), implication (4.7) becomes:

$$f_{n-i+1} \Rightarrow |F_{1,n}| \geq n - i + 1 \tag{4.9}$$

This can be translated in the same way as we translate (4.5).

Translation of $u_{1,n,i} \Rightarrow |U_{1,n}| \geq i$

First we prove the equivalence of $u_{1,n,i}$ and $(u_{1,\frac{n}{2},k} \vee u_{\frac{n}{2}+1,n,l})$ for $k+l = i+1$.

Lemma 1. *$u_{1,n,i} \Rightarrow (u_{1,\frac{n}{2},k} \vee u_{\frac{n}{2}+1,n,l})$ for $k + l = i + 1$*

In words, when $U_{1,n}$ contains at least i elements, then $U_{1,\frac{n}{2}}$ contains at least k elements or $U_{\frac{n}{2}+1,n}$ contains at least l elements. We will prove this by contradiction.

Proof. Let k_0 be the number of elements in $U_{1,\frac{n}{2}}$ and l_0 the number of elements $U_{\frac{n}{2}+1,n}$, then $i_0 = k_0 + l_0$ is the number of elements in $U_{1,n}$.
Suppose that the right part of the implication does not hold, then we have $k_0 < k$ and $l_0 < l$

$$\begin{array}{rcccl}
 & (& k_0 < k &) \text{ and } (& l_0 < l \quad) \\
\Leftrightarrow & (& k_0 \leq k-1 &) \text{ and } (& l_0 \leq l-1 \quad) \\
\Rightarrow & (& k_0 + l_0 &) \leq (& k+l-1-1) \\
\Leftrightarrow & & i_0 & \leq (& i-1 \quad) \\
\Leftrightarrow & & i_0 & < & i
\end{array}$$

which is per definition $\neg u_{1,n,i}$ and thus contradicts the left part of the implication. □

Using the above lemma we can split up the event sets and rewrite the definition of $u_{1,n,i}$ as:

$$u_{1,n,i} \Rightarrow (u_{1,\frac{n}{2},k} \vee u_{\frac{n}{2}+1,n,l}) \quad \text{for all} \quad k \geq 0, l \geq 0 \quad \text{s.t.} \quad k+l = i+1, i \in \{1, \ldots, n\}$$

This is equivalent to:

$$u_{1,n,i} \Rightarrow \bigwedge_{k,l} (u_{1,\frac{n}{2},k} \vee u_{\frac{n}{2}+1,n,l}) \quad \text{for all} \quad k \geq 0, l \geq 0 \quad \text{s.t.} \quad k+l = i+1, i \in \{1, \ldots, n\}$$

This implication is then formulated as a CNF formula:

$$\bigwedge_{k,l}(\neg u_{1,n,i} \vee u_{1,\frac{n}{2},k} \vee u_{\frac{n}{2}+1,n,l}) \quad \text{for all} \quad k \geq 0, l \geq 0 \quad \text{s.t.} \quad k+l=i$$

Given the limitations we can find $(i+2)$ pairs of values for k and l. Hence we need a total of $(i+2)$ clauses to represent this implication as a cnf formula. This procedure essentially expresses the variable $u_{1,n,i}$ in terms of lower level variables $u_{1,\frac{n}{2},k}$ and $u_{\frac{n}{2}+1,n,l}$. We need $2(i+2)$ lower level variables to represent the clauses. We must note here that we do not always need *all* of these lower level variables, indeed the variables $u_{1,\frac{n}{2},z}$ and $u_{\frac{n}{2}+1,n,z}$ with $z > (\frac{n}{2}+1)$ will trivially be false, since there can not be more than $(\frac{n}{2}+1)$ indices in the sets $U_{1,\frac{n}{2}}$ or $U_{\frac{n}{2}+1,n}$. In general, a variable $u_{x,y,z}$ with $z > (y-x+1)$ will always be trivially false. This brings the actual number of lower level to at most $2(\frac{n}{2}+1)$.

Generally, we need $min((n+2), 2(i+2))$ lower level variables. In the worst case, when i equals n, this is $(n+2)$.

Whereas the original definition of u_i used sets of size n, we are now left with sets of size $\frac{n}{2}$. We repeat this recursively until the sets are of size 1. The variables $u_{x,x,0}$ and $u_{x,x,1}$ then correspond to $\neg v_x$ and v_x respectively.

The 'left' variables $u_{1,\frac{n}{2},k}$ with different k will in their turn all use the same lower level variables $u_{1,\frac{n}{4},k'}$ and $u_{\frac{n}{4}+1,\frac{n}{2},l'}$. This is similar for the 'right' variables. The second iteration will thus introduce $2.(n/2+2)$ lower level variables. The total process will ultimately introduce the following number of lower level variables:

$$\begin{array}{rl} & n+2+2(n/2+2)+4(n/4+2)+\ldots+2^{k-1}(n/2^{k-1}+2) \quad \text{for } 2^k=n \\ = & n+2+\quad n+4\quad +\quad n+8\quad +\ldots+\qquad n+2n \end{array}$$

which equals

$$\begin{array}{rl} & (n+n+\cdots+n)+(2.2^0+2.2^1+2.2^2+\cdots+2.2^{k-1}) \\ & \qquad\qquad\qquad\qquad \text{with twice } k=log_2 n \text{ terms} \\ = & \quad nlog_2 n \quad + \quad \Sigma_{i=1}^{k}(2^i) \\ = & \quad nlog_2 n \quad + \quad 2n-2 \\ = & \quad O(nlogn+n) \\ = & \quad O(nlogn) \end{array}$$

The first iteration introduces $(i+2)$ lower level clauses. In the worst case i equals n, thus $(n+2)$ clauses are introduced. To count the total number of clauses, we first look at an example. When $n=7$, the variable $u_{1,7,7}$ is expressed using the variables $(u_{1,3,0}, u_{1,3,1}, u_{1,3,2}, u_{1,3,3})$ and $(u_{4,7,0}, u_{4,7,1}, u_{4,7,2}, u_{4,7,3}, u_{4,7,4})$.

The second iteration will introduce $(i+2)$ clauses for each of these variables $(i \in \{0, \ldots, \frac{n}{2}\})$, so in total $(3+4+5+6)+(3+4+5+6+7)$. In a formula, the set of k and l variables will produce the following number of clauses

$$\sum_{i=0}^{\lfloor n/2 \rfloor} (i+1) + \sum_{i=0}^{\lceil n/2 \rceil} (i+1)$$

which is $O(n^2)$.

Translation of total and perType constraints

For an employee p and numbering N_x, the total constraints impose restrictions on the minimum and maximum number of decision variables $v_{p,j}$ (for which N_x is defined) that can be assigned true in a given personal schedule S_p. Definition 11 to 13 can be rewritten:

Definition 14. *V is the set of variables $v_{p,i}$ for which the given numbering N_x is defined.*

For the following we assume that set V contains n elements.

Definition 15. *$U_{\alpha,\beta}$ is the set of indices (of time units) k between α and β for which v_k in V is assigned true in a given personal schedule S_p.*

Definition 16. *$F_{\alpha,\beta}$ is the set of indices (of time units) k between α and β for which v_k in V is assigned false in a given personal schedule S_p.*

More formally:

$$\begin{aligned} V &= \{v_{p,i} | i \in T_{N_x}\} \\ U_{\alpha,\beta} &= \{k | v_{p,k} \wedge k \in T_{N_x} \wedge (\alpha \leq k \leq \beta)\} \\ F_{\alpha,\beta} &= \{k | \neg v_{p,k} \wedge k \in T_{N_x} \wedge (\alpha \leq k \leq \beta)\} \end{aligned}$$

A $min\_total = min_t$ constraint can be translated to SAT by using the general procedure from the previous section to express:

$$u_{min_t} \Leftrightarrow |U_{1,n}| \geq min_t$$

A $max\_total = max_t$ constraint means at least $n - max_t$ variables should be assigned false. Analogously, this can be expressed by translating

$$f_{n-max_t} \Leftrightarrow |F_{1,n}| \geq n - max_t$$

The translation of pertType constraints is similar. The only difference is that the set of variables V is limited to those corresponding to a specific value j.

$$V = \{v_{p,i} | i \in T_{N_x} \wedge N_x(i) = j\}$$

4.3 Translation of generic constraints to MIP

We translate the nurse rostering problem consisting of the entire constraint set of Table 4.1 into a mixed integer program. The constraints follow the same structure as the constraints in the mathematical description in Section 3.3 but now with an individual numbering as starting point instead of the nurse rostering problem instance. Let P be the set of nurses and T the set of time units in the scheduling period. We introduce binary decision variables $v_{p,t}$ denoting whether nurse p is assigned a shift on time unit t.

$$\forall p \in P, \forall t \in T : v_{p,t} = \begin{cases} 1 \text{ if nurse } p \text{ works a shift on time unit } t \\ 0 \text{ otherwise} \end{cases} \tag{4.10}$$

Analogously to the preprocessing step in the SAT translation scheme in Section 4.2.1, we introduce auxiliary variables $t_{p,i}$ indicating for each sequence of consecutive time units with equal numbers whether an employee p is working on a time unit within that sequence. Generally, for an employee p and a sequence i starting at k and ending at l:

$$-(l-k+1)t_{p,i} + \sum_{j=k}^{l} v_{p,j} \leq 0 \quad \text{and} \quad -t_{p,i} + \sum_{j=k}^{l} v_{p,j} \geq 0 \tag{4.11}$$

In the following sections, to simplify the notation, we denote $v_{p,t}$ resp. $t_{p,j}$ as v_t and t_j.

Let C be the set of all constraints. Each $c \in C$ is of one of the eight generic constraint types. Each constraint c has a weight W^c associated. When violated, a constraint contributes to the objective function proportionally to the degree of violation, multiplied by the individual weight of the constraint. The goal is to minimize a sum of weighted constraint violations:

$$\text{MINIMIZE}\{\sum_{c \in C} W^c.V(c)\} \tag{4.12}$$

In the following section we show how to count constraint violations for each of the eight constraint types defined in Section 4.1.

4.3.1 Translation of consecutive constraints

Maximum number of consecutive events

Let CAS^{max_c} be the set of all contiguously ascending event sequences of length $max_c + 1$. Consider a contiguously ascending event sequence $cas \in CAS^{max_c}$. Every such sequence contains indices cas_i of the time units on which an event occurs. The constraint violations are calculated as follows:

$$V(max_c) = \sum_{cas \in CAS^{max_c}} max\{(\sum_{j=0}^{max_c} t_{cas_j}) - max_c, 0\} \tag{4.13}$$

Minimum number of consecutive events

To express this constraint type, we introduce the concept of a 'maximally contiguously ascending event sequence'. Consider t_k, respectively t_l, as the time unit associated with the first, respectively last event of the sequence $r = \{e_0, \ldots, e_{m-1}\}$.

Definition 17. *A sequence r is maximally contiguously ascending if no event e_i with number n_i on time unit t_i exists with $e_i = e_0 - 1$ and $t_i < t_k$ or with $e_i = e_{m-1} + 1$ and $t_i > t_l$*

For every maximally contiguously ascending event sequence r of length m:

$$V(min_c) = \sum_{j=0}^{m-1} (min_c.u_{j,0} - \sum_{k=1}^{min_c} u_{j,k}) \tag{4.14}$$

with

$$0 \leq u_{j,0} \leq t_{r_j} \text{ and } t_{r_{j+1}} - t_{r_j} \leq u_{j,0} \leq t_{r_{j+1}} \text{ and}$$
$$0 \leq u_{j,k} \leq t_{r_{k+1}} \text{ and } u_{j,k} \leq u_{j,k-1}$$

4.3.2 Translation of 'between' constraints

Maximum gap between events

Let CAS^{max_b} be the set of all contiguously ascending event sequences of length $max_b + 1$. Consider a contiguously ascending event sequence $cas \in CAS^{max_b}$.

Every such sequence contains indices cas_i of the time units on which an event occurs. The constraint violations are calculated as follows:

$$V(max_b) = \sum_{cas \in CAS^{max_b}} max\{(\sum_{j=0}^{max_b}(1 - t_{cas_j})) - max_c, 0\} \tag{4.15}$$

Minimum gap between events

For every maximally contiguously ascending event sequence r of length m:

$$V(min_c) = \sum_{j=0}^{m-1}(min_b.u_{j,0} - \sum_{k=1}^{min_b} u_{j,k}) \tag{4.16}$$

with

$$0 \leq u_{j,0} \leq t_{r_j} \text{ and } t_{r_j} - t_{r_{j+1}} \leq u_{j,0} \leq t_{r_{j+1}} \text{ and}$$
$$0 \leq u_{j,k} \leq 1 - t_{r_{k+1}} \text{ and } u_{j,k} \leq u_{j,k-1}$$

4.3.3 Translation of counting constraints

Maximum and minimum number of events

Let J be the set of indices of time units for which a numbering N_i is defined:

$$J = \{t \in T | N_i(t) \neq U\} \tag{4.17}$$

The constraint violations of a 'counting' constraint can be calculated as follows:

$$V(max_t) = max\{\sum_{j \in J} v_j - max_t, 0\} \tag{4.18}$$

$$V(min_t) = max\{min_t - \sum_{j \in J} v_j, 0\} \tag{4.19}$$

Note that we are using variables v_j instead of variables t_j as opposed to the previously discussed constraint types. The specific value of the numbering plays no role in the evaluation of counting constraints. Only the fact that a numbering is defined determines when a time unit needs to be taken into account.

Maximum and minimum number of events per type

The main difference with the previous constraint type is that, for evaluating this constraint, the specific value of a numbering does play a role. We still use the variables v_j. The limits are set on the number of events 'having the same number n'. Let J_n be the set of indices of time units for which a numbering N_i has value n:

$$J_n = \{t \in T | N_i(t) = n\} \tag{4.20}$$

For a numbering N_i, the constraint violations are calculated as follows:

$$V(max_{pt}) = \sum_{n \in N_i} max\{\sum_{j \in J_n} v_j - max_{pt}, 0\} \tag{4.21}$$

$$V(min_{pt}) = \sum_{n \in N_i} max\{min_{pt} - \sum_{j \in J_n} v_j, 0\} \tag{4.22}$$

4.4 Experiment

We translated the instances of the "First International Nurse Rostering Competition 2010" (Chapter 5) to SAT using the above presented translation scheme. The competition instances come in three categories:

- sprint: 10 employees, allowed to run for 10 seconds
- medium: 30 employees, allowed to run for 10 minutes
- long: 50 employees, allowed to run for 10 hours

	Instance	Time (msec)	Nb. vars.	Nb. clauses
sprint	early01	2535	79741	489530
	early02	1509	79681	488554
	early03	1534	79705	488676
	early04	1528	79521	486730
	early05	1775	79681	488540
	early06	1605	79613	487070
	early07	1514	79621	487744
	early08	1429	79485	484898
	early09	1624	79681	488588
	early10	1502	79673	488064
	late01	2243	82103	491873
	late02	625	58666	301847
	late03	1835	82119	491969
	late04	1487	82119	491994
	late05	1509	81995	491300
	late06	1442	78185	481400
	late07	1400	78241	481438
	late08	1372	78092	480933
	late09	1525	78288	481423
	late10	1417	78293	481998
	hidden01	1363	58365	300892
	hidden02	648	58357	300716
	hidden03	1687	82202	493117
	hidden04	1471	79530	486105
	hidden05	1499	82180	491778
	hidden06	621	58458	301207
	hidden07	650	58471	301121
	hidden08	1957	82287	493468
	hidden09	1710	79639	486547
	hidden10	1542	82291	492229

Table 4.4: Conversion of sprint instances to SAT

	Instance	Time (msec)	Nb. vars.	Nb. clauses
medium	early01	4974	264895	1743766
	early02	4274	265375	1752231
	early03	4040	265375	1751949
	early04	3641	264627	1742319
	early05	4083	265453	1756798
	late01	2749	265917	1706898
	late02	2624	265669	1702051
	late03	2475	256481	1689016
	late04	3318	264969	1705168
	late05	5439	388455	2450568
	hidden01	7088	340822	2445764
	hidden02	5176	340982	2447166
	hidden03	5332	340570	2445260
	hidden04	5426	340982	2447116
	hidden05	5254	339926	2443396

Table 4.5: Conversion of medium instances to SAT

As expected, all instances turned out to be unsatisfiable. At least two conflicting constraints remain and a zero cost cannot be obtained. In Tables 4.4 to 4.6, we show for each instance the time needed to convert the instance and the dimensions, number of variables and number of clauses, of the corresponding SAT problem. As expected, the running time increases as the problem size increases. The difference in calculation time within one category can be explained by the number of counting constraints present within an instance as the counting constraints types (total and pert) are the most time consuming constraints to convert.

	Instance	Time (msec)	Nb. vars.	Nb. clauses
long	early01	19876	560821	4347683
	early02	19096	562089	4370894
	early03	19003	562159	4374424
	early04	19147	560885	4364509
	early05	18875	561151	4358326
	late01	7640	594764	4518776
	late02	7899	594764	4518984
	late03	7575	594796	4518984
	late04	7643	594796	4519072
	late05	7644	592472	4506166
	hidden01	12413	595598	4519059
	hidden02	8008	595598	4519059
	hidden03	7715	596428	4514030
	hidden04	7786	594934	4515035
	hidden05	7863	592108	4513510

Table 4.6: Conversion of long instances to SAT

4.5 Conclusion

The main contribution of this chapter is automated translation of nurse rostering problem instances into SAT problems and MIP instances. To support automated translation, a formal description of nurse rostering problem instances is required. We utilised numberings (Burke et al., 2001) for clearly and unambiguously describing instances. We defined eight generic constraint types allowing representation of a large number of real world inspired constraints found in literature. For each constraint type, we presented efficient translation schemes to SAT and MIP.

The preprocessing step for both the SAT and MIP translation introduces $O(n)$ variables and clauses. The SAT translation of the *consecutiveness* and *between* constraints generates $O(n)$ clauses. The SAT translation of *counting* constraints generates $O(n^2)$ clauses and $O(nlogn)$ variables. Summarized, all constraints that can be expressed as monotonic ascending numberings can be translated to SAT and MIP.

The automated translation allows for a fast and error safe representation of nurse rostering problem instances as SAT and MIP models. The translation of the instances of the "First International Nurse Rostering Competition 2010" (Chapter 5) to SAT took at most 20 seconds. The translation of the smaller competition instances is performed in less than 2 seconds. This removes the need for a manual translation which would be very time consuming while complete equivalence between the original and the translated model may be hard to attain.

This approach allows studying problems from an alternative point of view. As stated in the introduction, the SAT translation scheme was successfully used to study the hardness of nurse rostering problem instances in (Bilgin et al., 2009). Aim of a hardness study is to identify a set of problem features to be used for predicting the behaviour of some algorithm for a specific performance indicator

on a particular instance. The SAT translation scheme allowed the use of a set of well known SAT features (Nudelman et al., 2004) to successfully predict the solution quality obtained by the winner algorithm of the nurse rostering competition on a large set of randomised competition instances. Although the aims of the algorithms for which the SAT feature set was developed is very different from the aims of algorithms within the field of nurse rostering research (satisfiability versus optimality), the relevance of SAT features for the hardness analysis of nurse rostering problem instances was demonstrated.

As an extension, the SAT translation scheme can be adapted to produce MAXSAT instances thereby 'incorporating information on the objective function'. The extended scheme could be used to apply MAXSAT solvers to nurse rostering problem instances. The MIP translation scheme could provide lower boundaries for problem instances. Glass and Knight (2010) use a manual approach to identify lower bounds for some of the smaller instances found in Brucker et al. (2010) via an MIP model. An automated approach may be suitable for larger instances. Furthermore, the MIP translation scheme allows the development of a modeling tool to express nurse rostering problems as MIP models.

Chapter 5

The First International Nurse Rostering Competition 2010

5.1 Introduction

The First International Nurse Rostering Competition (INRC2010) aimed at developing interest in the general area of rostering and timetabling, while providing researchers with models of the problems faced incorporating a significant number of real world constraints. The idea of organising a nurse rostering competition derives from the interest aroused by the two timetabling competitions, ITC2002 and ITC2007 (McCollum et al., 2009). Most of the rules of INRC2010 are imported from ITC2007, with some adjustments obtained from the lessons learnt.

Similarly to ITC2002 and ITC2007, the main objective of INRC2010 was to generate new approaches to the associated problem by attracting researchers from different areas of research. As with many cases in the past, significant advancements have been made in research areas by attracting multi-disciplinary approaches and comparing them on a common ground.

The second objective was to reduce the gap between research and practice within this important area of operations research. Although for the sake of the competitive element, we did not include all aspects of all 'real-world' problems but aimed to introduce significant depth and complexity.

The third objective of INRC2010 was to further stimulate debate within the widening rostering and timetabling research community.

The competition was composed of three tracks; named after the Olympic disciplines. As the features of the algorithm are often tuned to the available running time, the three tracks represented different challenges to the participants:

Sprint: Required a solution in a few seconds, typical for interactive use.

Middle Distance: Required the solution in a few minutes and simulated the practical situation in which the problem has to be solved a few times during a solving session. During a solving session, a nurse responsible for the rosters typically studies the effect of several possible assignments of shifts to nurses in order to find a good quality roster.

Long Distance: Granted the solver many hours of running time and simulated overnight solving.

This chapter describes the INRC2010, the format, and the results. In Section 5.2 we list the competitions rules, whereas in Section 5.3 we describe the benchmarking procedure. Section 5.4 gives details on the competition instances and the evaluation of instances. A description of how the solutions were ranked is given in Section 5.5. Results for the different tracks are given in Section 5.6. In the end, Section 5.7 contains a short discussion and the conclusions.

The research reported on in this chapter appeared in an adapted version as *S. Haspeslagh, P. De Causmaecker, A. Schaerf and M. Stølevik, The first international nurse rostering competition 2010, Annals of Operations Research, 2012, online first.*

5.2 Competition rules

To ensure a fair competition, a number of rules were given. A full listing of these rules can be found in Appendix D. We give a summary of the rules below.

The competition obviously included a set of instances upon which the solvers of the participants have been evaluated. For each track, a difference set of instances has been designed: smaller instances have been used for the Sprint Track, whereas the larger ones have been included in the Long Distance Track.

Following the same scheme experimented (successfully) for ITC2007, each of the three sets has been partitioned again into three subsets, called *early*, *late*, and *hidden* instances. The early instances have been released at the official start on the competition (March 1, 2010). The late instances have been made

public on May 15, 2010, roughly one month before the deadline set to June 20, 2010.

For all early and late instances, the participants were asked to submit their best solutions, according to the formulation provided in Section 5.4 and within the given time frame.

The hidden instances have been keep secret till the final adjudication, but have been used by the organizers to evaluate the participants' algorithms (Section 5.5).

The time granted to the solver was set using a benchmarking tool made available from the competition website (Section 5.3). The programs were allowed to run on a single processor PC only. Participants could code their solvers using any programming language; similarly to ITC2007 the use of third party software was allowed only under the following conditions: it should be free software, it's behaviour should be (reasonably well) documented and it should run under a commonly-used operating system (Unix/Linux, Windows, or Mac OS).

The input data of the solver was supplied through an instance file in the described format (Sect. 5.4) and the solution had to be delivered by means of a solution file in its predefined format.

The same version of the algorithm had to be used for all instances. This means that the solver could analyse the instance features and set the parameters accordingly, but it could not "recognise" the particular instance and use a parameter setting previously computed for that specific instance.

The algorithm could be either deterministic or stochastic. In both cases, participants had to be prepared to show that these results were reproducible in the given time frame. In particular, the participants that used a stochastic algorithm should have coded their program in such a way that the exact run that produced each solution submitted could be repeated simply. In other words, they could try several runs with different seeds to find the best solution to submit. As full repeatability was required, for stochastic algorithms, the random seed was required and entered as a command line parameter.

5.3 Benchmarking

The aim of providing a benchmarking tool was to eliminate the bias introduced by the differences in computational power of the participants' computers. The benchmark program tells each participant how long he/she could run his/her

solver on each instance on his/her computer. It is designed to test how fast the participants' machines are at doing the sort of operations similar to those that are involved in the solution of rostering problems.

Only software running on a single processor was accepted for the competition. Consequently, the benchmark code was only suitable for them. It was not suitable for multi-core machines or clusters, which could still be used but running the software (and the benchmark) separately on a single core/node.

On a relatively modern PC the benchmark program granted the participant approximately 10 seconds for the Sprint Track, 10 minutes for the Middle Distance Track, and 10 hours for the Long Distance Track.

The benchmark used was the C version of the publicly available popular "The Fhourstones Benchmark 3.1."[1] We modified the source code by hard-coding all input data (calibrated for our purposes), and writing the corresponding output. The benchmark was compiled with GNU C/C++ compiler under Ubuntu Linux, Windows XP (using cygwin) and Mac OS. The modified source code is available on request.

It is clear that it is not possible to provide a benchmark that is perfectly equitable across many platforms and algorithms. In any case, the programs of the finalists were run on our machine, thus creating a *level playing field* for the final round.

5.4 Problem description, competition instances and evaluation of solutions

The nurse rostering problem considered within this competition is described in Section 3.1. Next to this informal description, competitors were provided with a formal description using numberings (Section 3.2 and Appendix B).

The instances were generated from scratch. They incorporate a large set of real world constraints, of which many are commonly found in literature (Burke et al., 2004). The scheduling period is four weeks for each instance. The Sprint instances are limited to one skill. For the other tracks, two different skills are used. The Sprint instances count four different shift types, the Medium instances four or five and the Late instances five. Further, the instances per track mainly differ in the number of available nurses. For the Sprint instances ten nurses are available, for the medium instances 30 and for the long instances 51. The demand was generated accordingly. We tried solving the

[1] `http://homepages.cwi.nl/~tromp/c4/fhour.html`)

instances using CPLEX (using the mixed integer program translation scheme of Section 4.3). We allowed CPLEX to run for a multiple of the allowed runtime to ensure that solving the instances sets a real challenge.

We developed both an XML and text-only data format for representing the competition instances and solutions to the instances. Detailed information on the structure of both data formats can be found in Appendix E. A small example instance and a solution are also provided, in both XML and text. We provided conversion tools to translate 'text-only' instances and solutions to 'structured' ones and vice versa.

We implemented both an offline and online version of an extended evaluation procedure (Appendix A) based on the numbering method developed by Burke et al. (2001). Competitors were required to submit their solutions on the competition instances through the online evaluator. The value of the objective function, as reported by the online evaluator, was the official value that was taken into account when evaluating the results of competitors. This way, we aimed to avoid any dispute on the exact results obtained by competitors due to, for example, misinterpretations or faulty implementations of the objective function. Moreover, the use of numberings for a clear and unambiguous representation of the objective function, allowed competitors to securely and error-free build fully equivalent proprietary evaluators.

5.5 Solution Ranking

The competition was run in two steps: a preliminary round and a final. We first describe the preliminary round, in which the finalists were selected, and afterwards we describe how the final was run.

5.5.1 Preliminary Round

The preliminary round considered the solutions of the algorithms for early and late instances. Its aim was to select the *finalists*, that is the participants that entered into the final.

The selection criterion was based on solution *ranks*, similar to the one used for ITC2007 and described in (McCollum et al., 2009, Section 5). We report it here for the sake of selfcontainedness. Differently to ITC2007, no hard constraint violation is allowed, so the ranking is solely based on the number of soft constraint violations.

The selection worked as follows: Let m be the total number of early and late instances and k be the number of participants that produced a feasible solution for all m instances. Let X_{ij} be the result supplied (and verified) by participant i for instance j. Each X_{ij} is the value of the objective function s – the total penalty of breaking the soft constraints.

The matrix X of results is transformed into a matrix of rank R assigning to each X_{ij} a value R_{ij} from 1 to k. That is, for instance j the supplied X_{1j}, X_{2j}, ...,X_{kj} are compared with each other and the rank 1 is assigned to the smallest observed value, the rank 2 to the second smallest, and so on to the rank k, which is assigned to the largest value for instance i. Ranks are assigned for all the instances. We use average ranks in case of ties.

Table 5.1 is an example with $m = 6$ instances and $k = 7$ participants. The corresponding ranks are shown in Table 5.2, with the mean average rank in the last column.

Instance	1	2	3	4	5	6
Solver 1	34	35	42	32	10	12
Solver 2	32	24	44	33	13	15
Solver 3	33	36	30	12	10	17
Solver 4	36	32	46	32	12	13
Solver 5	37	30	43	29	9	4
Solver 6	68	29	41	55	10	5
Solver 7	36	30	43	58	10	4

Table 5.1: Example of results (the matrix X)

Instance	1	2	3	4	5	6	Avg.
Solver 1	3	6	3	3.5	3.5	4	3.83
Solver 2	1	1	6	5	7	6	4.33
Solver 3	2	7	1	1	3.5	7	3.58
Solver 4	4.5	5	7	3.5	6	5	5.17
Solver 5	6	3.5	4.5	2	1	1.5	3.08
Solver 6	7	2	2	6	3.5	3	3.92
Solver 7	4.5	3.5	4.5	7	3.5	1.5	4.08

Table 5.2: Example of solution ranks (the matrix R) plus the average rank

For each solver we computed the mean of the ranks. The finalists of the competition were the five solvers with the lowest mean ranks. In case of a tie for entering the last position in the finals, all the solvers with equal mean would have been included in the final (in this case the finalists would have been with more than five).

In the example, the finalists would have been solvers 1, 3, 5, 6, and 7. We checked the runs of the finalist with the submitted seed to make sure that the submitted runs were repeatable within the granted time according to the benchmarking tool.

5.5.2 Final

For the final, the same evaluation process was repeated for the five finalists with the following differences:

1. The solvers were run by us, the organisers, on our machines, with the support of the finalist for compiling and running their solvers.
2. All instances, including hidden ones, were used for ranking.
3. For each instance, we ran a set of independent replications with seeds chosen at random. For each instance, we computed the ranks on the pool of all replications, and averaged them on all instances.
4. The number of replications used was set to 10 for sprint and medium track and 4 for the long track (because of the long computing time).

The winner was the participant with the lowest mean rank. In case of a tie, which did not happen, one further replication at a time would have been added for all instances until a single winner would have been found.

Note that the process described above was possible because all finalists used stochastic solvers. In case of a deterministic solver, we would have used the single value produced by the solver as the score of all replicates. Furthermore, in case of a tie between two deterministic solvers, we would have added one extra instance at a time, instead of running one more replicate on the given instances.

5.6 Competition Tracks and Results

As already mentioned, the competition was composed of three tracks; called after the Olympic disciplines, *1. Sprint*, *2. Middle Distance*, and *3. Long Distance*. The tracks differed from each other based on the maximum running times and on the size of the instances, whereas the problem formulation was the same throughout the competition. These tracks represent distinct problem settings in practice.

Overall, there were 15 participants. Not all participants contributed to all tracks. Various techniques were applied. A significant number of competitors implemented metaheuristic-based algorithms such as tabu search, simulated annealing, genetic and evolutionary algorithms. Some competitors designed hybrid solvers, combining for example a genetic algorithm with tabu search or a branch and price algorithm with variable depth search. Other competitors modeled the nurse rostering problem as a constraint optimisation problem, a multilevel assignment problem or answer set programming and other applied state-of-the-art solvers. Some competitors proposed multiple phase approaches. One competitor designed a hyperheuristic-based solver. In the following sections we discuss the results and the solvers of the finalists. For a full list of competitors, we refer to the competition website (INRC2010).

5.6.1 Sprint track

The Sprint track required a solution within a time frame of about ten seconds, which is typical for interactive use. The 30 instances of this track consisted of ten nurses. The planning horizon was 28 days. Table 5.3 shows the ranking of the finalists for this track.

The winning team - C. Valouxis et al. - produced a best result for 23 out of the 30 instances. Their method partitions the original problem into sub-problems of computationally manageable size. The sub-problems are solved using Mathematical Programming. They implemented a two phase strategy. The first phase assigns nurses to working days. In the second phase, nurses assigned to a day are scheduled to one of the required shifts on that day.

K. Nonobe and the team of E.K. Burke and T. Curtois both produced 20 best results. K. Nonobe reformulated the problem instances as a Constraint Optimisation Problem (COP). A powerful, general-purpose COP solver is used. The main advantage of this approach is that not much time has to be spent for developing and implementing algorithms.

E.K. Burke and T. Curtois converted the problem instances to their staff rostering problem formulation and used an ejection chain-based variable depth search algorithm as the solution method.

Zhipeng L. and Jin-Kao Hao produced 12 best results. They implemented an algorithm that switches between a local search procedure - starting from a randomly generated feasible solution - and an elite solution restart mechanism.

The team consisting of B. Bilgin et al. produced six best results. Their method is based on a hyper-heuristic approach combined with a greedy shuffle heuristic.

For 80 % of the time, a state-of-the-art hyper heuristic is applied. The greedy shuffle heuristic consumes the remaining time.

A complete list of best results and abstracts of all competitors describing their techniques and approaches can be found at the competition website (INRC2010). A summary with the results for this track is in Table 5.4. The fact that Burke et al. lost on the hidden instances must be traced back to a technical problem in their implementation of the model and does not allow concluding on their method. Their dominance on early and late instances seems to indicate that their sophisticated and mature solver actually outperforms the new developments of the other competitors. Nonobe et al. and Valouxis et al. are seen to be competitive. Worthwhile to mention here is that, on the hidden instances, Valouxis et al. is exclusively best on three, Nonobe et al. on two and Bilgin et al. on one instance.

Competitor	Rank
C. Valouxis, C. Gogos, G. Goulas, P. Alefragis, E. Housos	2,08
K. Nonobe	2,45
Zhipeng L. and Jin-Kao Hao	3,10
E.K. Burke and T. Curtois	3,30
B. Bilgin, P. Demeester, M. Misir, W. Vancroonenburg, G. Vanden Berghe, T. Wauters	4,07

Table 5.3: Competitor ranking for the Sprint Track

Competitor	Early (10)	Late (10)	Hidden (10)	Total (30)
Valouxis et al.	10	7	6	23
Nonobe	10	4	6	20
Burke et al.	10	10	0	20
Zhipeng et al.	9	3	0	12
Bilgin et al.	4	0	2	12

Table 5.4: Summary of results; the number of best solutions for the Sprint Track

5.6.2 Middle Distance track

The Middle Distance track required a solution in a few minutes and mimicked the practical situation in which the problem has to be solved a few times in a solving session. The 15 instances of this track consisted of 31 nurses. The planning horizon was 28 days. Table 5.5 shows the ranking of the finalists for this track.

The winning team - C. Valouxis et al. - produced a best results for eight of the fifteen instances using the same method as for the Sprint Track.

Runners up, E.K. Burke and T. Curtois, obtained ten best results. They implemented a branch and price algorithm. The pricing problem is solved using the same ejection chain method as implemented for the Sprint track.

K. Nonobe found three best results using the COP approach described above. For one instance, Zhipeng L. and Jin-Kao Hao, found a best result.

Table 5.6 displays similar results for the middle distance track as for the sprint track. Again Burke et al. suffer from the implementation problem. Valouxis et al. and Nonobe are seen to dominate the game.

Competitor	Rank
C. Valouxis, C. Gogos, G. Goulas, P. Alefragis, E. Housos	1,77
E.K. Burke and T. Curtois	2,27
K. Nonobe	2,30
Zhipeng L. and Jin-Kao Hao	3,67
B. Bilgin, P. Demeester, M. Misir, W. Vancroonenburg, G. Vanden Berghe, T. Wauters	5,00

Table 5.5: Competitor ranking for the Middle Distance Track

Competitor	Early (5)	Late (5)	Hidden (5)	Total (15)
Burke et al.	5	5	0	10
Valouxis et al.	5	0	3	8
Nonobe	2	0	2	4
Zhipeng et al.	1	0	0	1
Bilgin et al.	0	0	0	0

Table 5.6: Summary of results; the number of best solutions for the Medium Track

5.6.3 Long Distance track

The Long Distance track granted the solver about ten hours of running time and simulated overnight solving. The fifteen instances of this track consisted of 50 nurses. The planning horizon was 28 days.

Table 5.7 shows the ranking of the finalists for this track. The winning team of C. Valouxis et al. and the team of E.K. Burke and T. Curtois found a best

result for ten of the fifteen instances. Both teams used the same methods as for the Middle Distance track.

B. Bilgin et al. obtained a best result for five instances using their hyper-heuristic approach. The COP-method of K. Nonobe found four best results.

D. Rizzato et al. model the problem as a Multilevel Assignment Problem and implemented a two phase algorithm consisting of a construction and an improvement phase. Similar remarks hold as for the other tracks, but Valouxis et al. have taken the lead.

Table 5.8 shows that Rizato et al. (who took over from Zhipeng et al. in the top five) did not produce any best results but realized a good overall performance. Most competitors reported having a hard time to use the full ten hours of running time. The algorithms seem to arrive at their final results after 60 to 120 minutes and are not able to improve afterwards.

Competitor	Rank
C. Valouxis, C. Gogos, G. Goulas, P. Alefragis, E. Housos	1,93
E.K. Burke and T. Curtois	2,27
B. Bilgin, P. Demeester, M. Misir, W. Vancroonenburg, G. Vanden Berghe, T. Wauters	2,60
K. Nonobe	3,73
D. Rizzato, A. Constantino, E. Luiz de Melo, D. Landa-Silva, W. Romeo	4,47

Table 5.7: Competitor ranking for the Long Distance Track

Competitor	Early (5)	Late (5)	Hidden (5)	Total (15)
Burke et al.	5	5	0	10
Valouxis et al.	5	1	4	10
Nonobe	4	0	0	4
Bilgin et al.	4	0	1	5
Rizzato et al.	0	0	0	0

Table 5.8: Summary of results; the number of best solutions for the Long Track

5.6.4 Lessons learned

The results of the competition may not allow drawing final conclusions but do contain important hints.

- **Dedication helps.** The solver of Burke et al., using an ejection chain based variable depth search, was developed for their staff rostering problem to which the competition problems could be mapped. This solver performed best on the early and late instances for all tracks. The method seems to be well fit for these problems.

- **Hybridization and decomposition is effective.** The all track winner of Valouxis et al. is a heuristic in combination with mathematical programming. It decomposes the problem in manageable bits which can be solved to optimality. It is furthermore a two stage approach.

- **Modeling power is important.** The COP solver used by Nonobe et al. allows the developer to concentrate on the model which in the nurse rostering problem case is particularly complicated. The fact that they ended up so high, especially in the sprint and middle distance tracks, demonstrates the power of their solver and modeling technique. The same can be said about the approach of Rizzato et al. who used a multi-level assignment problem solver and managed to end up fifth in the long distance track.

- **Local search and restart.** The approach followed by Zhipeng en al. combining a local search method and an intelligent restart mechanism, entered the top five in the two shorter tracks. This is an interesting demonstration of how these goal function driven methods allow to tackle hard to model and difficult optimisation problems in an acceptable way. It looks as if, given more time for refinement, the method has the potential to produce top solutions.

- **Hyperheuristics work.** The results of Bilgin et al. illustrate the potential of hyperheuristic approaches. They seem to climb up in the ranking when more time is given. This may indicate that the layered approach with a high level algorithm steering low level heuristics is not very reactive. However, it can be used in situations where sufficient time is given to learn about the behaviour of the low level heuristics on a particular problem instance.

- **Time is not used proportionally.** The long distance track allowed solvers to take ten hours. No one was able to improve after two hours. This could mean that at some point a plateau is reached which requires extensive search to find further improvements or that the algorithms simply reached their limits in the kind of complexity they were able to grasp: the algorithm might be stuck at some local optimum and fails to escape it. An observation supporting the latter is that the results obtained are not uniform.

5.7 Conclusion

We presented the results and detailed information about the First International Nurse Rostering Competition 2010 (INRC2010). The competition rules and set-up have been discussed. The Nurse Rostering Problem instances considered in the competition have been illustrated. The results are presented and the evaluation procedure has been described.

It should be noted that the first goal of the competition, to generate new approaches, ideas and insights to the problem, has been achieved. For example, there are contributions from the Constraint Programming community with some techniques that have a high emphasis on modelling the problem. We observed that indeed, modelling is an important issue within this area of research. In particular, worse behaviour in the hidden instances and eventually a resulting worsening of the final ranking could be traced back to a quite trivial modelling error. Accurate modelling is essential in a complex problem domain as nurse rostering. The hidden instances protect against modelling errors as well as against the so-called Mongolian Horde approach (Schaerf and Di Gaspero, 2007): "run as many trials as you can with no time limit and report only the best of all of them".

Furthermore, we can conclude that we faced similar problems as the organisers of ITC2007 McCollum et al. (2009) did. We also found it hard to provide an unambiguous problem formulation. Therefore, following the directions given by Schaerf and Di Gaspero (2007), beside a natural language description, we used numberings to clearly describe the problem constraints and objectives. We provided solution validators to help detect ambiguities and to help competitors to correctly model the problem. A second problem was that submissions slightly differed with respect to the required command-line arguments. It was quite time consuming to adjust our evaluation procedure to reproduce the results and to run the algorithms on the hidden instances. A clear and precise description of the required interface would have allowed more automation for the evaluation of the submissions.

Particularly hard in the organisation was coming up with an appropriate set of problem instances. We detected a problem with the Early instances as they appeared to be easily solvable. Therefore we translated the instances to Mixed Integer Programs and tried to solve them with CPLEX. As expected, CPLEX was able to solve the instances to optimality within the required time frame. For the late and hidden instances, we considered an instance a challenge if CPLEX failed to solve the instance within a multiple of the allowed runtime for solving the instance. This highlights the need for complexity measures in this domain. All problem instances and currently best found results can be found on the

competition website (INRC2010). It is our belief that similar benchmark sets can help to improve and to stimulate further research and debate within and beyond the rostering and timetabling community.

Chapter 6

Negotiation protocols for short term nurse rostering

The purpose of this chapter is to study the possibilities of a distributed approach in the case of nurse rostering problems. We envisage a system consisting of units (wards) with a large degree of autonomy and highly detailed local decision making. The independence of these units is limited by the sharing of resources and by the opportunities or obligations for inter unit assistance. This assistance may increase the efficiency of one unit while only slightly hampering the performance of another one. On the long term, the performance of the larger system should benefit. So, in its simplest form, we must consider two levels of decision making which we label local and global. The local system takes the highest level of detail into account. We are especially interested in the case where the local decision problem is of a combinatorial complexity. Various local details are aggregated and translated into variables that can be handled at the global level, which, at the same time, introduce supplementary requirements, constraints and preferences. The decision problem at this global level may be of a combinatorial complexity as well.

A system consisting of autonomous units naturally leads to an agent based model. Such models have been intensively studied over the last decade, e.g. by Valckenaers et al. (2006). Rather recently the link with optimisation has obtained some attention. Shen et al. (2006) give an updated review in the field of intelligent manufacturing. We think that the optimising behaviour of an agent based system should be studied as close to the real world problems as possible. We study the behaviour of a system of negotiating agents for a distributed nurse rostering problem. We select several negotiation protocols for

the global problem and an optimisation algorithm from literature (Burke et al., 2008) for the local problem. Our ultimate aim is to demonstrate the feasibility of such an approach.

At the level of an individual ward, the criteria are those that have been analysed in Chapter 3. At the hospital level other criteria are to be considered. Here, the manager wants to guard the fairness of the work load distribution, wants to raise certain quality levels, wants to minimize the personnel cost, wants to decide where resource shortages are allowable at peak moments ...

As an example of a local problem that may be resolved at the global level, we consider the cases of sudden staff shortages due to absence or unexpected overload. These may be hard to deal with at the local level where a longer term schedule should not be disturbed too drastically, while chances are higher that a peer ward accidentally has some spare capacity at or about this specific moment in time.

For solving shortages, Moz and Vaz Pato (2003) try altering the schedules of other nurses. The changes to the rosters should be minimal and, analogously to the original nurse rostering problem, may not be in conflict with legal, organisational and contractual regulations. The problem is called the nurse rerostering problem and is studied at a hospital in Lisbon. The problem is formulated as an integer multi commodity flow problem with additional constraints. The authors applied a heuristic to solve the problem and solved an integer linear programming formulation of the model using CPLEX. In follow-up papers, a genetic algorithm approach (Moz and Vaz Pato, 2007) and a utopic Pareto genetic heuristic (Vaz Pato and Moz, 2008) are applied to the same problem.

Maenhout and Vanhoucke (2011) tackle the rerostering problem using an evolutionary approach. A benchmark dataset for the rerostering problem is introduced. The set is based on the artificial instances of NSPLib. Similarly as for NSPLib, a set of schedule disruption characteristics is used to generate the rerostering problem instances.

Bard and Purnomo (2006) consider both altering the schedules of a subset of nurses and hiring 'traveling nurses', identified as the *nurse addition problem*, as a solution for shortages. They studied the problem for hospitals in the UK and Great Britain where the schedules of nurses are fixed by contract and therefore difficult to change. A column generation method is proposed for the 'alternation problem'. A branch and price algorithm is applied to the nurse addition problem.

Section 6.1 elaborates on the framework for negotiation: we discuss the local nurse rostering problem at a ward, a solution method for that problem and we

identify the subjects of the negotiation. In Section 6.2 we describe some details on the five negotiation protocols we consider within this chapter. Section 6.3 describes experiments and results. A conclusion is given in Section 6.4.

This Chapter is a summary of the following two papers:

- *S. Haspeslagh, P. De Causmaecker and G. Vanden Berghe , Distributed Decision Making in Hospital Wide Nurse Rostering Problems, in Proceedings of the 3th Multidisciplinary International Scheduling Conference: Theory & Applications, Paris, France, 2007*
- *R. Lagatie, S. Haspeslagh and P. De Causmaecker, Negotiation protocols in Distributed Nurse Rostering, in Proceedings of the* 21^{st} *Benelux Conference on Artificial Intelligence, Eindhoven, the Netherlands, 2009*

6.1 Framework for negotiation

We consider a hospital with n wards. Each ward is responsible for its personnel rosters. When shortages arise, when the quality of rosters drops below a certain quality level or when the good operation of its tasks is jeopardised, wards start a negotiation process to increase their level of efficiency by exchanging nurses. The wards are modelled as agents. The agents are self-interested, each agents want to optimise its rosters, but collaborative, agents agree on slightly worsened rosters if the global cost at the level of the hospital is reduced. The agents have the following tasks:

- Constructing the rosters of its ward
- Identifying local problems such as personnel shortages
- Negotiating with other agents for outsourcing problematic shifts

We first discuss the algorithm used to construct the local working rosters that are the starting points of the negotiation process (Section 6.1.1). The same algorithm will be used to determine the value of proposals during negotiation. Next, in Section 6.1.2, we elaborate on the method used to identify the shifts that are the subjects of negotiation.

6.1.1 Constructing local rosters and evaluation of offers

For the construction of the local rosters, we implemented a simplified version of the algorithm proposed by Burke et al. (2008). We chose this algorithm

because, at the time of experimenting, it was a state-of-the-art algorithm performing well on the targeted benchmark instances. The algorithm consists of three phases. First, a feasible roster is constructed. The algorithm starts by a heuristic ordering of the shifts. Idea is to assign 'difficult' shifts first to nurses for which the assignment results in the largest improvement or the smallest worsening. For more information on the heuristic ordering we refer to the original paper (Burke et al., 2008). As a result of this first phase, a roster is constructed where every shift is assigned to a nurse having the required skills for that shift type. When any shift is left unassigned, the roster is infeasible and there is a shortage of (qualified) staff.

The second phase of the algorithm aims at improving the quality of the roster using a Variable Neighbourhood Search (VNS) algorithm. Two commonly used neighbourhoods are considered, defined by the following moves:

1. Assign a shift to a different nurse.
2. Swap the assignments of two shifts between the originally assigned nurses.

The first neighbourhood is explored until no improving moves are found[1]. Then, the second neighbourhood is searched for an improving move. If a move is found, the first neighbourhood is examined again after applying the move. As the second neighbourhood is small, it is searched exhaustively. The algorithm stops when no improving move has been found in either neighbourhood. After a successful move, the algorithm attempts to assign any of the unassigned shifts.

The third phase, omitted in our implementation, unassigns all shifts of a fixed number of nurses with the worst personnel rosters. These shifts are then reassigned using the above described VNS. This "schedule disruption and repair" phase aims at escaping local optimal solutions. Instead of this phase, in our setting, a negotiation for exchanging these shifts is initiated.

6.1.2 Subject of negotiation

After the initial rosters have been constructed, the wards have to determine a set of 'problematic shifts' that will be the subject of negotiation. We consider two possibilities. As stated in the previous section, it may be impossible to assign each shift to a nurse during the heuristic ordering initial roster construction phase. Wards negotiate hoping to acquire extra personnel to

[1] For large problems, this neighbourhood may become too large to be searched exhaustively. In this case, a time limit should be set. The instances used for the experiments do not require a time limit.

cover these unassigned shifts. Secondly, the levels of efficiency of the wards may increase by assigning shifts that are 'hard to schedule' for a ward to external nurses. Penalties are often raised because of a particular sequence of shifts that have been assigned. Ideally, we would like to negotiate about the shifts that, when removed from those sequence, result in resolving the constraint violations caused by that shift. However, this relation between individual shifts within a specific sequence and the penalties raised because of the occurrence of those shifts within that sequence is still hard to identify. Furthermore, removing a shift from one sequence may result in new violations in other sequences. Therefore, we limit the search for suitable shifts to searching the shift that, when unassigned, introduces the best overall improvement. When found, the shift is unassigned and added to the list of negotiable shifts. The search is repeated until such a shift cannot be found. The number of unassigned shifts is limited to a percentage of the total number of shifts. Experiments showed that a value of 10 percent suits best in our test cases.

6.2 Negotiation protocols

We apply five different negotiation protocols to the above stated negotiation problem. In each of the proposed protocols, agents make bids to acquire extra shifts. An agent uses the same algorithm as for the construction of the initial rosters to estimate the cost or gain of adding an extra shift. In case the addition of a shift raises a cost, a bid for that shift is only accepted if the global penalty (the sum of the penalty of the individual rosters of the wards) is reduced. Dependent on the negotiation protocol under consideration, two bidding approaches are studied. Some protocols require the ability to raise bids. For those protocols, a monetary system, similar to the one proposed in (Di Gaspero et al., 2004) is designed. Each ward receives an 'amount of money' to bid on certain shifts. Initially, this amount equals the ward's own penalty value. This way, each ward has enough money for solving its problems. The starting bid for each shift is zero and the incrementation step size is one. A bid is only raised by a ward if the new bid is lower than a maximum determined beforehand. For protocols that do not rely on increasing bids, the bid simply equals the difference in penalty.

We give a brief description of each protocol discussing relevant design decisions specific for this negotiation setting.

6.2.1 Contract Net Protocol

The Contract Net Protocol (CNP), designed by Smith (1980), starts by sending a *call for bid* message to each ward for an unassigned shift. Each ward is allowed to reply with a bid. After all bids have been received, the highest bidder is assigned the shift. This is repeated for each unassigned shift until none is left. To improve the quality of the outcome, we first order the shifts using the heuristic ordering algorithm described above.

6.2.2 Extended Contract Net Protocol

Aknine et al. (2004) designed the *Extended Contract Net Protocol* (ECNP) to overcome some limitations of the original CNP. For example, the use of CNP in a multi-agent setting may cause lengthy negotiation processes, especially for applications where contractors are forced to sequence their negotiations. To resolve this problem, temporary bidding and temporary accepting bids is introduced. Multiple negotiations run in parallel, reducing the length of the negotiation process. Using this protocol, bidders can more easily study the impact of bidding for a specific combination of shifts. Because bids are made on a temporary basis, bidders can easily withdraw offers to cover some specific shift, when more appropriate calls for bids are received or when the previous bid resembles to be incompatible with future ones.

6.2.3 Simultaneous Ascending Auction

The Simultaneous Ascending Auction (SAA) is a multi-round auction protocol (Milgrom, 2000). The nurses send bids for all tasks at the same time. After each round, the nurses are informed of the highest bids received for each task. The minimum bid for a task is the maximum bid for the task in the previous round. If no (improved) bids are received for any task, the auction stops and the tasks are assigned to the highest bidding nurses.

6.2.4 Limited Vickrey Auction

We implemented a variant of the Generalised Vickrey Auction (Clarke, 1971; Groves, 1973) called Limited Vickrey Auction (LVA). In such auctions, with M goods, bidders make offers for one (or more) of the $2^M - 1^2$ possible packages. Although this protocol is optimal, it is very inefficient and may even become

[2] We do not take empty packages into account.

computationally intractable for a large number of goods. For example, consider a small nurse rostering problem consisting of a planning horizon of 4 weeks, 4 shift types. The demand is 4 nurses per shift type per day. This results in a total of 448 shifts. Suppose 10% of the shifts are hard to schedule and therefore problematic. This results in at least 2^{44} possible packages. Therefore, instead of bidding for all combinations of shifts, we bid for packages consisting of all or some of the 10 shifts raising the largest penalties. When computation time is of less importance, this number may be increased.

6.2.5 Ascending Proxy Auction

In an Ascending Proxy Auction (APA), a combinatorial auction designed by Ausubel and Milgrom (2004), nurses do not bid on all shifts simultaneously, but on a limited number of shifts, bundled in a package. After each round, provisional winners are announced. The packages are constructed incrementally during the negotiation process. A nurse starts bidding for a package consisting of a single shift. If the package is accepted, the nurse adds another shift to the package. After each round, the auctioneer (e.g. the hospital manager in this case) is responsible for determining the combination of non-overlapping packages maximising its revenue. This problem is known as the winner determination problem (Cramton et al., 2006) and is known to be NP-hard. We implemented a simple algorithm to tackle this problem: if an overlap between two packages is found, the package resulting in the lowest improvement from the global point of view is omitted. The auction stops when all unassigned shifts belong to a package.

6.3 Experiments and results

We perform a qualitative analysis of the aforementioned negotiation protocols. We compare the protocols on the following criteria:

- Speed: we measure the CPU time, the time effectively used by the cpu(s) when running the protocol.
- Quality: the quality for a ward is expressed by the penalty associated with its roster. The global cost is defined by the sum of the penalties of the individual rosters.
- Network load: we measure the number and size of messages transmitted over the network.

We also compare the negotiation protocols with a centralised approach. The experiments are performed on variations of the *BCV-A.12.1*, *BCV-1.8.1* and *BCV-5.4.1* benchmarks of Brucker et al. (2010). The original benchmarks instances were not suitable for testing the negotiation protocols because the planning horizon is non overlapping. Therefore, the planning horizons are shifted so that they fully overlap. We study negotiation between 3 agents, each representing a ward responsible for solving one of the benchmark instances, and a server. For most protocols, the server mainly offers administrative functionality such as keeping track of the wards present in the hospital. For the combinatorial APA, the server is responsible to solve the underlying winner determination problem. Each agent and the server is running on a separate computer (no virtualisation) hosting a Linux operating system. We run the negotiation protocols for 15 different starting rosters.

6.3.1 Speed

Figure 6.1 shows the comparison of the protocols on total computation time. The LVA is the slowest protocol, taking about 10 minutes. Cause is the naive method for searching the optimal combination of packages. The CNP is reasonably fast. The shifts are allocated immediately within one round. The ENCP consists of multiple rounds in which the best bidders often change after considering provisional winners. All bids have to be confirmed, resulting in extra rounds and more computation time. The ENCP is therefore slower. For large problems, the SAA appears to be the fastest approach. The bidders bid on all items simultaneously, so less rounds are needed to find an acceptable solution.

6.3.2 Quality

The aim is to use negotiation to improve the quality of the individual rosters. In other words, we would like to lower the total sum of the penalties of the individual rosters. Figure 6.2 shows the penalties obtained after negotiation. The dotted line shows the penalty of the original rosters (t.i. the penalty after running the local VNS algorithm). Most protocols sometimes return an inferior solution compared to the original solution. Shifts, unassigned from a certain nurse, are sometimes very specific (e.g. due to the specific skills required to cover the shift) and no better nurse may be found to cover the shift. If such a shift is selected for negotiation, the overall quality will be very hard to improve and negotiation often will result in a higher penalty.

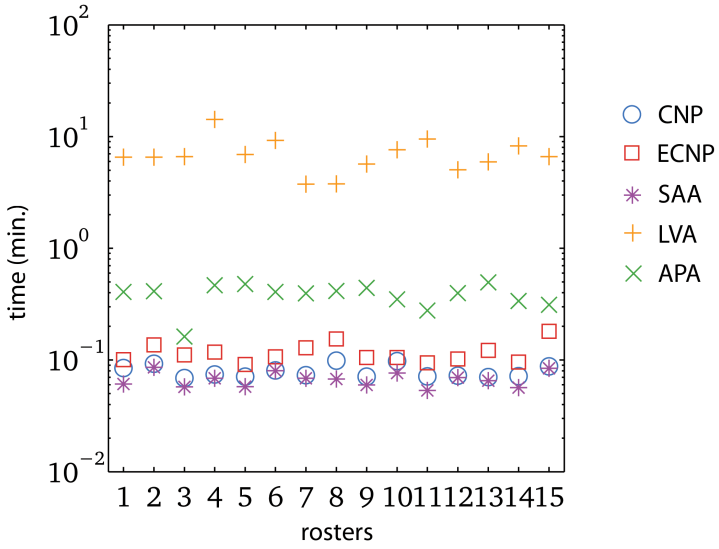

Figure 6.1: Comparison of protocols on total computation time

The lowest penalty value obtained is about 2550. If we zoom in on this part of the graph, we see that in most cases three protocols obtain similar results: the CNP, the ECNP and the SAA. In all cases the ECNP either finds the best solution or one that differs little from the best.

Remarkably, the CNP does not perform much worse than the other, more complex approaches. As expected though, the ECNP obtains better results than the CNP in most cases and where the CNP fails (i.e. for roster 4), the ECNP still manages to find excellent solutions.

Looking at the combinatorial auctions, we see that the LVA often improves the local rosters and outperforms the APA auction, yet both do not perform as well as the SAA auction. For the LVA, this can be explained by the limited number of shifts (10) eligible for negotiation. If this value is increased, better solutions are found. However, this affects the total negotiation time which is already quite high (see Figure 6.1).

For half of the rosters, the APA results in (slightly) worse rosters. This is mainly because of the way the packages are constructed, it is possible that a suboptimal package is accepted, and because of the suboptimal method for solving the winner determination problem.

Figure 6.3 shows the ranking of the protocols. The ENCP offers the best

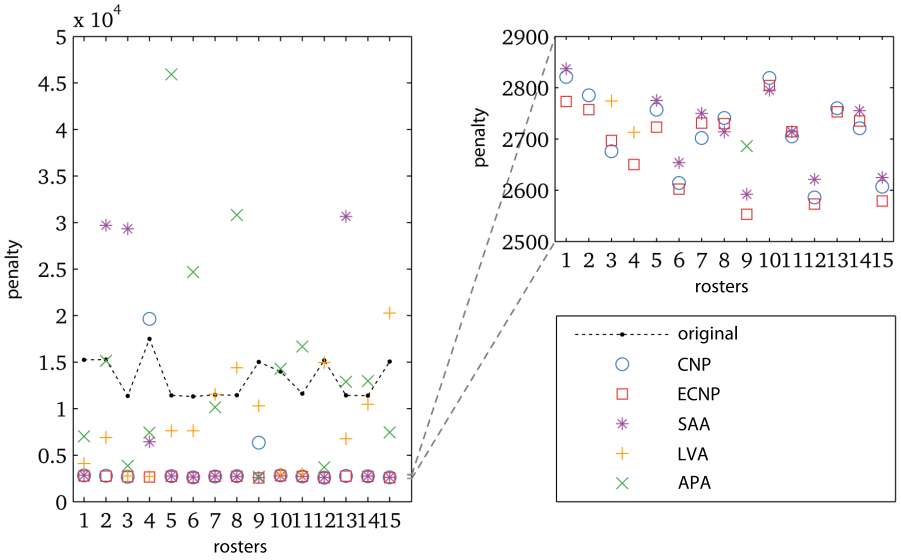

Figure 6.2: Comparison of protocols on penalty

solution in 60% of the cases and in more than 90% either the best or second best solution.

6.3.3 Network load

The negotiation protocols are implemented on a distributed system. It is therefore important to consider the communication time of the different protocols. Figure 6.4 (a) shows the number of messages sent between all participants and the server. The most notable difference to the CPU time (Section 6.3.1) is that of the ECNP. Although the total cpu time used by the ECNP is low, a large amount of messages is sent. So, seemingly faster than the APA, the wall clock time of the ECNP is presumably longer. The SAA needs not only the least computation time, the number of messages sent is also minimal. The size of the messages sent however, is larger (Figure 6.4 (b)).

6.3.4 Comparison with centralised approach

Figure 6.5 shows the comparison of the negotiation protocols with a centralised approach using two criteria, computation time and penalty. As expected,

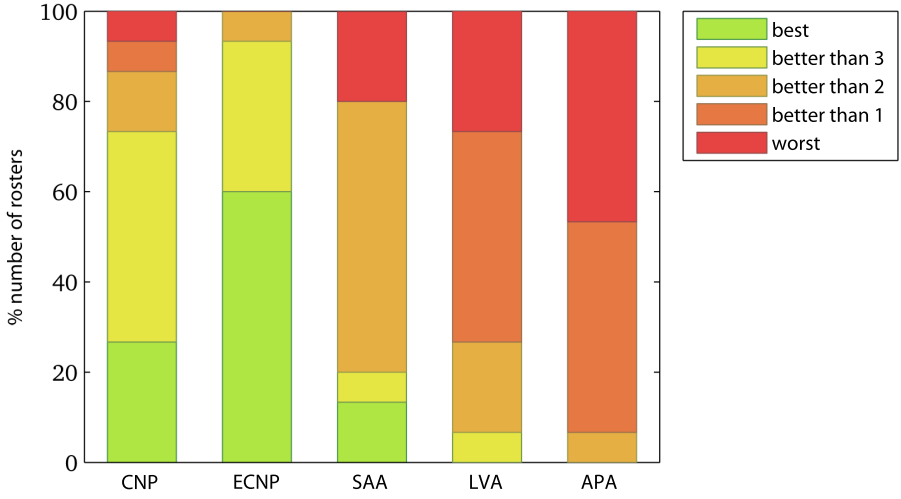

Figure 6.3: Ranking of protocols

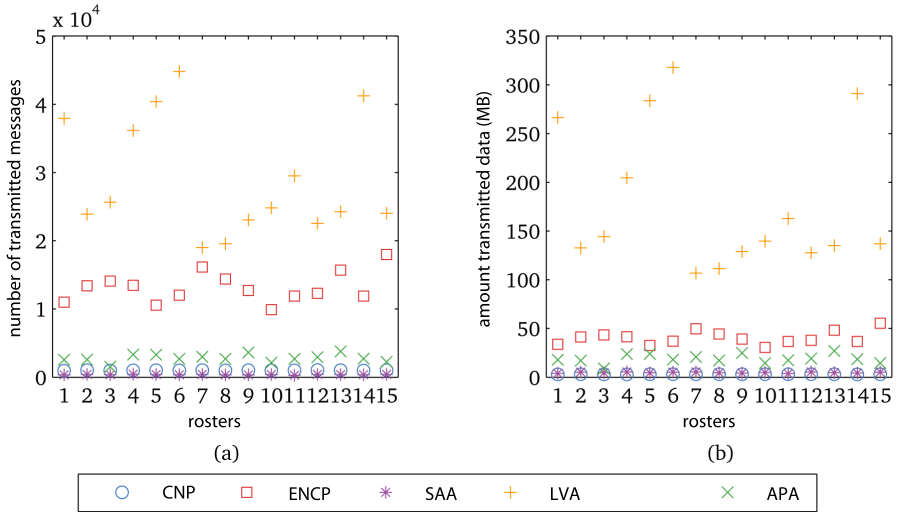

Figure 6.4: Comparison based on number of transmitted messages (a) and the amount of transmitted data (b)

solving smaller rosters in parallel results in a performance boost. The computational overhead caused by the negotiation protocols is, except for the

LVA, minimal. After negotiation, the global penalty is lowered significantly. On average, the rosters obtained by applying the ECNP after a local rostering algorithm are better than 'central' rosters. Solely based on quality, the ECNP is better than the other presented approaches.

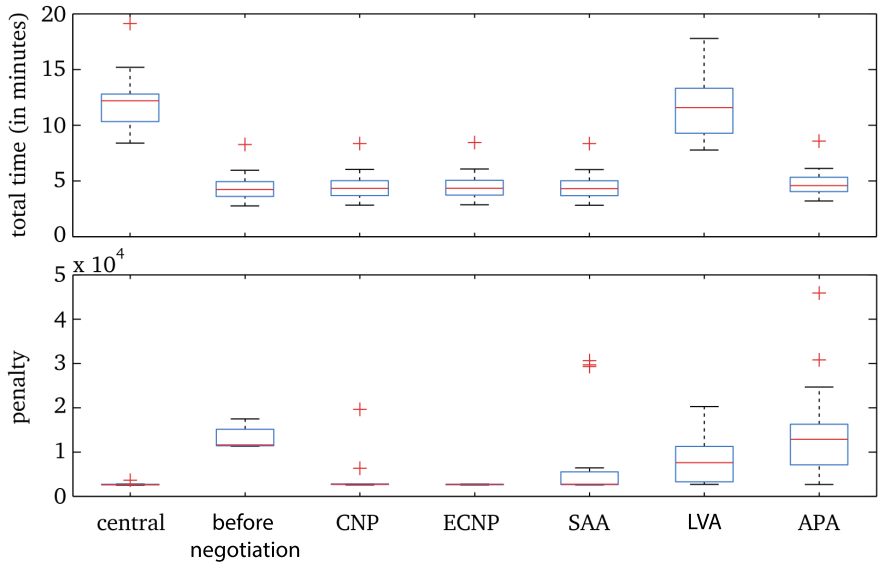

Figure 6.5: Comparison with a central solution method)

6.4 Conclusion

The aim of this research was to prove the feasibility of applying negotiation in the context or nurse rostering problems in the broader context of a hospital. By negotiation, the overall computational time is kept within acceptable boundaries while still preserving satisfactory quality levels at the wards' rosters. Most of the presented negotiation protocols, except for LVA, add little computational overhead. The quality of the solutions after negotiation is comparable to the quality of the solutions obtained by a central approach while the performance is significantly improved. Of the presented negotiation protocols, contracting offers the best results. If speed is an issue, the Contract Net Protocol is often sufficient. When better quality is required, the Extented Contract Net Protocol is a better candidate. On average, the ECNP obtains even better results than the heuristic central solution method.

We found combinatorial auctions either very inefficient, as the results of the Limited Vickrey Auction show, or very complex as, for example, the incorporation of the Ascending Proxy Auction was not straightforward due to underlying complex (decision) problems such as the winner determination problem of finding the best combination of non-overlapping packages. The naive algorithm we implemented is likely a major cause of the inferiority of the approach. We cannot draw general conclusions on the applicability and performance of combinatorial auction, but reports from literature show that sequential, single item auctions are much easier and in many cases perform faster and give better results than combinatorial auctions (Koenig et al., 2006; Wellman et al., 2003, 2001)

Solving personnel shortages has been put forward as an example problem for which negotiation can provide a solution. In such a setting, the problematic shifts subject to negotiation are those shifts for which the proposed staff is suddenly unavailable. By negotiation, a ward tries to outsource the shifts to the other wards in a hospital.

We spent some time to construct suitable benchmark instances for testing the negotiation protocols. Most benchmark instances available at the time of this research were defined in isolated planning horizons and do not support experiments in a negotiation setting.

In the current setting, agents are allowed to make extensive changes to the rosters of the nurses. In a realistic setting, nurses do not like major changes of their rosters, especially in case of last minutes changes caused by sudden personnel shortages. Similar to the nurse rerostering problem (Moz and Vaz Pato, 2003) only minimal changes should be made to the rosters.

In further research, the results of the negotiation setting should be compared with other methods for resolving shortages such as the nurse rerostering problem. Therefore, the negotiation techniques can possibly be applied to the nurse rerostering benchmark instances of Maenhout and Vanhoucke (2011). In a first phase, the suitability of the instances for negotiation should be studied. The model of the underlying nurse rostering problem of the benchmarks is different from the model targeted in this research. The currently used local heuristic algorithm must be changed in order to make it suitable to solve problems of the benchmark model. Futhermore, as the setting is completely different, proper criteria for comparison have to be identified. One possible criterion is the number of actual changes made to the rosters to resolve shortages. The fewer changes are needed, the better the performance of the approach. In the worst case, when the proposed benchmark set is found to be unsuitable, the disruption characteristics, used to construct the benchmark set, can be used to design a new benchmark set supporting negotiation.

In order to apply the presented techniques in real world settings, an adaptation and extension of the nurse rostering model is essential. The current (and the rerostering) model do not take the willingness of nurses to work in other wards into account. Not all nurses are eager to frequently change their working environment, while others do not want to be permanently assigned to one ward. Similar to Smet et al. (2012b), who study fairness objectives for the division of workload among nurses at the level of one ward, we should incorporate fairness measures when distributing (individual) shifts among the wards in a hospital. It should not always be the same ward that needs to cover additional shifts. The sole objective considered at the global level is to improve the global solution quality. In a realistic setting, the manager wants to raise certain quality levels, wants to minimize the personnel cost, wants to decide where resource shortages are allowable at peak moments ...

Chapter 7

Pareto optimal negotiation through algorithm analysis

This chapter presents an effort to narrow down the gap between short term level nurse rostering and the mid term level problem of distributing nurses and shifts between wards in a hospital. We treat this problem as a multi-agent multi-issue negotiation problem. We develop a negotiation protocol for handling such problems. Details of the short term level problem are input for the negotiation process.

We propose curves of constant utility to represent the operational performance of a ward. These 'operational performance level curves' are the input for the decision making process presented in this contribution. The levels of performance are determined by a nurse rostering algorithm at the short term level. The operational performance level curves hide the details of the short term problem that are irrelevant for the decision making process at the mid term level. Because nurse rostering algorithms take minutes if not hours of computing time to produce reliable results, we decide to use an offline approach. We use a large number of aggregated utility curves for each ward. The borderlines of the platforms can be thought of as the points where utility goes below a certain threshold. In our examples, the instability of the rosters tends to grow near these borderlines. We hypothesise that this is a general characteristic useful in delineating the platforms.

We model the wards as self interested agents negotiating distribution of personnel and workload. Approaching the problem through negotiation will allow individual wards to hide part of their internals. However, this will not

turn out to be an essential element. In cases where such hiding is not allowed, the negotiation protocol boils down to determining intersection points of curves of constant utility.

In Section 7.1 we position the above described negotiation problem in the field of multi-issue negotiation. Section 7.2 introduces indifference curves as a means to represent operational performance levels of a ward. We show how an in depth analysis of those indifference curves leads to solving the negotiation problem in hospitals were information hiding is not allowed in Section 7.3. A negotiation protocol for use in hospitals where information is kept private between the wards is proposed in Section 7.4. An evaluation of the protocol can be found in Section 7.5. Finally, Section 7.6 concludes.

This work has been submitted as *S. Haspeslagh and P. De Causmaecker, Bridging the gap between short term and mid term nurse rostering through a negotiation protocol, Journal of Scheduling*

7.1 Negotiation model

We use the taxonomy introduced in Section 2.2 to position the current contribution within the multi-issue negotiation research domain.

We consider a hospital with n nurses, w wards and s shifts to be covered. Each ward needs to solve a nurse rostering problem. Through negotiation the wards distribute the n nurses and s shifts among themselves while ensuring an acceptable operational level. Wards try to perform their tasks at minimum cost, given by a penalty function taking personal rules and preferences into account. The exact definition of the penalty function is the same as in Chapter 3. We refer to this penalty function as 'the objective function'. We use the terms 'penalty function' and 'objective function' interchangeably.

The number of nurses to be acquired and the number of shifts to be covered by a ward are the issues within the negotiation setting. Each ward w_i has a preferred workload s_i of shifts to cover and staff n_i of nurses. Each ward can operate within some boundaries. For a certain number of shifts to cover, a ward has a minimum and a maximum number of employees it wishes to acquire. Having either too many or too few employees results in an unworkable situation. The same holds from the point of view of employing a certain amount of nurses with respect to the number of shifts a ward is willing to cover. Those limits determine the operational platform of a ward.

From the above, it is clear that the issues considered within the current negotiation setting are strongly interrelated. The preferences of the nurses

and the local constraints are possible attributes in the negotiation setting.

In many hospitals, it is customary that information about the internal operation of the ward, e.g. details of the local nurse rostering problem such as preferences and the associated penalty function, is kept private. Moreover, this information is evolving over time, for example due to sudden illness of nurses.

Because we decided for an offline approach, i.e no rerostering takes place during negotiation, the negotiation problem considered belongs to protocol category (MI,SA) and the utility functions of the wards are invariant over time.

Summarised, we study a negotiation problem with strongly interrelated issues, where self-interested agents need to agree on the distribution of resources (employees) and tasks (shifts) among themselves. The protocol category is double-sided multilateral. We study both a setting with complete information and with private information. In the latter case, no mediator is present.

7.2 Determining the operational performance levels of a ward

The performance platform of a ward is determined by its capability to handle a certain amount of work (the demand) given a certain number of personnel. This capability depends on the details of the shift structure and of the personnel rostering constraints. The efficiency is expressed through the objective function. When determining the operational platform, the exact details of the demand and the personnel constraints are not relevant. What is of interest is an expected performance and stability against small disruptions.

In this section, we determine these quantities by studying the objective function values resulting from the solution of a large number of rostering problems. By plotting the value of the expected penalty function against the workload and the number of personnel, we obtain a graphical representation of the expected ward performance. As we will show, this allows us to determine the operational platforms as well as indifference curves to be used in negotiation.

Our instance distribution is based on the instances of the First International Nurse Rostering Competition 2010 (see Chapter 5). Within this sample problem setting, all employees are uniform, e.g. they are working following the same contractual regulations. A contract contains information on the constraints to be taken into account when building rosters. Contracts may differ per ward. For each instance, we varied the number of personnel available and the workload. The demand is distributed uniformly per day per shift type,

with an offset for the weekend days. The range of the demand is calculated as follows. Within this sample setting, on every day, we require at least 1 person for every shift type. Taking the offset for the weekend into account this results in the minimum demand:

$$\text{min. demand} = (\#\text{weeks}).(\#\text{shift types}).(7 + 5.\text{offset})$$

The maximum demand is determined by the number of available personnel in the ward. For every nurse, we add the maximum number of shifts that may be assigned to that nurse. The maximum demand is a multiplication of that number. Table 7.1 summarizes dimensions of the experiment to determine the operational level of a ward employing between 15 and 50 nurses. The planning horizon is 4 weeks and there are 4 shift types. The offset for the weekends demand is 2.

Nb. employees	15	20	25	30	35	40	45	50
Min. demand	264	264	264	264	264	264	264	264
Max. demand	348	488	600	740	852	988	1108	1228
Nb. instances	9	26	40	58	71	86	97	107

Table 7.1: Size of one experiment: number of instances to determine operational performance levels of a ward.

We solved the generated instances using the winning algorithm of the nurse rostering competition (Valouxis et al., 2012). Each instance was allowed a running time of 450 sec. The instances are publicly available at our website.[1].

The experiments have been run on the KU Leuven-UHasselt cluster[2]. We limited the runtime on the cluster to 3 hours. Thus, depending on the size of the experiment, we allocated a specific number of Xeon 5560 2.8 Ghz CPUS with 10GB of RAM. Table 7.2 shows the total runtime and memory usage.

Nb. of nurses	**Nb. of instances**	**Nb. of nodes**	**Total time**	**Total memory**
15-50	494	4	19:19:52	47GB
15-50	515	18	17:27:49	48GB

Table 7.2: Runtime and memory usage

Figures 7.1 and 7.2 show the value of the objective function plotted against increasing workload and increasing availability of nurses in a typical experiment.

[1] `http://www.kuleuven-kortrijk.be/codes/`
[2] `https://vscentrum.be/`

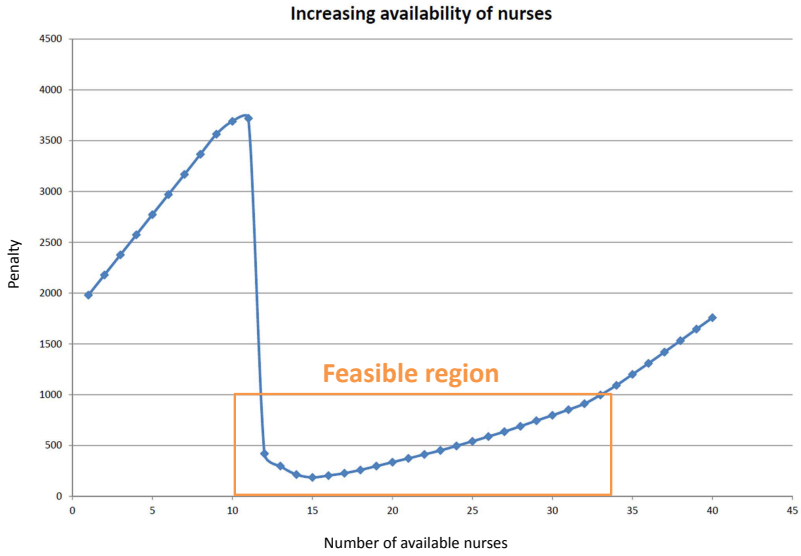

Figure 7.1: Increasing availability of nurses

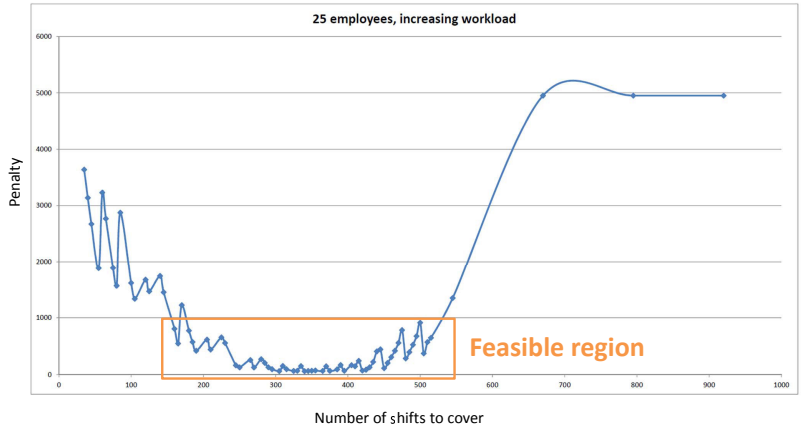

Figure 7.2: Increasing workload

We can clearly identify the regions of acceptable penalty for a ward. Near the borders of these feasible regions, the curves seem to show increasing instability. This instability makes it hard for wards outside their feasible region to decide whether a specific combination of demand and personnel is acceptable or not.

The diminishing number of data point over 500 shifts, seen in Figure 7.2, is also a cause of instability. Many instances in that region are unsolvable by the nurse rostering algorithm. For both demand and availability of nurses, within the feasible region, the penalty function displays a concave tendency.

To automatically detect the borders of the feasible region, we applied DBScan (Ester et al., 1996), a density based clustering algorithm. For DBScan, a cluster is defined by density reachability: a point q is directly density reachable from a point p if the distance between p and q is smaller than a distance ϵ. A point q is density reachable if there is a sequence of points $p = p_i, \ldots, p_n = q$, where each p_{i+1} is directly density reachable from p_i. Two points p and q are density connected if there is a point o such that both p and q are density reachable from o. A cluster is then a set of points that are all mutually density connected.

We apply this procedure to the border detection problem. Instability arises where the difference in penalty between a point and its neighbours is becoming too large. Points are within the feasible region if they are mutually density connected for a well chosen distance ϵ.

We used the DBScan implementation of the Weka Data Mining Software Suite (Hall et al., 2009). An advantage of Weka is that the algorithms it provides can be easily used within Java programs. The algorithm takes two parameters, the minimum number of points (minPoints) required to form a cluster and the distance ϵ. These values were experimentally determined. We ran the DBScan algorithm for different values of ϵ and minPoints. For the resulting clusters, we tried fitting the data points to a second degree polynomial. We studied the R^2 values of the resulting fits and obtained the best results when ϵ is set to 0.03 and minPoints to 4.

The penalty curves obtained are converted to utility curves as follows (Figure 7.3). A wards' utility (and thus performance level) is at the highest level if the penalty of its roster is as low as possible. The utility is at the lowest level when the penalty of its roster is at the highest level, thus at the borders of the feasible region. A point (s_x, p_x) is converted with the following formula:

$$u_x = \frac{p_x - p_{max_1}}{p_{min} - p_{max_1}} \text{ if } s_{min} \leq s_x < s_{pmin}$$

$$u_x = \frac{p_x - p_{max_2}}{p_{min} - p_{max_2}} \text{ if } s_{pmin} \leq s_x \leq s_{max_2}$$

The resulting utility curve is shown in Figure 7.4. For each possible number of employees, we obtain such a utility curve. Using those utility curves, we can identify all combinations of a number of nurses and a certain workload

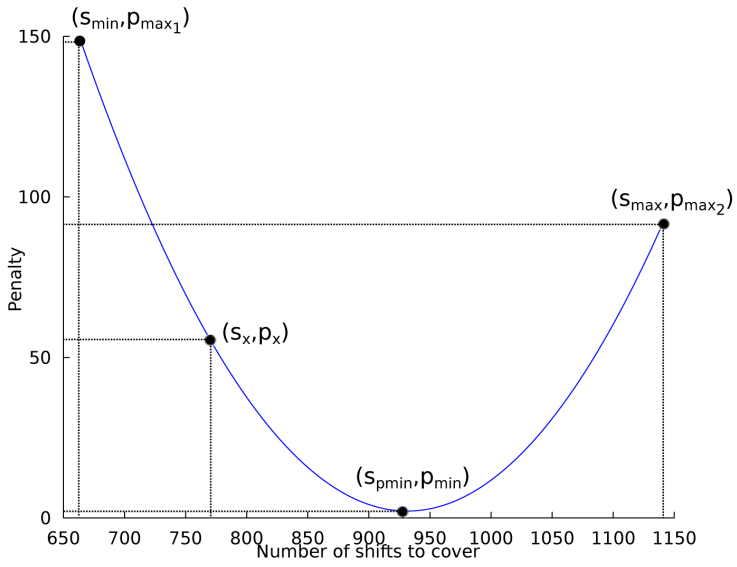

Figure 7.3: Conversion of a penalty curve to a utility curve

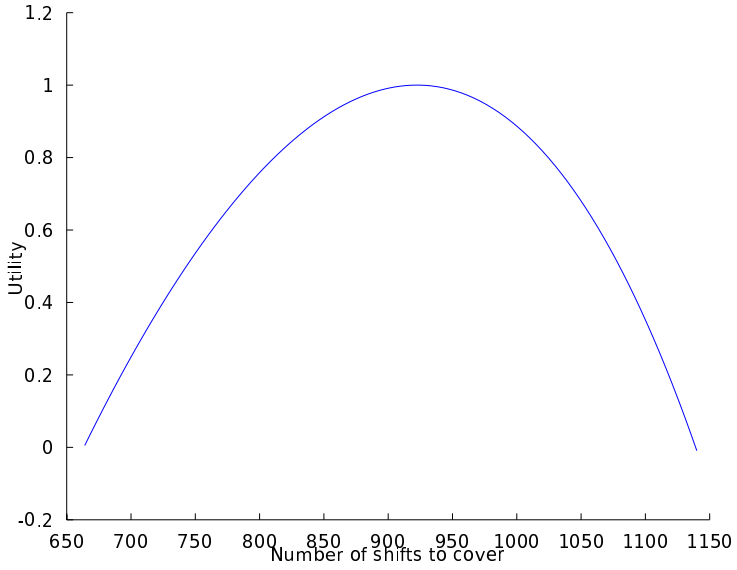

Figure 7.4: Utility curve

for which a ward operates at a certain performance level. So, for each performance level we can construct an indifference curve. A ward does not have a clear preference for points on that curve because for each combination of a workload and employee count on that curve the ward obtains the same level of utility. The concave utility function in Figure 7.4 shows that for utility level 1 only one indifference curve exists. The lower utility levels all produce two indifference curves. In fact, multiple indifference curves exist because we omit the restriction to monotonous utility functions as in (Wu et al., 2009) and therefore have to deal with strongly interrelated issues. An example of a set of indifference curves is shown in Figure 7.5. For the y-axis we use the scale $1/Demand$.

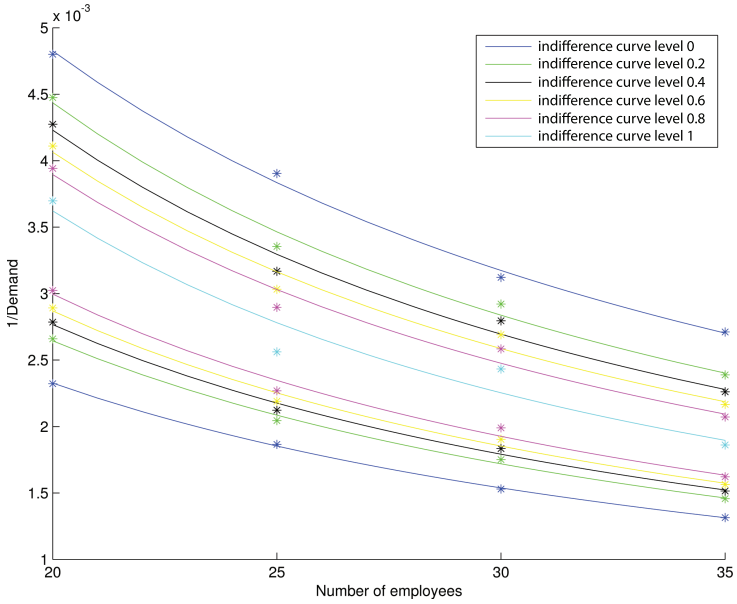

Figure 7.5: Indifference curves

The curves take the shape of power curves with an equation of the form $ax^b + c$. Table 7.3 contains the fitting data for the indifference curves of Figure 7.5. The R^2 values indicate that as the utility level decreases, the quality of the fit also decreases. This supports our hypothesis that near the borders of the feasible region more and more instability arises.

Utility level	a	b	c $(.10^{-4})$	R^2
1	0,0422	−0,8178	−0,2645	0,9961
Upper 0,8	0,0422	−0,8433	−0,1034	0,9959
Upper 0,6	0,0656	−0,9294	0,7400	0,9944
Upper 0,4	0,0584	−0,9140	0,8700	0,9905
Upper 0,2	0,0539	−0,9083	0,3890	0,9848
Upper 0	0,0491	−0,9167	0,1605	0,9601
Lower 0,8	0,1208	−0,9724	−0,3034	0,9944
Lower 0,6	0,1240	−0,9725	−0,1671	0,9918
Lower 0,4	0,1273	−0,9636	−0,2865	0,9904
Lower 0,2	0,1398	−0,9663	−0,3587	0,9867
Lower 0	0,1746	−0,9601	−0,8203	0,9674

Table 7.3: Fitting data

7.3 Indifference curves and negotiation in a complete information setting

In this section we discuss the negotiation problem presented in Section 7.1 in an environment where all information is commonly known by the wards. The objective of the wards is to be as close as possible to their optimal operational performance level. Using the utility curves identified in Section 7.2 the wards can measure the goodness of possible outcomes of the negotiation process and can thus determine the performance level they will be able to operate at by accepting a certain deal. We refer to the set of all possible outcomes as the solution space of the negotiation setting. Recall that the indifference curves, constructed in Section 7.2 aggregate all possible combinations of issues for which a certain performance level can be reached.

In this section we perform a more in depth analysis of those indifference curves. We show that in an environment with complete information, the negotiation process boils down to finding intersections with indifference curves.

Without loss of generality and for clarity we consider negotiations in a hospital consisting of two wards. In the sample setting, we consider a hospital that has 65 nurses available with a set of 1025 shifts that need to be covered.

In Figure 7.6 an indifference curve for utility level 1 is shown. As stated in Section 7.2 there exists only one curve for that level. The vertical solid line represents the maximum number of nurses available in the hospital. The horizontal solid line represents the total number of shifts to be worked in the

hospital (demand). Within the feasible region of the ward, in a realistic setting, the indifference curve cannot intersect any of those two lines.

If the curve intersects the vertical line within the feasible region, the ward should be able to cover all the work of the hospital when all the nurses are assigned to the ward. If the curve intersects the horizontal line within the feasible region then the ward would be able to cover all shifts even when not all of the nurses are assigned to the ward.

The intersection point *in* of the indifference curve with the vertical line highlights that the ward cannot cover all the shifts even when he employs all the nurses. The intersection point *is* with the horizontal line shows the number of nurses that is required when the ward would have to take care of all shifts.

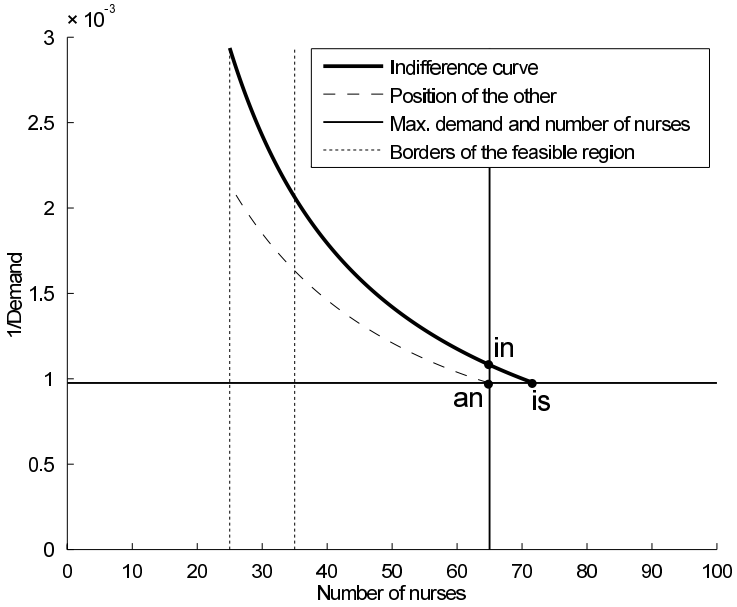

Figure 7.6: Indifference and positioning curve utility level 1

Furthermore, the information contained by the indifference curve combined with the information of the horizontal and vertical solid lines reveals important information on what offers the other ward in the negotiation setting must be able to accept. Concretely, for every point the ward may choose on its indifference curve, we can calculate the number of remaining shifts that need to be covered by the nurses that are left to be employed by the other ward. That way, a new curve arises. The curve positions the other ward within the

solution space. We refer to this curve as the 'positioning curve'. The curve 'starts' in the intersection point *an* of the horizontal and the vertical solid lines. If the ward is not assigned any nurse, then the other ward needs to cover all shifts and has all nurses available. The more nurses the ward employs, the more shifts it is able to cover. The other ward thus has fewer and fewer employees available to cover a decreasing number of shifts. Note that the positioning curve always lays below the indifference curve because the indifference curve always intersects the vertical solid line in a point higher than point *an* and the positioning curve starts in *an*. Furthermore, both curves cannot intersect.

Analogously, in Figure 7.7, we show similar results for lower utility levels. Below level 1 there exist two indifference curves per utility level. Each indifference curve induces a positioning curve. The upper positioning curve always corresponds with the lower indifference curve and vice versa.

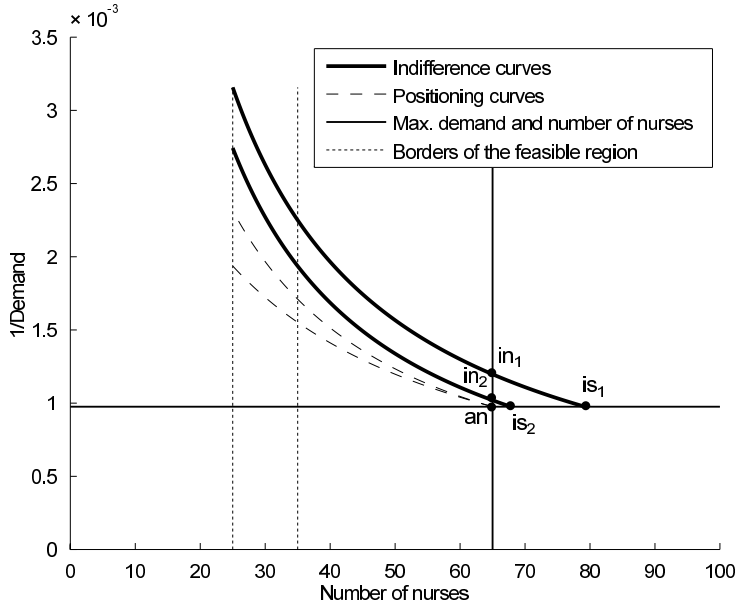

Figure 7.7: Indifference and positioning curves utility level 0,9

A ward will only accept offers laying on or between the two indifference curves. Outside that region the ward is not able to work at the required operational performance level. Furthermore, a solution to the negotiation process exists when the positioning curve intersects an indifference curve of the other ward on that level or if the positioning curve lays between the two indifference curves. A sample situation where a solution exists is shown in Figure 7.8.

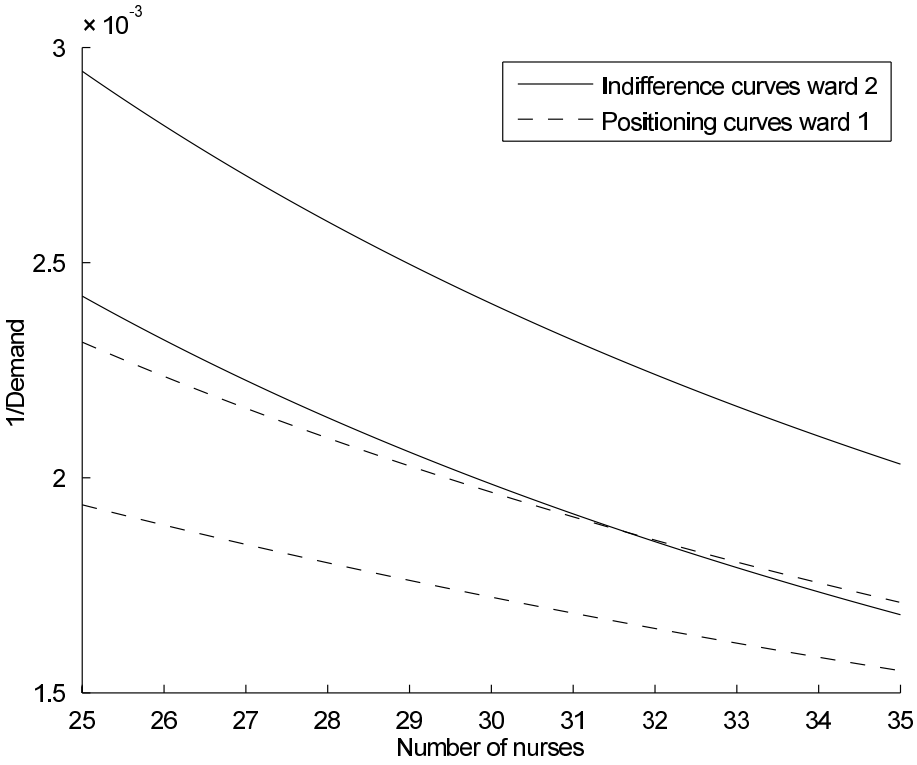

Figure 7.8: Intersection of positioning curve with utility curve

7.4 Negotiation with incomplete information

This Section elaborates on a negotiation setting where information is kept private. Agents do not know the indifference curves of their opponents and thus are unable to calculate the 'positioning curves'. The protocol discussed within this section is an alternating offer protocol (Rubinstein, 1982), based on the *Pareto optimal orthogonal bidding* protocol developed by Wu et al. (2009). The authors of the latter paper introduce the *reference point* indicating which number of issues is left for the agent whose turn it is to make the next counter-offer. By making offers as close as possible to this reference point (the bid is the orthogonal projection of the reference point on the agent's indifference curve) agents try to move their proposals towards each other to reach an agreement.

Initial experiments exposed some problems when applying the orthogonal search strategy in its original form to the negotiation problem tackled in

the current contribution. The convergence of the search method is very slow due to the shape of the indifference curves as presented in Section 7.2. The protocol above is developed for negotiation settings where issues are weakly interdependent, thus with only one indifference curve per utility level. In the current negotiation setting, two indifference curves per level exist (except for utility level 1). We present measures to circumvent these two problems.

In each round of the orthogonal search strategy of Wu et al. (2009), a step is taken from the reference point towards the indifference curve of the bidding agent. The step size and direction is determined by the orthogonal projection. In our protocol, a direction is chosen and consecutive steps are taken in the same direction. The step size is determined by the Fibonacci sequence, analogous to a Fibonacci search algorithm (Ferguson, 1960; Knuth, 1998). Thus, the step size is increasing within consecutive rounds of the protocol. The advantage of using *Fibonacci-steps* is fast convergence of the search. The protocol described below determines the direction and the size of each step to iteratively improve the approximation of the intersection point.

We start the description of our protocol with some definitions. W is the set of wards negotiating to distribute two issues, nurses (N) and (S) shifts, among themselves. A bid in round k is a tuple $(n_{i,k}, s_{i,k})$ with $n_{i,k}$ the number of nurses ward w_i wants to acquire for covering $s_{i,k}$ shifts. The reference point $(rn_{j,k}, rs_{j,k})$ for an agent j is defined by:

$$rn_{j,k} = |N| - \sum_{t \in \{1..|W|\} \setminus \{j\}} (n_{t,k})$$

$$rs_{j,k} = \frac{1}{\frac{1}{|S|} - \frac{1}{\sum_{t \in \{1..|W|\} \setminus \{j\}} (s_{t,k})}}$$

Note that each reference point always lies on a positioning curve and that the reference point can be calculated by each agent.

For both issues, a tolerance level is set, ϵ_n and ϵ_s respectively for the number of nurses and the number of shifts. Both tolerance levels are used for the stop criterion and, ϵ_n is also used to determine the step size within each round of the negotiation process. The step size in round k is given by: $\epsilon_n.Fib(k)$, with $Fib(k)$ denoting the k^{th} Fibonacci number. For each agent j, f_j keeps track of the index of the last Fibonacci number used. The indifference curve of agent j is represented by a function c_j such that for each bid, $c_j(n_{j,k}) = s_{j,k}$.

For the sake of simplicity, we further describe the protocol for two agents. The agent initiating the negotiation is chosen randomly. Without loss of generality, we refer to this agent as w_1. Agent w_1 randomly selects one of his indifference

curves and randomly generates an offer on that curve. Agent w_2 calculates the reference point. The direction of the first bid of agent w_2 is determined by orthogonal projection of the reference point on the closest of the two indifference curves. The distance is measured by the Euclidean distance. From the second round on, the direction of bidding for both agents is determined by the position of the newly calculated reference point with respect to the bid made by the agent in the previous round. For agent w_j and round k, the next bid $n_{j,k}$ is calculated as follows:

$$direction(j,k) = sign(rn_{j,k} - n_{j,k-1})$$

$$n_{j,k} = rn_{j,k} + direction(j,k).Fib(f_j).\epsilon_n$$

In each round, two checks are necessary for the successful operation of the protocol. First, an agent can detect that the negotiation is going into the wrong direction. Convergence in a direction can only occur if the distance between the reference point and the indifference curve in consecutive rounds is decreasing:

$$\begin{aligned} d_1 &= d((rn_{j,k}, rs_{j,k}), (rn_{j,k}, c(rn_{j,k}))) \\ d_2 &= d((rn_{j,k-1}, rs_{j,k-1}), (rn_{j,k-1}, c(rn_{j,k-1}))) \\ &\quad \text{change direction if } (d_1 < d_2) \end{aligned}$$

See Figure 7.9 for an illustration. Once both directions on an indifference curve have been explored, no agreement is possible on the current curve and the agent needs to make a bid on his other indifference curve.

Second, because of the increasing step size, an agent can cross a possible intersection point. An intersection point has been crossed when the position of the reference point with respect to the indifference curve has changed:

$$sign(rn_{j,k} - c_j(rn_{j,k})) \neq sign(rn_{j,k-1} - c_j(rn_{j,k-1}))$$

This situation is shown in Figure 7.10. Whenever an intersection point has been crossed, the direction of search is inverted and the step size is reset ($f_j = 1$). Several direction changes of this type might be necessary. The search oscillates around the intersection point until the required tolerance levels ϵ_e and ϵ_s are met. Thus, a direction change caused by this check does not count as a direction check as mentioned in the first check. Note that for a successful working of the protocol, the second check has to be performed first.

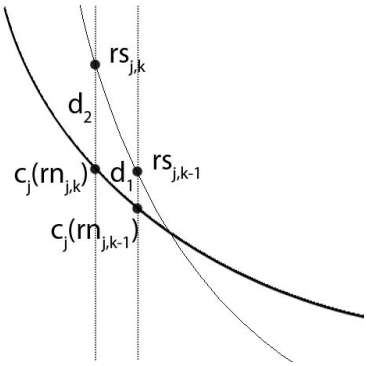

Figure 7.9: Direction check

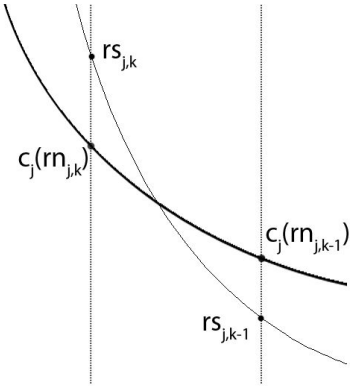

Figure 7.10: Position check

The search continues until all indifference curves in all directions have been considered without success or when the following stop criterion is met:

$$rn_{j,k} - n_{j,k} \leq \epsilon_n \text{ and } rs_{j,k} - s_{j,k} \leq \epsilon_s$$

A sample run of the algorithm is shown in Figure 7.11. If no agreement is found on a certain utility level, the agents concede to a lower utility level. Various concession strategies are possible. Wu et al. (2009) consider four different strategies. We list the three strategies of interest within the light of this contribution:

- fixed amount of utility: the agents concede a fixed amount au of utility.

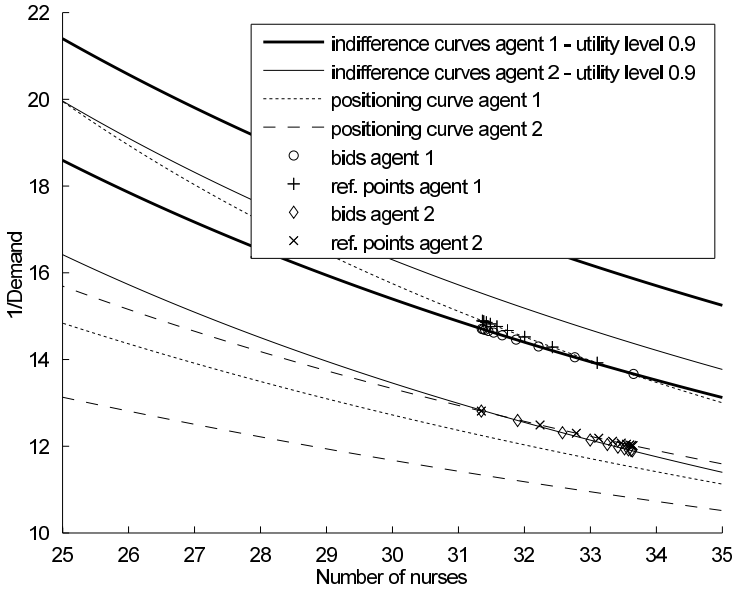

Figure 7.11: Sample run of the protocol

- fraction of utility: the agents concede a fixed fraction $fu \leq 1$ of the current utility level.
- fraction of difference: the agents concede a fixed fraction $fd \leq 1$ of the difference between the current utility level and the utility that he would get by bidding the last reference point.

The concession strategy and the corresponding parameter setting (au, fd and fr) directly influence Pareto optimality, the difference between the solution found and a Pareto optimal solution. Generally, small conceding steps lead to better solutions regarding Pareto optimality. In our setting, the Pareto optimal solution is the intersection point introduced in Section 7.3.

7.5 Theoretical analysis and experimental evaluation

In this section we evaluate the negotiation protocol of Section 7.4 from two points of view. In Section 7.5.1 we elaborate on a more theoretical analysis of the protocol, based on the criteria highlighted by Fatima et al. (2004). The

theoretical analysis mainly discusses properties of the solution. In Section 7.5.2, we experimentally perform a qualitative and performance analysis. The latter analysis focuses mainly on properties of the method used to search for the solution.

7.5.1 Theoretical analysis

According to Fatima et al. (2004), there are four main criteria for evaluating the outcome of a negotiation protocol:

- uniqueness: is the solution of the negotiation protocol unique? It the solution is unique, it can be determined unequivocally.
- efficiency: an agreement is efficient when Pareto optimality is achieved.
- symmetry: a negotiation protocol is symmetric if it does not treat the players differently on the basis of inappropriate criteria. For example, the outcome of the negotiating process should be independent of the agent starting the negotiation.
- distribution: is the outcome fair or is one agent favoured more than the others by the negotiation protocol?

Per pair of indifference curve and positioning curve, theoretically, at most two intersection points exist. As one positioning curve can intersect with at most two possible indifference curves and because there are at most two positioning curves, this results in at most eight possible intersection points. All those intersection points are equally valued by both agents. Thus, independently on which intersection point has been found, the negotiation protocol will always result in an outcome yielding the same amount of utility. Possibly, not all intersection points lay in the feasible region of the agents. In every experiment (Section 7.5.2), only one intersection point out of the theoretical possible eight lays within the feasible region. The negotiation protocol under study is efficient. Under the same assumptions of (Wu et al., 2009), Pareto optimality is achieved if the conceding step is small enough. The outcome of the protocol is independent of the starting agent. Regardless from which agent's perspective the analysis of the indifference curves start, both agents consider the same intersection points. Therefore, the proposed negotiation protocol satisfies symmetry. Distribution: as both agents concede the same amount of utility per round, no agent is favoured more than the other agent.

7.5.2 Experimental evaluation

We consider a set of hospitals with the numbers of nurses ranging between 55 and 65, with a step size of 2. The number of shifts to cover ranges between 900 and 1100, with step size 25. The utility levels we consider range from 0 to 1, with step size 0.1. For each combination of the above parameters, we perform 50 repeats to take into account the randomness of the starting agent and the initial bid. This results in a total number of 29.700 experiments. We discuss the following properties of the negotiation process:

- Speed of convergence: number of rounds needed to reach agreement or to detect infeasibility at a certain utility level.
- Quality of agreement: what is the distance between the theoretical intersection point and the solution found by the negotiation protocol?
- Success rate: when a theoretical intersection point exists, does the negotiation protocol find an agreement?

Speed of convergence

Figure 7.12 shows the results of the experiments on the speed of convergence. The use of a step size varying according to the Fibonacci sequence results in fast convergence, in at most 40 rounds. The number of rounds decreases if e_n, the nurse tolerance, increases. The smaller e_n, the fewer the effect of e_s, the shift tolerance. For $e_n = 1$, e_s has almost no effect. For $e_n = 0.1$ and $e_s = 0.01$ the following holds: the smaller e_s, the less rounds necessary to reach agreement. For a setting where no convergence can be reached, e.g. when no intersection point exists, only 8 rounds are necessary: 2 per direction for *two* indifference curves.

Quality of agreement

The quality of a solution is determined by the difference between the solution of the negotiation process and the corresponding intersection point. The quality is directly influenced by the settings of the tolerance levels. Figure 7.13 shows the difference in terms of percentage with respect to e_n. Analogously, the difference in terms of percentage with respect to e_s is shown in Figure 7.14. We only discuss the results in Figure 7.13. The results in Figure 7.14 can be discussed analogously. For an increased tolerance e_n, for example $e_n = 1$, the least quality is attained. Also, for this tolerance level, the influence of e_s is

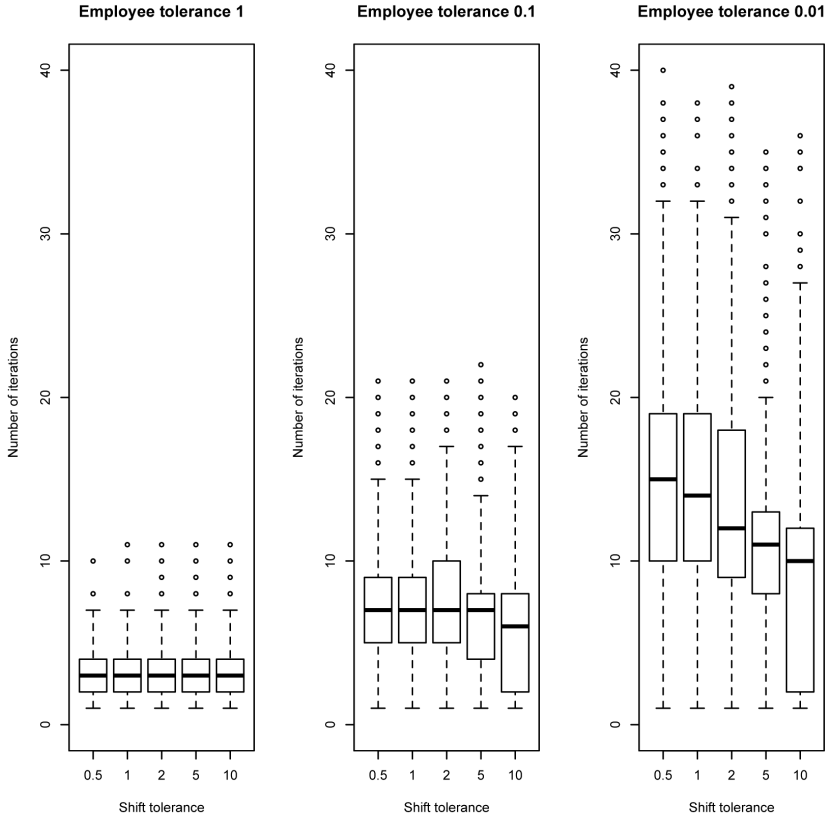

Figure 7.12: Speed of convergence: number of rounds

minimal. For an average nurse tolerance level, like $e_n = 0.1$, the quality of the solution is best for an average shift tolerance level ($e_s = 2$). For $e_n = 0.01$, the most strict nurse tolerance level, the quality increases when e_s decreases.

Success rate

Figure 7.15 shows the results for the success rate. Recall that within the negotiation protocol, steps of increasing size are taken, starting from e_n and according the Fibonacci sequence. For a low tolerance level, $e_n = 1$, this increases the fail rate of the protocol because the protocol may keep on 'crossing' the intersection point due to a too large step size. The success rate increases when e_n decreases. For a certain nurse tolerance level e_n, the success rate decreases as e_s decreases.

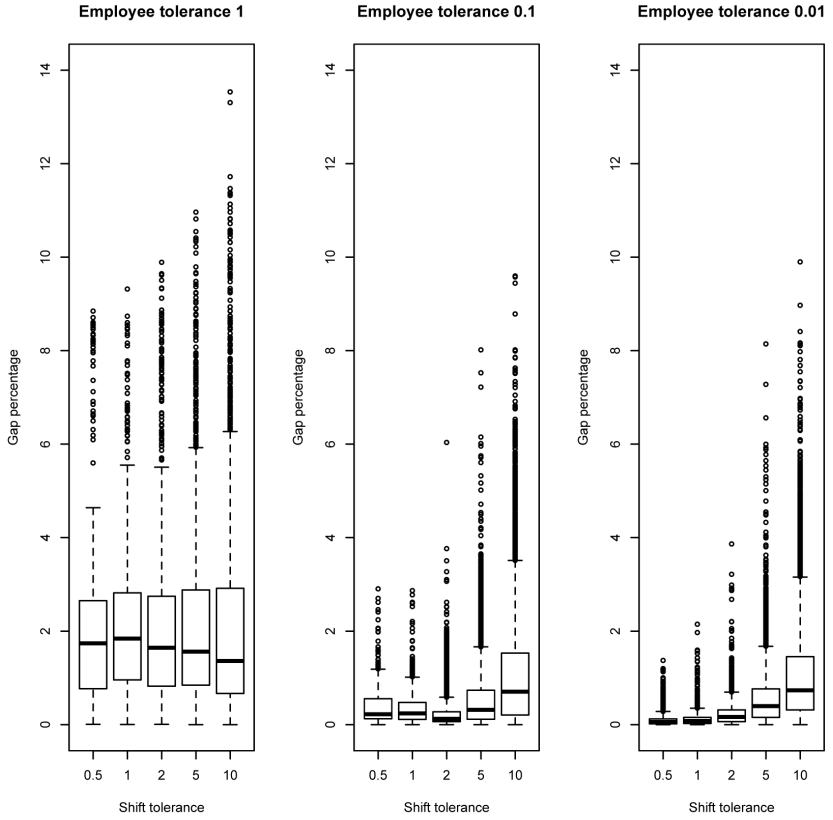

Figure 7.13: Gap percentage for the number of nurses

7.6 Conclusion

This contribution is an attempt to bridge the gap between short term and mid term level nurse rostering. An in depth study of an extensive set of nurse rostering problem instances, solved with a state-of-the-art nurse rostering algorithm resulted in the identification of operational platforms on which wards are able to guarantee good operation. Therefore, we identified feasible regions where wards are able to operate at acceptable penalty. We showed that instability grows near the borders of those regions. We successfully designed a method for the automated detection of these borders. The resulting penalty curves were translated to utility curves, which in turn gave rise to indifference curves. Those indifference curves in fact represent the operational performance levels of a ward. Hiding the irrelevant details of the short term level nurse

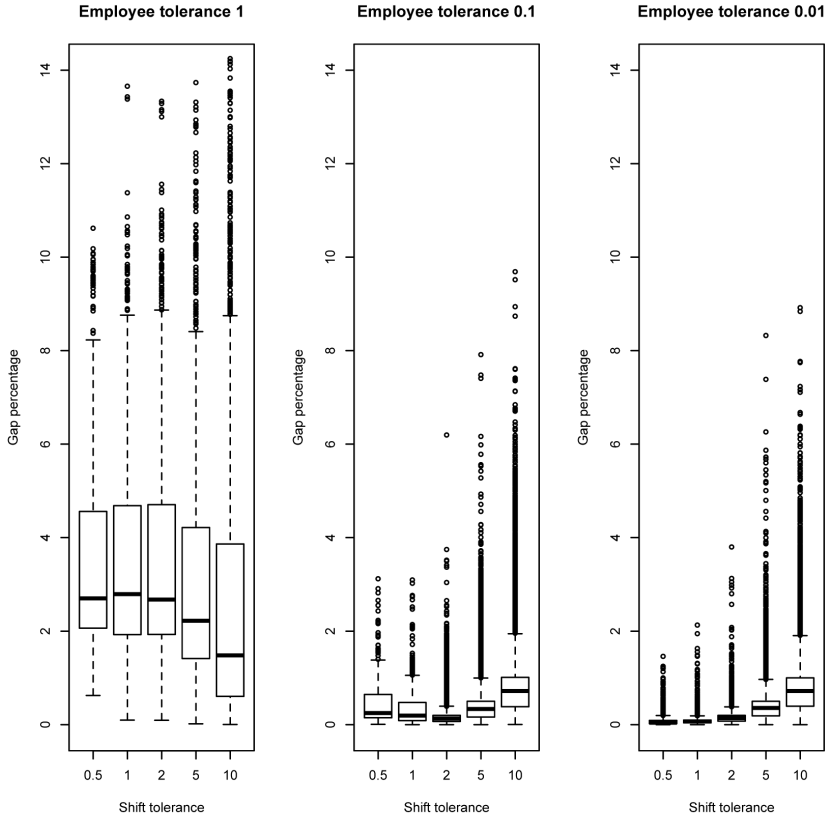

Figure 7.14: Gap percentage for the number of shifts

rostering problem, the indifference curves are input to the mid term problem of distributing nurses and shifts to cover between the wards of a hospital.

This problem was modelled as a multi-agent negotiation problem with strongly interdependent issues. In hospitals where all information on the indifference curves is common knowledge, we showed that the negotiation problem essentially is solved by determining intersection points of curves of constant utility. For hospitals with information hiding, we proposed a negotiation protocol for searching those intersection points. A proprietary protocol has been designed because either the proposed protocols for interdependent issues in literature put too many restrictions on the utility curves (e.g. linear additivity) or the proposed solutions methods are too complex to apply for this type of strongly interdependent issues. We designed a negotiation protocol exploiting certain properties of the resulting shape of the indifference curves.

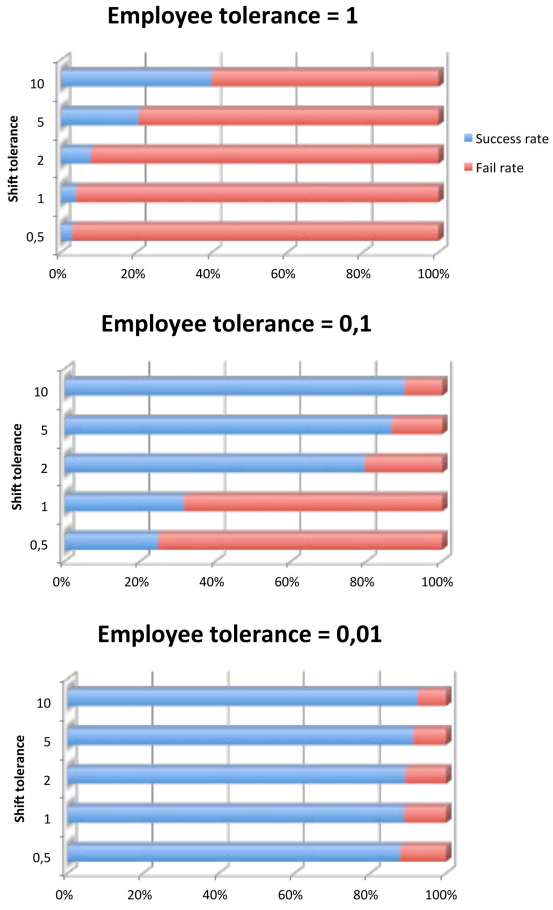

Figure 7.15: Success rate

As shown, the protocol is able to detect in an early stage that searching the indifference curve for an intersection point in a particular direction will fail. The application of a variable step size according to the Fibonacci series, as only two directions are possible and as there are only two indifference curves to explore, we were able to design an efficient negotiation protocol.

We showed the protocol satisfies important theoretical properties. In the experiments, at most one intersection point per utility level is found (uniqueness of solution), the protocol obtains Pareto optimal solutions (efficiency), the result is independent of the starting agent (symmetry) and no agent is favoured in the negotiation process (distribution).

Next, a qualitative analysis has been performed. The influencing parameters are the nurse and shift tolerance levels. For the lowest tolerance levels, at most 40 iterations are necessary to reach conversion. In a setting when no convergence can be reached, the protocol is able to detect in only 8 iterations that no intersection point exists. The experiments show best quality results are obtained when the tolerance levels are balanced: a low (high) tolerance level for one issue requires a low (high) tolerance level for the other issues, for an average nurse tolerance an average shift tolerance level and vice versa suits best. For low tolerance levels, the lowest success rate is achieved. This result is due to the "Fibonacci step size". For low tolerance levels, the algorithm may keep "crossing" the intersection point.

We studied the problem for two agents. In a multi-agent setting with complete information, the adjustments are straightforward. The same concepts of the two agent setting can be applied. Main difference is that the "positioning curve" has to intersect with the indifference curves of all the other agents. The changes to the protocol for the incomplete information case is not trivial. The reference point, used to position the bids of the other agents and to determine a suitable counter offer, remains the same. The original convergence checks are not valid any more. An agent cannot detect whether the protocol converges based solely on the "direction check". The original idea was that if the distance between the reference point and the indifference curve is increasing, no intersection point when bidding in that direction. As the reference point is based on the aggregation of the bids of all opponents, the direction of movement of the reference point offers no information on convergence; it is not because a bid of one agent increases the distance between the reference point and the indifference curve, that a counteroffer of another agent cannot decrease that distance. More advanced convergence checks need to be developed in order to apply the negotiation protocol in a setting with more than two agents. One possible alternative is to start bilateral negotiations between every pair of agents, using the original protocol. If an agreement is found in every bilateral negotiation, an outcome for the multi-agent problem is found[3]. For n wards, The number of pairs is $\binom{n}{2} = \frac{n(n-1)}{2}$. According to the experiments, the complexity in terms of number of iterations to find a solution is about $\frac{40n(n-1)}{2}$, which is $O(n^2)$.

Although the results of the nurse rostering algorithm, necessary for the construction of the indifference curves, are obtained in less than 3 hours, the naive approach we used is rather resource consuming and computationally intensive. One possible solution is to predict the value of the objective function using hardness analysis of nurse rostering problems (Bilgin et al., 2009). The computational process itself can be made more effective by concentrating on

[3]Hereby we assume at most one intersection point per pair of positioning curve and indifference curves exists.

the search for the points where instability comes in, effectively compromising between accuracy and computational efficiency. Also, in a real world hospital, the simulation step may be omitted, because information on the rosters from the past (e.g. the last 6 months) can be used as input to calculate the operational performance platforms. In this case, it is required that the hospital already uses some software for constructing the rosters (whether automated or not) and information on the valuation of the rosters is present, preferably in the form of an objective function.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

One aim of this thesis was to further close the gap between research and practice in (nurse) rostering research and to further stimulate and support research within this research area by following directions (for proper research) given by Schaerf and Di Gaspero (2007). An important keystone in this ambition was the organisation of "The First International Nurse Rostering Competition 2010". The model for the instances that were subject of the competition consisted of a set of constraints derived from 'real world' nurse rostering problems. New approaches, ideas and insights were gained. The lessons learnt from the competition (Section 5.6.4) contain important hints for researchers and practitioners when designing solvers for rostering problems. We were able to attract competitors from different areas of research. The contributions from the Constraint Programming (CP) community highlight once more the importance of accurately modelling of the problems under study. One competitor, not from the CP community, was performing excellent on the Early and Late instances but the algorithm failed on the Hidden instances due to a faulty assumption on the planning horizon. The competitor assumed the weekend days to be fixed at certain time units, because the planning period of the Early and Late instances started on the same day. This resulted in an incorrect modelling of the constraints related to weekends. We acknowledge that adding instances with different planning horizons to the set of Early and Late instances could have avoided making the wrong assumption. Notwithstanding, the competitor was the only one with the mistaken assumption.

To avoid misinterpretations and ambiguities as much as possible, we developed an exact description of the constraints and their evaluation based on numberings. We were able to express all constraints targeted by the competition with numberings. While a numbering provides an unambiguous representation of the constraints, quite a few competitors encountered problems with interpreting the numberings. Apparently, interpreting the numbers requires an extensive knowledge of (all details) of the evaluation procedure (Burke et al. (2001)). This raises the question for a more semantic definition of numberings. As an alternative, in this thesis, a mathematical description of the constraints is given. In any case, expressing the constraints in more than one "language" may help to reach a larger number of researchers. We believe it is good practice to provide a complete and formal description for any rostering problem along the above lines. We acknowledge this may seem irrelevant or may be infeasible due to for example space limitations. If not put in the contribution itself, we believe the information should still be available in for instance some technical report or easily accessible at some (non transient) website together with the instances on which experiments have been performed.

An implementation of the evaluation method, based on numberings and developed by Burke et al. (2001), is offered in both an online and offline version. Using the evaluators, researchers can thoroughly test proprietary implementations of the objective function. Both a structured XML and a text-only data format for instances and solutions to the instances was developed. We hope to reach a larger group of researchers by providing the instances in two data formates. Some researchers are not eager to use an XML format. One reason may be that acquiring the required skills to handle the XML format would simply take too much time. Tools for automated conversion between both formats are available. An automated conversion between the two formats is essential to ensure complete equivalence of the XML and the TEXT representation of the instances. The competition instances and the conversion and evaluation tools are available at the competition website (INRC2010). We keep track of best results, either found during the competition or afterwards.

Particularly hard in the organisation of the competition was to come up with an appropriate set of instances. The problems encountered with the Early instances of the competition highlights the need for complexity measures in this domain. Although the instances of the competition consisted of real-world constraints, the instances are artificial and only cover nurse rostering problem instances of one particular type. For example, nurse rostering problems in hospitals where cyclic rosters are required are not addressed in the competition. A next competition should incorporate at least a track with industrial rosters, collected from different departments, different hospitals and even different countries. In general, a wider variety of problem types creates a new challenge

for competing algorithms: to cover and to perform as good as possible on an as large as possible set of different nurse rostering problems.

The use of numberings as a formal representation allowed for the design of efficient schemes for the automated translation of nurse rostering instances into problems in different research domains such as SAT and MIP. The use of such schemes for exchanging problems between different domains ensures full compatibility of the instances. It allows to study the problems from a different point of view. The SAT translation was successfully used for a hardness study of nurse rostering problem instances (Bilgin et al., 2009). The MIP translation scheme allowed us to assess the complexity of the competition instances.

In the second part of the thesis we demonstrated the application of five state-of-the art negotiation algorithms within the context of nurse rostering. We modelled the wards in the hospital as self-interested agents, responsible for their rosters. Objective was to improve the global efficiency in the hospital. This can be seen as a goal at the managerial level of the hospital.

We studied the computation time, the communication time and the quality of the outcome of the negotiation protocol and compared it with a centralised approach. Most of the presented negotiation protocols, except for LVA, add little computational overhead. The quality of the solutions after negotiation is comparable to the quality of the solutions obtained by a central approach while the performance is significantly improved. Of the presented negotiation protocols, contracting offers the best results. If speed is an issue, the Contract Net Protocol is often sufficient. When better quality is required, the Extented Contract Net Protocol is a better candidate. On average, the ECNP obtains even better results than the heuristic central solution method. In general, negotiation significantly reduces computation time while still preserving satisfiable solution quality. We found combinatorial auctions either very inefficient, as the results of the Limited Vickrey Auction show, or very complex as, for example, the incorporation of the Ascending Proxy Auction was not straightforward due to underlying complex (decision) problems such as the winner determination problem of finding the best combination of non-overlapping packages. The naive algorithm we implemented is likely a major cause of the inferiority of the approach. We cannot draw general conclusions on the applicability and performance of combinatorial auctions, but reports from literature show that sequential, single item auctions are much easier, in many cases perform faster and give better results than combinatorial auctions (Koenig et al., 2006; Wellman et al., 2003, 2001)

The presented negotiation techniques can be applied for scheduling a mobile equipe in a hospital. A mobile equipe is a pool of floating nurses, not belonging to a particular ward of the hospital. If a shortage arises in a ward, the ward

can call in the help of a floating nurse instead of rearranging its rosters. The application of a mobile equipe is studied in 26 hospitals (Gobert et al., 2009). The rostering policies of the wards needs adjustments to support a mobile equipe. Questions arise such as what wards may call on help from the mobile equipe and in what case? For how long (in terms of consecutive days or shifts) may a ward use the services of a particular nurse of the pool? It should be avoided that certain nurses are always assigned to the same department. In this case, the dynamic character of the mobile equipe is lost and fewer shortages may be resolved due to a lack of free floating nurses. Nurses from the mobile equipe should only be resolved for shortages of a few shifts. Longer absences of nurses may be better solved by hiring agency nurses. The study in the 26 hospitals resulted in a set of guidelines to adjust roster policies to support the deployment of a mobile equipe. To apply negotiation, the model of the nurse rostering problem under study has to be altered taking those guidelines into account. The mobile equipe can then be modelled as a "virtual" ward. The manager of that virtual ward is responsible for the quality of the rosters of the floating nurses. In case of shortage, the wards in the hospital negotiate with the virtual ward to acquire floating nurses.

Finally, we gave an onset to close the gap between different levels of personnel management, more specifically between the mid and short term level of nurse rostering problems. We designed operational performance level curves as a means to feed short term level information to the mid term level problem of distribution of workload and personnel. We treated this problem as a multi-issue multi-negotiation problem. The curves can be seen as a way of controlling the complexity of the mid term level decision problem while still taking a vast set of attributes of the short term decision problem into account. More particularly, the curves hide attributes, of which the details are irrelevant to the mid term level, such as personal preferences (day off requests, ...) and small variations in demand between different days in the planning horizon (e.g. less demand in weekends) influencing the utility function of wards. This reduction of complexity allowed us to design a negotiation protocol that does not have to take into account the difficulties associated with methods designed for negotiation with highly rugged utility functions while not putting too many restrictions on the utility functions. We evaluated the protocol against four key criteria, uniqueness of outcome, efficiency, symmetry and distribution, as put forward by Fatima et al. (2004). In the experiments, at most one intersection point per utility level is found (uniqueness of solution), the protocol obtains Pareto optimal solutions (efficiency), the result is independent of the starting agent (symmetry) and no agent is favoured in the negotiation process (distribution). Next, a qualitative analysis has been performed. The influencing parameters are the nurse and shift tolerance levels. For the lowest tolerance levels, at most 40 iterations are necessary to reach convergence. In a setting

when no convergence can be reached, the protocol is able to detect in only 8 iterations that no intersection point exists. The experiments show best quality results are obtained when the tolerance levels are balanced: a low (high) tolerance level for one issue requires a low (high) tolerance level for the other issues, for an average nurse tolerance an average shift tolerance level and vice versa suits best. For low tolerance levels, the lowest success rate is achieved. This result is due to the "Fibonacci step size". For low tolerance levels, the algorithm may keep "crossing" the intersection point.

The protocol was designed for use in an environment with incomplete information. In hospitals with complete information, e.g. where the operational performance level curves are common knowledge, we showed that the negotiation problem boils down to finding intersection points between the operational performance level curves and the positioning curves derived from those curves.

In this research the introduced operational performance level curves were addressed for automating by negotiation the distribution of personnel and workload between wards in a hospital. The curves can also serve as a tool for a proper workload measurement in wards. As (Myny et al., 2010) point out, constructing quality rosters starts with a proper workload measurement. Several parameters are proposed for workload measurement in a hospital. The operational performance levels curves should take those parameters into account. Using the improved operational performance level curves, more objective decisions can be taken when distributing workload and personnel between the wards. In reality, the decisions are often made based on the subjective view the various parties at the negotiation table have on the problem under discussion.

8.2 Future Research

Some issues were not addressed in the research project. We give an overview per contribution of this dissertation of some of the challenging problems still left. We more specifically suggest research directions for the modelling and representation of rostering problems in Section 8.2.1. Section 8.2.2 elaborates on research opportunities for using the SAT (and MIP) translation schemes in combination with SAT (resp. MIP) solvers. We discuss and propose improvements of and extensions to the model for the short term negotiation problem of exchanging work, along with suggestions to improve the performance of the applied negotiation techniques in Section 8.2.3. Section 8.2.4 lists possible research lines for the negotiation protocol designed to tackle

a workload and personnel division problem and proposes solutions to the computational expensive data collection process for constructing operational performance level curves of wards. Finally, Section 8.2.5 elaborates on the integration of dependent decision problems either at the same level or at different levels of personnel management, for example in a hospital.

8.2.1 Modelling and representation of rostering problems

In Chapter 3, we used numberings as a formal representation of the constraints considered within this dissertation. As contributions by Burke et al. (2004) and De Causmaecker and Vanden Berghe (2011) show, the constraints (and models) found in literature are numerous and diverse in nature. It is clear that not all constraints can be represented using numberings. More constraints need to be added to our model and possibly the numbering methodology needs to be extended in order to add constraints that currently cannot be represented. For example, fairness constraints such as balancing the number of constraints violations between the nurses cannot be expressed using numberings. Currently the roster of one nurse can raise the majority of constraint violations while the other nurses' roster have low costs assigned.

8.2.2 Translation schemes to other problems domains

Chapter 4 presents translation schemes to SAT. As already mentioned, we translated the nurse rostering problem, which is an optimisation problem, to a satisfiability problem. Information on the objective function is lost. A natural extension is to translate the instances to MAX-SAT. In a first effort, MAX-SAT solvers (Argelich and Manyà, 2006) can be applied to study the solution quality obtained by those solvers. Another research direction is to design hybrid solvers. Current efforts first try to solve a partial problem with an exact solver (Burke and Curtois, 2011; Valouxis et al., 2012). The obtained solution is then optimised using for example a metaheuristic. One interesting research challenge is to study the opposite. SAT solvers are able to search the entire solution space. A metaheuristic search method only explores the solution space partially. In a sense, metaheuristics are designed for trying to escape local optima in the solution space, e.g. a metaheuristic is used to incorporate diversification in the search process. By adding extra constraints, based on the solutions obtained by a metaheuristic search method, we can force the (MAX)-SAT solvers not to explore those parts of the solution space covered by the metaheuristic search and therefore try to intensify diversification. The same ideas can be applied to the combination of a MIP solver and a metaheuristic.

8.2.3 Short term negotiation

In Chapter 6, we made some abstractions in the study of negotiation techniques for solving personnel shortages and improving efficiency. By applying negotiation, we tried to improve the global solution quality. Other managerial considerations need to be taken into account. For instance, not every nurse is willing to work in other wards, while others do not want to be permanently assigned to one ward. Similar to (Smet et al., 2012b), who study fairness objectives for the division of workload among nurses at the level of one ward, we should incorporate fairness measures when distributing (individual) shifts among the wards in a hospital. It should not always be the same ward that needs to cover additional shifts. Furthermore, the manager wants to raise certain quality levels, wants to minimize the personnel cost, wants to decide where resource shortages are allowable at peak moments ... In order to apply the presented techniques in real world settings, an adaptation and extension of the nurse rostering model is essential. We assume the subjects of negotiation (the shifts) have the same meaning in every ward. In reality, this is not the case. Shift types can have different definitions in different wards. An interesting research direction is to study ontologies for supporting negotiation (Tamma et al., 2005) in the field of nurse rostering research. The negotiation protocol does not need to be 'hard-coded in agents'. By agreeing on an ontology, agents match their vocabularies, acquire (background) knowledge on the negotiation protocol and their specific settings. For selecting 'problematic shifts' in a roster, we used an algorithm based on the heuristic ordering of shifts (Burke et al., 2008). More advanced measures need to be researched in order to identify better candidate shifts. One possibility is to develop measures based on SAT features, analogous to the hardness research (Bilgin et al., 2009) based on the translation scheme of Chapter 4. The performance of the combinatorial auction showed to be inferior to the performance of the other negotiation protocols. We used a simple algorithm for searching the best combination of non overlapping packages (combination of shifts). This problem, the winner determination problem, is known to be a complex (NP-hard) combinatorial optimisation problem (Cramton et al., 2006). Better algorithms exist, which presumably will result in better solutions. Similar features as for the 'candidate selection' can be used to better estimate the impact of assigning a specific package to a certain ward. Finally, applying negotiation in this context is in a sense a parallelisation or decomposition of larger nurse rostering problems. Currently, we studied decomposition up to the level of a ward and only for a limited number of wards. An interesting research direction is to study the impact of this decomposition on the solution quality, which are the boundaries (max. number of divisions) of the decomposition, is quality lost or gained and to what extent, ...

8.2.4 Mid term, multi-issue negotiation

We studied a negotiation protocol for the division of two issues among multiple agents in a ward in Chapter 7. Although the results were only discussed for two agents, the extension to multiple agents is straightforward. Because we consider an alternating offer protocol, we can aggregate the bids of the agents into a combined offer. This combined offer is used to calculate the reference point and no further changes to the protocol are necessary. A second extension is negotiation for more than two issues. For a small number of issues (ie. 3 or 4), a similar methodology as presented in Chapter 7 may be feasible. For a larger number of issues, such reasoning is probably inadequate. One possible solution could be to aggregate multiple issues into a limited number of categories, analogously to the aggregation of the attributes of the local nurse rostering problem into operational performance curves. Because we decided for an offline approach, the construction of the operational performance curves is resource consuming and computationally intensive. One possible solution is to predict the value of the objective function using hardness analysis of nurse rostering problems (Bilgin et al., 2009). The computational process itself can be made more effective by concentrating on the search for the points where instability comes in, effectively compromising between accuracy and computational efficiency. Also, in a real world hospital, the simulation step may be omitted, because rosters from the past (e.g. the last 6 months) can be used as input to calculate the operational performance platforms.

8.2.5 Integration of dependent decision problems

A challenging research direction in the line of the previous, is to further study the integration of different, dependent decision problems either at the same level or at different levels of personnel management. At the short term level in a hospital, we can for instance identify three dependent decision problems:

- the classical nurse rostering problems (Burke et al., 2004; De Causmaecker and Vanden Berghe, 2011), subject of this thesis.
- the patient admission problem (Demeester et al., 2010). Patients are assigned to beds in appropriate departments, dependent on medical requirements, satisfying as much of the patients' preferences as possible while balancing the number of patients between the different departments.
- the surgery admission problem (Riise and Burke, 2011). The problem involves assigning operating rooms and dates to a set of elective surgeries as well as the scheduling of surgeries of each day and room.

The number of surgeries performed influences the number of beds required for the patients and influences the number of nurses required to provide care for the patients. For each of the presented problems, decisions are taken at different levels. The number of beds available in a department/hospital influences the number of surgeries that can be performed and the number of required surgeons. A hospital can decide to open a new department or to close others to free resources allowing to specialise in particular treatments, ... The decision problems at the different levels affect each other and should be studied in this broader, dynamic context (De Causmaecker and Vanden Berghe, 2012).

Appendix A

Changes to the numbering evaluation method

The original evaluation method consists of three phases. The initialisation phase sets the start values induced by the solution of the previous planning period. The intermediate phase evaluates the constraints on consecutive events and rest periods for the current planning period. When the evaluation has reached the last event in the planning period, a final evaluation on the constraints is required. More details on these phases can be found in the original paper (Burke et al., 2001).

As we do not consider the previous planning period within the competition, we can also simplify the initialisation phase. It is sufficient to initialise $last\_nr$ to a certain value instead of running the initialisation algorithm.

For some constraints within the Nurse Rostering Competition, we need to denote how to evaluate them at the end of planning period. In this thesis, $last\_nr$ and $future\_nr$ denote how to evaluate a constraint at the borders of the planning period. The final evaluation phase taking $future\_nr$ into account, is given below.

```
FOR i=1, ... , I
    IF (total > max_total) THEN
        penalty_max_total = penalty_max_total +
        cost_max_total * (total - max_total)

    IF (total < min_total) THEN
        penalty_min_total = penalty_min_total +
        cost_min_total * (total - min_total)
```

```
    IF (future_nr != U)
        IF (last_nr = future_nr - 1)
            consecutive = consecutive + 1;

    IF (consecutive > max_consecutive) THEN
        penalty_max_consecutive = penalty_max_consecutive +
        cost_max_consecutive * (consecutive - max_consecutive)

    IF (consecutive < min_consecutive) THEN
        penalty_min_consecutive = penalty_min_consecutive +
        cost_min_consecutive * (min_consecutive - consecutive)

    FOR EACH number t in numbering i
        IF (pert[t] > max_pert[t]) THEN
            penalty_max_pert = penalty_max_pert +
            cost_max_pert * (pert[t] - max_pert[t])
        IF (pert[t] < min_pert[t]) THEN
            penalty_min_pert = penalty_min_pert +
            cost_min_pert * (min_pert[t] - pert[t])

    IF (last_nr != U AND future_nr != U) THEN
     between = future_nr - last_nr - 1
        IF (between > max_between) THEN
            penalty_max_between = penalty_max_between +
            cost_max_between * (future_nr - last_nr - 1)

    IF (last_nr != U AND future_nr != U) THEN
     between = future_nr - last_nr - 1
        IF (between > 0 AND between < min_between) THEN
            penalty_min_between = penalty_min_between +
            cost_min_between * (future_nr - last_nr - 1)

i=i+1
```

Appendix B

Formal description of constraints using numberings

We give a formal description using numberings for each constraints presented in Section 3.1 of Chapter 3.

B.1 Maximum number of assignments

Consecutive time units have consecutive numbers assigned. The variable max_total is set to the maximum number of assignments. last_nr and future_nr are Undefined. An example is given in Table B.1.

Mon		Tue		Wed		Thu		Fri		Sat		Sun	
E	L	E	L	E	L	E	L	E	L	E	L	E	L
0	1	2	3	4	5	6	7	8	9	10	11	12	13

Table B.1: Maximum number of assignments

B.2 Minimum number of assignments

The same numbering as for the maximum number of assignments constraint can be used (Table B.1). The variable min_total is set to the minimum number of assignments. last_nr and future_nr are Undefined.

B.3 Maximum and minimum number of consecutive working days

Each time unit on the same day has the same number. The first day is assigned 0. Consecutive days have consecutive numbers. Max_consecutive is set to the maximum number of consecutive working days. Min_consecutive is set to the minimum number of consecutive working days. last_nr and future_nr are Undefined. An example is given in Table B.2.

Mon		Tue		Wed		Thu		Fri		Sat		Sun	
E	L	E	L	E	L	E	L	E	L	E	L	E	L
0	0	1	1	2	2	3	3	4	4	5	5	6	6

Table B.2: Number of consecutive working days

B.4 Maximum and minimum number of consecutive free days

These two constraints use the same numbering as the number of consecutive working days constraints. Max_between is set to the maximum number of free days. Min_between is set to the minimum. last_nr is assigned -1 and future_nr is assigned ($numberofdays$), 7 for the numbering in table B.2.

B.5 Maximum number of consecutive working weekends

Each time unit of the days of the same weekend has the same number. Consecutive weekends have consecutive numbers. All other time units are assigned $Undefined$. Max_consecutive is set to the maximum number of consecutive working weekends. last_nr and future_nr are Undefined. An example is given in Table B.3.

Fri		Sat		Sun		Mon		Tue	
E	L	E	L	E	L	E	L	E	L
U	U	0	0	0	0	U	U	U	U
Wed		Thu		Fri		Sat		Sun	
E	L	E	L	E	L	E	L	E	L
U	U	U	U	U	U	1	1	1	1

Table B.3: Number of consecutive working weekends

B.6 Complete weekends

Time units on the same day are assigned the same number. Consecutive days in the same weekend have consecutive numbers. There must be a difference of at least two between the number of a weekend and the last day of the previous weekend. Min_consecutive is set to the number of days in a weekend, in our case this is three. last_nr and future_nr are Undefined. A higher penalty is raised if the working days in the weekend are not consecutive. In this case, with uniform cost, X 0 X (X = working, 0 = not working), raises a penalty of 4. An example numbering is given in Table B.4.

Fri			Sat			Sun			Mon			Tue		
E	L	N	E	L	N	E	L	N	E	L	N	E	L	N
0	0	0	1	1	1	2	2	2	U	U	U	U	U	U
Wed			Thu			Fri			Sat			Sun		
E	L	N	E	L	N	E	L	N	E	L	N	E	L	N
U	U	U	U	U	U	3	3	3	4	4	4	5	5	5

Table B.4: Complete weekends

B.7 Identical complete weekends

The time units for the same shift type on weekend days are assigned the same number. Different shift types are assigned different numbers. Different weekends are assigned different numbers. For each assigned number, min_pert is set to the number of days in the weekend. last_nr and future_nr are Undefined. An example is given in Table B.5.

Fri		Sat		Sun		Mon		Tue	
E	L	E	L	E	L	E	L	E	L
U	U	0	1	0	1	U	U	U	U
Wed		Thu		Fri		Sat		Sun	
E	L	E	L	E	L	E	L	E	L
U	U	U	U	U	U	2	3	2	3

Table B.5: Identical complete weekends

B.8 Single assignment per day

Time units on the same day are assigned the same number. Time units of different days are assigned different numbers. max_pert for each assigned number is set to one. last_nr and future_nr are Undefined. An example is given in Table B.6.

Mon		Tue		Wed		Thu		Fri		Sat		Sun	
E	L	E	L	E	L	E	L	E	L	E	L	E	L
0	0	1	1	2	2	3	3	4	4	5	5	6	6

Table B.6: Single assignment per day numbering

B.9 Two free days after a night shift

The time unit of the first night shift is assigned a number. The time units of all the shift types except for that of the night shift of the following days get a consecutive number. All other time units are assigned undefined. max_consecutive is set to one. last_nr and future_nr are Undefined. For this constraint, multiple numberings are required (see appendix 3.2.1). An example is given in Table B.7.

Mon			Tue			Wed		
E	L	N	E	L	N	E	L	N
U	U	0	1	1	U	1	1	U

Table B.7: Two free days after a night shift

B.10 Requested day on/off

All time units of a requested day off are assigned the same number. Different requests are assigned a different number. All other time units are assigned undefined. max_pert for each number of each requested day off is set to zero. The numbering is the similar for requested days on. min_pert is then set to one. last_nr and future_nr are Undefined. The example in table B.8 illustrates three day on/off requests: the employee either wants to work or wants to be free on Monday, Thursday and Saturday.

Mon		Tue		Wed		Thu		Fri		Sat		Sun	
E	L	E	L	E	L	E	L	E	L	E	L	E	L
0	0	U	U	U	U	1	1	U	U	2	2	U	U

Table B.8: Day off requests

B.11 Requested shift on/off

The time unit of the requested shift off is assigned a number. Time units of different shift off requests are assigned different numbers. All other time units are assigned undefined. max_pert for each assigned number is set to zero. The numbering is the same for requested shifts on. min_pert is then set to one. last_nr and future_nr are Undefined. An example is given in Table B.9.

Mon		Tue		Wed		Thu		Fri		Sat		Sun	
E	L	E	L	E	L	E	L	E	L	E	L	E	L
0	U	U	U	U	U	U	1	U	U	2	U	U	U

Table B.9: Shift off requests

B.12 Alternative skill

The time units for which the employee does not have the required skill are assigned a number. In the example in Table B.10, the employee is not allowed to cover a late (L) shift type.. Different time units are assigned different numbers. All other time units are assigned undefined. max_pert for each assigned number is set to zero. last_nr and future_nr are Undefined.

Mon		Tue		Wed		Thu		Fri		Sat		Sun	
E	L	E	L	E	L	E	L	E	L	E	L	E	L
U	0	U	1	U	2	U	3	U	4	U	5	U	6

Table B.10: Alternative skill - Employee cannot work shift type L

B.13 Unwanted patterns

An unwanted pattern is a sequence of assignments that a nurse does not want to work. We distinguish between patterns that are unwanted on specific days (e.g. a nurse does not want to work a night shift before a free weekend, a nurse wants to work on Friday before a working weekend, ...) and patterns that are unwanted throughout the entire planning period (e.g. a nurse does not want to work a late shift before an early shift, ...).

A pattern consists of a number of pattern entries X: $[X]_{1...\ n}$. A pattern entry X can be one of the following:

- ST: a specific shift type
- W: any shift type on a day
- F: free (no shift type) on a day

A pattern entry X can occur on any day in the scheduling period or on a specific day. Some patterns need multiple numberings (see Section 3.2.1).

We introduce the following pattern types:

1. $\{W, ST\} - [F]_{2...n}$: No shift or a specific shift cannot be worked before a number of free days. E.g. a night shift cannot be worked before a free weekend (Table B.12). We assign the time unit of the shift type (or all time units of the day) that cannot be worked a number. The free days following get a consecutive number. *min_consecutive* is set to two. last_nr is Undefined, future_nr is assigned a consecutive number of the free days, in this case two. A general example is given in Table B.11.

W			F			F			F			F		
E	L	N	E	L	N	E	L	N	E	L	N	E	L	N
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Table B.11: $W - F - F - F - F$

Fri			Sat			Sun		
E	L	N	E	L	N	E	L	N
U	U	0	1	1	1	1	1	1

Table B.12: $N - F - F$: No night shift before a free weekend

2. $F - [W, ST]_{2...n}$: No free day can occur before working any of a number of consecutive days or shift types. E.g. If an employee works a shift in a weekend, the employee should also work on Friday (Table B.14.). The free day before the working days (shift types) is assigned a number n. The working days (shift types) following get a consecutive numbers starting from $n+1$. *last_nr* is set to $n-3$. *min_between* is set to two. A seperate numbering is required for every series of shifts subject to the pattern. For the weekend example (Table B.14), a numbering is required per weekend in the planning horizon. A general example is given in Table B.13.

F			W			W			W			W		
E	L	N	E	L	N	E	L	N	E	L	N	E	L	N
3	3	3	4	4	4	5	5	5	6	6	6	7	7	7

Table B.13: $F - W - W - W - W$

Fri			Sat			Sun		
E	L	N	E	L	N	E	L	N
3	3	3	4	4	4	4	4	4

Table B.14: $F - W - W$: Working on Friday if working a weekend

3. $[ST]_{2...n}$: Unwanted shift type successions. E.g. L-E-L. Consecutive shift types in the pattern are assigned consecutive numbers. The other shift types are assigned undefined. There is a difference of 1 between the first number of a new series and the last number of the previous series. max_consecutive is set to the length of the pattern. last_nr and future_nr are Undefined. An example is given in Table B.15. Multiple numberings are required to express this pattern type (Section 3.2.1)

Mon			Tue			Wed		
E	L	N	E	L	N	E	L	N
U	0	U	1	U	U	U	2	U
Thu			Fri			Sat		
E	L	N	E	L	N	E	L	N
U	4	U	5	U	U	U	6	U

Table B.15: $L - E - L$

Appendix C

Rewriting non monotonous numberings

We show that the restriction to monotonous numberings (Definition 7 in Chapter 4) does not limit the expressiveness of the nurse rostering models. In fact, every constraint as presented in Chapter 3 can be expressed using monotonous numberings.

Consider for example the unwanted pattern late (L) - early (E), stating that a nurse should not have an early shift assigned after having worked a late shift on the day before. Numbering N_i in Table C.1 is an example numbering expressing this constraint for nurse rostering problem with a planning horizon of two weeks and two shift types. *max_consecutive* is set to 1.

Numbering N_i can be rewritten into multiple numberings. For the numbering method presented in (Burke et al., 2001), non monotonous numberings were introduced for compactness. Combining numbering $N_{i,1}$ and $N_{i,2}$ into one numbering N_i results in a more efficient evaluation. For use with the SAT translation (Chapter 4), replacing non monotonous numberings by a set of monotonous numberings will not increase the number of contiguously ascending sequences. Hence, this does not influence complexity.

Days	Mo		Tu		We		Th		Fr		Sa		Su	
Shift type	L	E	L	E	L	E	L	E	L	E	L	E	L	E
Time units	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Numbering N_i	0	22	24	1	3	25	27	4	6	28	30	7	9	31
Numbering $N_{i,1}$	0	U	U	1	3	U	U	4	6	U	U	7	9	U
Numbering $N_{i,2}$	U	22	24	U	U	25	27	U	U	28	30	U	U	31

Mo		Tu		We		Th		Fr		Sa		Su		Days
L	E	L	E	L	E	L	E	L	E	L	E	L	E	Shift type
15	16	17	18	19	20	21	22	23	24	25	26	27	28	Time units
33	10	12	34	36	13	15	37	39	16	18	40	42	19	N_i
U	10	12	U	U	13	15	U	U	16	18	U	U	19	$N_{i,1}$
33	U	U	34	36	U	U	37	39	U	U	40	42	U	$N_{i,2}$

Table C.1: Rewriting non monotonous numberings

Appendix D

Competition Rules

The competition has a set of rules that the participants have to obey. The rules are the following:

Rule 1: This competition seeks to encourage research into automated nurse rostering methods, and to offer prices to the most successful methods in particular tracks. It is the spirit of these rules that is important, not the letter. With any set of rules for any competition it is possible to work within the letter of the rules but outside the spirit. The organisers ask that you: "Please don't do this". It's not fair, it's not good science, and it will result in disqualification.

Rule 2: The organisers reserve the right to disqualify any participant from the competition at any time if the participant is determined by the organisers to have worked outside the spirit of the competition rules. The organisers' decision is final in any matter. Decisions will be made democratically by the organisers.

Rule 3: The organisers reserve the right to change the rules at any time and without notice. Any change of rules will be accompanied by a general email to all participants.

Rule 4: The competition has an opening day and a deadline when all submissions must be uploaded. This deadline is absolute and no extensions will be given under any circumstances because to do so would be unfair to other participants.

Rule 5: Participants have to implement an algorithm to tackle the problem on a single processor machine; they can use any programming language. The use of third-party software is allowed under the following restrictions:

- it is free software
- its behaviour is (reasonably) documented
- it runs under a commonly-used operating system (Unix/Linux or Windows)

Rule 6: The goal is to produce rosters in which all hard constraints are satisfied (i.e. feasible rosters) and to minimise the number of broken soft constraints.

Rule 7: Instances of different size and type will appear on the web site from the opening day. Two weeks before the deadline more instances will be placed on the web. A third set of datasets will be used to internally rank the top participants. The datasets are therefore classified as Early Instances, Late Instances and Hidden Instances. Participants should refer to the information associated with each track for further specifics on datasets. The Hidden Instances will be released after the competition is closed.

Rule 8: Participants have to benchmark their machine with the program provided in order to know how much time they have available to run their program on their machines.

Rule 9: The algorithms should take as input a problem file in the format described, and produce as output a solution in the described format. It should do so within the allowed CPU time. Obviously the algorithm should not take account of additional hard coded knowledge about the instance (e.g. introducing instance specific heuristics). The same version of the algorithm must be used for all instances. That is, the algorithm should not "know" which instance it is solving - while your particular algorithm might analyse the problem instance and set parameters accordingly, it should not "recognise" the particular instance. The programmer should not set different parameters for different instances except when the program is doing this automatically, then this is acceptable.

Rule 10: The algorithm can be either deterministic or stochastic. In both cases, participants must be prepared to show that these results are repeatable in the given computer time. In particular, the participants that use a stochastic algorithm should code their program in such a way that the exact run that produced each solution submitted can be repeated (by recording the random seed, etc..). They can try several runs to produce each submitted solution (each with the allowed computer time), but they must be able to repeat the run for any solution submitted.

Rule 11: Participants should submit for each Early and Late Instance the best score found by their algorithm in the specified computer time, by uploading it onto the web site.

Rule 12: Participants should also submit a concise and clear description of their algorithm, so that in principle others can implement it. A template will be made available one month before the end date for this purpose. This is a fundamental part of a participants' submission.

Rule 13: For each track, a set of 5 finalists will be chosen after the competition deadline. Ordering the participants will be based on the scores provided on the Early and Late Instances. The actual list will be based on the ranks of solvers on each single instance. The mean average of the ranks will produce the final place list. More details on how the orderings will be established can be found in Section 5.5.

Rule 14: The finalists will be asked to provide the executable that will be run and tested by the organisers. The finalists' solver will be rerun by the organisers on all instances (including the Hidden ones). It is the responsibility of the participant to ensure all information is provided to enable the organisers to recreate the solution.

The solver submitted by the finalist should require as command-line arguments, input and output file names and, for stochastic solvers only, the random seed. For example (stochastic solver):

```
> my_solver.exe sprint1.txt my_solution.txt 1542955064
```

If appropriate information is not received or indeed the submitted solutions cannot be recreated, another finalist will be chosen from the original participants.

Rule 15: Finalists' eventual place listings will be based on the ranks for each single instance for a set of trials on all instances (including the Hidden ones). As with Rule 10, an explanation of the procedures to be used can be found in Section 5.5.

Rule 16: In some circumstances, finalists may be required to show source code to the organisers. This is simply to check that they have stuck to the rules and will be treated in the strictest confidence.

Rule 17: Entries from participating organising partners will not be permitted. However, results from participants who choose to work on the problems will be presented for comparison.

Appendix E

XML and text dataformat

We provide an XML and text-only data format for representing nurse rostering problem instances (and solutions of the instances) as described in Chapter 3. Next to a description of the schemes of both data formats, a small sample instance and solution are given as an example.

XML

An XML-schema (.xsd) is available to describe the structure of the instances:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:annotation>
  <xs:documentation>
      Schema for personnel scheduling problems.
    </xs:documentation>
 </xs:annotation>
 <xs:element name="SchedulingPeriod">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="StartDate" type="xs:date"/>
    <xs:element name="EndDate" type="xs:date"/>
    <xs:element name="Skills" type="Skills" minOccurs="0"/>
    <xs:element name="ShiftTypes" type="ShiftTypes"/>
    <xs:element name="Patterns" type="Patterns" minOccurs="0"/>
    <xs:element name="Contracts" type="Contracts"/>
    <xs:element name="Employees" type="Employees"/>
    <xs:element name="CoverRequirements" type="CoverRequirements"/>
    <xs:element name="DayOffRequests" type="DayOffRequests" minOccurs="0"/>
    <xs:element name="DayOnRequests" type="DayOnRequests" minOccurs="0"/>
    <xs:element name="ShiftOffRequests" type="ShiftOffRequests" minOccurs="0"/>
    <xs:element name="ShiftOnRequests" type="ShiftOnRequests" minOccurs="0"/>
   </xs:sequence>
   <xs:attribute name="ID" type="xs:string" use="required"/>
```

```
  <xs:attribute name="OrganisationID" type="xs:string" use="optional"/>
 </xs:complexType>
</xs:element>
<xs:complexType name="Skills">
 <xs:sequence>
  <xs:element name="Skill" type="ID" minOccurs="0" maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="ShiftTypes">
 <xs:sequence>
  <xs:element name="Shift" maxOccurs="unbounded">
   <xs:complexType>
    <xs:all>
     <xs:element name="StartTime" type="xs:time"/>
     <xs:element name="EndTime" type="xs:time"/>
     <xs:element name="Description" type="xs:string" minOccurs="0"/>
     <xs:element name="Skills" minOccurs="0">
      <xs:complexType>
       <xs:sequence>
        <xs:element name="Skill" type="ID" maxOccurs="unbounded"/>
       </xs:sequence>
      </xs:complexType>
     </xs:element>
    </xs:all>
    <xs:attribute name="ID" type="ID" use="required"/>
   </xs:complexType>
  </xs:element>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="Patterns">
 <xs:sequence maxOccurs="unbounded">
  <xs:element name="Pattern">
   <xs:complexType>
    <xs:all>
     <xs:element name="PatternEntries">
      <xs:complexType>
       <xs:sequence minOccurs="2" maxOccurs="unbounded">
        <xs:element name="PatternEntry">
         <xs:complexType>
          <xs:all>
           <xs:element name="ShiftType" type="xs:string"/>
           <xs:element name="Day" type="WeekDayOrAny"/>
          </xs:all>
          <xs:attribute name="index"/>
         </xs:complexType>
        </xs:element>
       </xs:sequence>
      </xs:complexType>
     </xs:element>
    </xs:all>
    <xs:attribute name="weight"/>
    <xs:attribute name="ID" type="xs:string"/>
   </xs:complexType>
  </xs:element>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="Contracts">
 <xs:sequence>
  <xs:element name="Contract" maxOccurs="unbounded">
   <xs:complexType>
    <xs:all>
     <xs:element name="SingleAssignmentPerDay"
                 type="WeightOnly" minOccurs="0"/>
```

```
      <xs:element name="MaxNumAssignments"
                  type="OnAndWeight" minOccurs="0"/>
      <xs:element name="MinNumAssignments"
                  type="OnAndWeight" minOccurs="0"/>
      <xs:element name="MaxConsecutiveWorkingDays"
                  type="OnAndWeight" minOccurs="0"/>
      <xs:element name="MinConsecutiveWorkingDays"
                  type="OnAndWeight" minOccurs="0"/>
      <xs:element name="MaxConsecutiveFreeDays"
                  type="OnAndWeight" minOccurs="0"/>
      <xs:element name="MinConsecutiveFreeDays"
                  type="OnAndWeight" minOccurs="0"/>
      <xs:element name="MaxConsecutiveWorkingWeekends"
                  type="OnAndWeight" minOccurs="0"/>
      <xs:element name="MinConsecutiveWorkingWeekends"
                  type="OnAndWeight" minOccurs="0"/>
      <xs:element name="MaxWorkingWeekendsInFourWeeks"
                  type="OnAndWeight" minOccurs="0"/>
      <xs:element name="WeekendDefinition"
                  type="Weekend" minOccurs="0"/>
      <xs:element name="CompleteWeekends"
                  type="WeightOnly" minOccurs="0"/>
      <xs:element name="IdenticalShiftTypesDuringWeekend"
                  type="WeightOnly" minOccurs="0"/>
      <xs:element name="NoNightShiftBeforeFreeWeekend"
                  type="WeightOnly" minOccurs="0"/>
      <xs:element name="TwoFreeDaysAfterNightShifts"
                  type="WeightOnly" minOccurs="0"/>
      <xs:element name="AlternativeSkillCategory"
                  type="WeightOnly" minOccurs="0"/>
      <xs:element name="UnwantedPatterns" minOccurs="0">
       <xs:complexType>
        <xs:sequence minOccurs="0" maxOccurs="unbounded">
         <xs:element name="Pattern" type="ID"/>
        </xs:sequence>
       </xs:complexType>
      </xs:element>
      <xs:element name="Description"/>
     </xs:all>
     <xs:attribute name="ID" type="ID" use="required"/>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
 <xs:complexType name="Employees">
  <xs:sequence>
   <xs:element name="Employee" maxOccurs="unbounded">
    <xs:complexType>
     <xs:all>
      <xs:element name="ContractID" type="ID"/>
      <xs:element name="Name" type="xs:string" minOccurs="0"/>
      <xs:element name="Skills" minOccurs="0">
       <xs:complexType>
        <xs:sequence>
         <xs:element name="Skill" type="xs:string" maxOccurs="unbounded"/>
        </xs:sequence>
       </xs:complexType>
      </xs:element>
     </xs:all>
     <xs:attribute name="ID" type="ID" use="required"/>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
```

```
</xs:complexType>
<xs:complexType name="ConstraintAttributes">
 <xs:attribute name="on" type="xs:boolean" use="optional"/>
 <xs:attribute name="weight" type="xs:nonNegativeInteger" use="optional"/>
</xs:complexType>
<xs:complexType name="OnAndWeight">
 <xs:simpleContent>
  <xs:extension base="xs:nonNegativeInteger">
   <xs:attribute name="on" type="xs:boolean" use="optional"/>
   <xs:attribute name="weight" type="xs:nonNegativeInteger" use="optional"/>
  </xs:extension>
 </xs:simpleContent>
</xs:complexType>
<xs:complexType name="WeightOnly">
 <xs:simpleContent>
  <xs:extension base="xs:boolean">
   <xs:attribute name="weight" type="xs:nonNegativeInteger" use="optional"/>
  </xs:extension>
 </xs:simpleContent>
</xs:complexType>
<xs:complexType name="CoverRequirements">
 <xs:choice minOccurs="0" maxOccurs="unbounded">
  <xs:element name="DayOfWeekCover">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="Day" type="WeekDay"/>
     <xs:element name="Cover" type="Cover" maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
  <xs:element name="DateSpecificCover">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="Date" type="xs:date"/>
     <xs:element name="Cover" type="Cover" maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:complexType>
  </xs:element>
 </xs:choice>
</xs:complexType>
<xs:complexType name="Cover">
 <xs:sequence>
  <xs:choice>
   <xs:element name="Shift" type="ID"/>
  </xs:choice>
  <xs:element name="Preferred" type="xs:nonNegativeInteger" minOccurs="0"/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="DayOffRequests">
 <xs:sequence>
  <xs:element name="DayOff" minOccurs="0" maxOccurs="unbounded">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="EmployeeID" type="ID"/>
     <xs:element name="Date" type="xs:date"/>
    </xs:sequence>
    <xs:attribute name="weight" type="xs:nonNegativeInteger" use="required"/>
   </xs:complexType>
  </xs:element>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="DayOnRequests">
 <xs:sequence>
```

```
  <xs:element name="DayOn" minOccurs="0" maxOccurs="unbounded">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="EmployeeID" type="ID"/>
     <xs:element name="Date" type="xs:date"/>
    </xs:sequence>
    <xs:attribute name="weight" type="xs:nonNegativeInteger" use="required"/>
   </xs:complexType>
  </xs:element>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="ShiftOffRequests">
 <xs:sequence>
  <xs:element name="ShiftOff" minOccurs="0" maxOccurs="unbounded">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="ShiftTypeID" type="ID"/>
     <xs:element name="EmployeeID" type="ID"/>
     <xs:element name="Date" type="xs:date"/>
    </xs:sequence>
    <xs:attribute name="weight" type="xs:nonNegativeInteger" use="required"/>
   </xs:complexType>
  </xs:element>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="ShiftOnRequests">
 <xs:sequence>
  <xs:element name="ShiftOn" minOccurs="0" maxOccurs="unbounded">
   <xs:complexType>
    <xs:sequence>
     <xs:choice>
      <xs:element name="ShiftTypeID" type="ID"/>
     </xs:choice>
     <xs:element name="EmployeeID" type="ID"/>
     <xs:element name="Date" type="xs:date"/>
    </xs:sequence>
    <xs:attribute name="weight" type="xs:nonNegativeInteger" use="required"/>
   </xs:complexType>
  </xs:element>
 </xs:sequence>
</xs:complexType>
<xs:simpleType name="WeekDay">
 <xs:restriction base="xs:string">
  <xs:enumeration value="Sunday"/>
  <xs:enumeration value="Monday"/>
  <xs:enumeration value="Tuesday"/>
  <xs:enumeration value="Wednesday"/>
  <xs:enumeration value="Thursday"/>
  <xs:enumeration value="Friday"/>
  <xs:enumeration value="Saturday"/>
 </xs:restriction>
</xs:simpleType>
<xs:simpleType name="WeekDayOrAny">
 <xs:restriction base="xs:string"/>
</xs:simpleType>
<xs:simpleType name="Weekend">
 <xs:restriction base="xs:string">
  <xs:enumeration value="SaturdaySunday"/>
  <xs:enumeration value="FridaySaturdaySunday"/>
  <xs:enumeration value="FridaySaturdaySundayMonday"/>
  <xs:enumeration value="SaturdaySundayMonday"/>
 </xs:restriction>
</xs:simpleType>
```

```
 <xs:simpleType name="ID">
  <xs:restriction base="xs:string">
   <xs:pattern value="([a-zA-Z0-9._])+"/>
  </xs:restriction>
 </xs:simpleType>
</xs:schema>
```

An example instance using this scheme is given in Section E.1.

Text

The template for the text format is listed below. The value of `boolean` is 0 or 1, `int+` means a positive integer. Furthermore, when an element refers to another element, capital letters are used. E.g. the type of required skill(s) for a shift type is given under `SHIFT_TYPES` by the element '`Skill`', which is of type '`SKILLS.Skill`'. This means that the type of shift type skill is a string (same as Skill under SKILLS) and that the value of `SHIFT_TYPES.Skill` equals a value listed under `SKILLS`.

The percent sign, %, is used in the text template to indicate the different elements of a "mother" element (which is typically written in CAPITAL letters. All lines starting with '%' will be replaced by data.

```
SCHEDULING_PERIOD
% ID <string>,
% StartDate <YYYY-MM-DD>,
% EndDate <YYYY-MM-DD>;

SKILLS = n1
% Skill <string>;

SHIFT_TYPES = n2
% ID <string>,
% Description <string>,
% StartTime <HH:MM:SS>,
% EndTime <HH:MM:SS>,
% NumberOfRequiredSkills <int>,
% RequiredSkill <SKILLS.Skill>; // separate by space, not comma.

CONTRACTS = n3
% ID <string>,
% Description <string>,
% SingleAssignmentPerDay (on|weight) (<boolean>|<int>),
% MaxNumAssignments (on|weight|value) (<boolean>|<int>|<int>),
% MinNumAssignments (on|weight|value) (<boolean>|<int>|<int>),
% MaxConsecutiveWorkingDays (on|weight|value) (<boolean>|<int>|<int>),
% MinConsecutiveWorkingDays (on|weight|value) (<boolean>|<int>|<int>),
% MaxConsecutiveFreeDays (on|weight|value) (<boolean>|<int>|<int>),
% MinConsecutiveFreeDays (on|weight|value) (<boolean>|<int>|<int>),
% MaxConsecutiveWorkingWeekends (on|weight|value) (<boolean>|<int>|<int>),
% MinConsecutiveWorkingWeekends (on|weight|value) (<boolean>|<int>|<int>),
% MaxWorkingWeekendsInFourWeeks (on|weight|value) (<boolean>|<int>|<int>),
% WeekendDefinition <Weekend>
```

```
% CompleteWeekends (on|weight) (<boolean>|<int>),
% Ident.ShiftTypesDuringWeekend (on|weight) (<boolean>|<int>),
% NoNightShiftBeforeFreeWeekend (on|weight) (<boolean>|<int>),
% TwoFreeDaysAfterNightShifts (on|weight) (<boolean>|<int>),
% AlternativeSkillCategory (on|weight) (<boolean>|<int>),
% NumberOfUnwantedPatterns <int>,
% UnwantedPatterns <PATTERNS.ID>;
// separate pattern IDs by space (NOT comma)

PATTERNS = n4
% ID <string>,
% Weight <int>,
% NumberOfShiftTypes <int>,
% ShiftType (<SHIFT_TYPES.ID|Weekday OR Any>);

EMPLOYEES = n5
% ID <string>,
% Name <string>,
% ContractID <CONTRACTS.ID>,
% NumberOfSkills <int>,
% EmployeeSkills <SKILLS.Skill>;

// COVER_REQUIREMENTS

DAY_OF_WEEK_COVER = n6
% Day <WeekDay>,
% Shift <SHIFT_TYPES.ID>,
% Preferred <int+>;

DATE_SPECIFIC_COVER = n7
% Date <YYYY-MM-DD>,
% Shift <SHIFT_TYPES.ID>,
% Preferred <int+>;

DAY_OFF_REQUESTS = n8
% EmployeeID <EMPLOYEES.ID>,
% Date <YYYY-MM-DD>,
% weight <int+>;

DAY_ON_REQUESTS = n9
% EmployeeID <EMPLOYEES.ID>,
% Date <YYYY-MM-DD>,
% weight <int+>;

SHIFT_OFF_REQUESTS = n10
% EmployeeID <EMPLOYEES.ID>,
% Date <YYYY-MM-DD>,
% ShiftTypeID <SHIFT_TYPES.ID>,
% weight <int+>;

SHIFT_ON_REQUESTS = n11
% EmployeeID <EMPLOYEES.ID>,
% Date <YYYY-MM-DD>,
% ShiftTypeID <SHIFT_TYPES.ID>,
% weight <int+>;
```

An example using the above template is given in Section E.2.

E.0.1 Output format

Output format templates, for representing solutions, are also available in XML and text.

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2009 (http://www.altova.com) by SSS (SSS) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Solution">
    <xs:complexType>
      <xs:sequence>
          <xs:element name="SchedulingPeriodID" type="xs:string"/>
          <xs:element name="Competitor"/>
          <xs:element name="SoftConstraintsPenalty" type="xs:nonNegativeInteger"/>
          <xs:element name="Assignment" minOccurs="0" maxOccurs="unbounded">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="Date" type="xs:date"/>
                <xs:element name="Employee" type="xs:string"/>
                <xs:element name="ShiftType" type="xs:string"/>
              </xs:sequence>
            </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

An example of a solution, using this scheme, is given in Section E.3

Text

The text format for solutions is:

```
// The ID of the instance that this is a solution to.
ProblemInstance = <SCHEDULING_PERIOD.ID>
// Name of the competitor. Use the same name for all solutions you provide.
% Competitor <string>,
% SoftConstraintsPenalty <int>;

ASSIGNMENTS = n1 // The number of assignments that follow.
% Date <YYYY-MM-DD>,
% Employee <EMPLOYEES.ID> ,
% ShiftType <SHIFT_TYPES.ID>;
```

An example of a solution, using this format, is given in Section E.4.

E.1 Example of an XML Instance File

Below is an example of an xml instance file.

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<SchedulingPeriod ID="EXAMPLE"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="competition.xsd">
    <StartDate>2010-01-01</StartDate>
    <EndDate>2010-01-29</EndDate>
    <Skills>
        <Skill>Nurse</Skill>
    </Skills>
    <ShiftTypes>
        <Shift ID="E">
            <StartTime>06:30:00</StartTime>
            <EndTime>14:30:00</EndTime>
            <Description>Early</Description>
            <Skills>
                <Skill>Nurse</Skill>
            </Skills>
        </Shift>
        <Shift ID="L">
            <StartTime>14:30:00</StartTime>
            <EndTime>22:30:00</EndTime>
            <Description>Late</Description>
            <Skills>
                <Skill>Nurse</Skill>
            </Skills>
        </Shift>
    </ShiftTypes>
    <Patterns>
        <Pattern ID="0" weight="1">
            <PatternEntries>
                <PatternEntry index="0">
                    <ShiftType>L</ShiftType>
                    <Day>Any</Day>
                </PatternEntry>
                <PatternEntry index="1">
                    <ShiftType>E</ShiftType>
                    <Day>Any</Day>
                </PatternEntry>
            </PatternEntries>
        </Pattern>
    </Patterns>
    <Contracts>
        <Contract ID="0">
            <Description>fulltime</Description>
            <SingleAssignmentPerDay weight="1">true</SingleAssignmentPerDay>
            <MaxNumAssignments on="1" weight="1">16</MaxNumAssignments>
            <UnwantedPatterns>
                <Pattern>0</Pattern>
            </UnwantedPatterns>
        </Contract>
        <Contract ID="1">
            <Description>75_time</Description>
            <SingleAssignmentPerDay weight="1">true</SingleAssignmentPerDay>
            <MaxNumAssignments on="1" weight="1">12</MaxNumAssignments>
            <UnwantedPatterns>
                <Pattern>0</Pattern>
            </UnwantedPatterns>
```

```
        </Contract>
    </Contracts>
    <Employees>
        <Employee ID="0">
            <ContractID>0</ContractID>
            <Name>0</Name>
            <Skills>
                <Skill>Nurse</Skill>
            </Skills>
        </Employee>
        <Employee ID="1">
            <ContractID>0</ContractID>
            <Name>1</Name>
            <Skills>
                <Skill>Nurse</Skill>
            </Skills>
        </Employee>
    </Employees>
    <CoverRequirements>
        <DayOfWeekCover>
            <Day>Monday</Day>
            <Cover>
                <Shift>E</Shift>
                <Preferred>2</Preferred>
            </Cover>
        </DayOfWeekCover>
        <DayOfWeekCover>
            <Day>Tuesday</Day>
            <Cover>
                <Shift>L</Shift>
                <Preferred>2</Preferred>
            </Cover>
        </DayOfWeekCover>
    </CoverRequirements>
    <DayOffRequests>
        <DayOff weight="1">
            <EmployeeID>0</EmployeeID>
            <Date>2010-01-03</Date>
        </DayOff>
    </DayOffRequests>
    <ShiftOffRequests>
        <ShiftOff weight="1">
            <ShiftTypeID>L</ShiftTypeID>
            <EmployeeID>1</EmployeeID>
            <Date>2010-01-15</Date>
        </ShiftOff>
    </ShiftOffRequests>
</SchedulingPeriod>
```

E.2 Example of a Text Instance File

Below is an example of a text instance file based on the same instance as the xml example in Appendix E.1.

```
// Comments can be added by prefixing the comment with '//'.

// Text following '//' should not be treated.
```

```
//////////////////////////////////////////////////////////////////
SCHEDULING_PERIOD;
//////////////////////////////////////////////////////////////////
EXAMPLE, 2010-01-01, 2010-01-29;

//////////////////////////////////////////////////////////////////
SKILLS = 1;
//////////////////////////////////////////////////////////////////
Nurse;

//////////////////////////////////////////////////////////////////
SHIFT_TYPES = 2;
//////////////////////////////////////////////////////////////////
E, Early, 06:30:00, 14:30:00, 1, Nurse;
L, Late, 14:30:00, 22:30:00, 1, Nurse;

//////////////////////////////////////////////////////////////////
CONTRACTS = 2;
//////////////////////////////////////////////////////////////////
0, fulltime, (1|1), (1|1|16), , , , , , , , , , , , , , , , , 0, , 1, 0;
1, 75_time, (1|1), (1|1|12), , , , , , , , , , , , , , , , , 0, , 1, 0;

//////////////////////////////////////////////////////////////////
PATTERNS = 1;
//////////////////////////////////////////////////////////////////
0, 1, 0, L Any E Any;

//////////////////////////////////////////////////////////////////
EMPLOYEES = 2;
//////////////////////////////////////////////////////////////////
0, 0, 0, 1, Nurse;
1, 1, 0, 1, Nurse;

//////////////////////////////////////////////////////////////////
```

```
DAY_OF_WEEK_COVER  = 2;

//////////////////////////////////////////////////////////////////////

Monday, , E, 2;

Tuesday, , L, 2;

//////////////////////////////////////////////////////////////////////

DATE_SPECIFIC_COVER  = 0;

//////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////

DAY_OFF_REQUESTS = 1;

//////////////////////////////////////////////////////////////////////

0, 2010-01-03, 1;

//////////////////////////////////////////////////////////////////////

DAY_ON_REQUESTS = 0;

//////////////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////

SHIFT_OFF_REQUESTS  = 1;

//////////////////////////////////////////////////////////////////////

1, 2010-01-15, L, 1;

//////////////////////////////////////////////////////////////////////

SHIFT_ON_REQUESTS  = 0;

//////////////////////////////////////////////////////////////////////
```

E.3 Example of a XML Solution File

```
<Solution>
  <SchedulingPeriodID>EXAMPLE</SchedulingPeriodID>
  <Competitor>PlanCo</Competitor>
  <SoftConstraintsPenalty>3</SoftConstraintsPenalty>
  <Assignment>
    <Date>2010-01-01</Date>
    <Employee>1</Employee>
```

```
    <ShiftType>L</ShiftType>
  </Assignment>
  <Assignment>
    <Date>2010-01-01</Date>
    <Employee>2</Employee>
    <ShiftType>E</ShiftType>
  </Assignment>
  <Assignment>
    <Date>2010-01-01</Date>
    <Employee>3</Employee>
    <ShiftType>D</ShiftType>
  </Assignment>
  <Assignment>
    <Date>2010-01-01</Date>
    <Employee>4</Employee>
    <ShiftType>N</ShiftType>
  </Assignment>
  <Assignment>
    <Date>2010-01-01</Date>
    <Employee>5</Employee>
    <ShiftType>E</ShiftType>
  </Assignment>
  <Assignment>
    <Date>2010-01-01</Date>
    <Employee>6</Employee>
    <ShiftType>N</ShiftType>
  </Assignment>

  ....

</Solution>
```

E.4 Example of a Text Solution file

Here is an example of a text solution file based on the same solution as in the xml example in Appendix E.3.

```
//////////////////////////////////////////////////////////////////
SOLUTION = EXAMPLE;
//////////////////////////////////////////////////////////////////
PlanCo, 3;

//////////////////////////////////////////////////////////////////
ASSIGNMENTS = 6;
//////////////////////////////////////////////////////////////////
2010-01-01, 1, L;
2010-01-01, 2, E;
2010-01-01, 3, D;
2010-01-01, 4, N;
2010-01-01, 5, E;
2010-01-01, 6, N;

...
```

Bibliography

S. Acharyya. A SAT Approach for Solving The Nurse Scheduling Problem. In *IEEE Region 10 Conference*, 2008.

S. Aknine, S. Pinson, and M. F. Shakun. An extended multi-agent negotiation protocol. *Autonomous Agents and Multi-Agent Systems*, 8(1):5–45, Jan. 2004.

B. An, V. Lesser, and K. Sim. Strategic agents for multi-resource negotiation. *Autonomous Agents and Multi-Agent Systems*, 23(1):114–153, 2011.

J. Argelich and F. Manyà. Exact max-sat solvers for over-constrained problems. *Journal of Heuristics*, 12:375–392, 2006.

R. Asín, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell. Cardinality networks and their applications. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing*, SAT '09, pages 167–180, 2009.

L. M. Ausubel and P. Milgrom. Ascending proxy auctions. Discussion Papers 03-035, Stanford Instititute for Economic Policy Research, Aug. 2004.

O. Bailleux and Y. Boufkhad. Efficient CNF Encoding of Boolean Cardinality Constraints. In F. Rossi, editor, *Principles and Practice of Constraint Programming - CP 2003*, volume 2833 of *Lecture Notes in Computer Science*, pages 108–122. Springer Berlin / Heidelberg, 2003.

J. Bard and H. Purnomo. Incremental changes in the workforce to accommodate changes in demand. *Health Care Management Science*, 9:71–85, 2006.

B. Bilgin, P. De Causmaecker, S. Haspeslagh, T. Messelis, and G. Vanden Berghe. Hardness studies for nurse rostering problems. In *LION, Trento, Italy, 14-18 January 2009*, Jan. 2009.

B. Bilgin, P. De Causmaecker, B. Rossie, and G. Vanden Berghe. Local search neighbourhoods for dealing with a novel nurse rostering model. *Annals of Operations Research*, 194:33–57, 2012.

P. Brucker, E. Burke, T. Curtois, R. Qu, and G. Vanden Berghe. A shift sequence based approach for nurse scheduling and a new benchmark dataset. *Journal of Heuristics*, 16:559–573, 2010.

E. Burke and T. Curtois. New computational results for nurse rostering benchmark instances. technical report, 2011. Technical report, School of Computer Science, University of Nottingham, 2011.

E. K. Burke, P. De Causmaecker, S. Petrovic, and G. Vanden Berghe. Fitness Evaluation for Nurse Scheduling Problems. In *Proceedings of the Congress on Evolutionary Computation (CEC2001)*, pages 1139–1146, 2001.

E. K. Burke, P. De Causmaecker, G. Vanden Berghe, and H. Van Landeghem. The State of the Art of Nurse Rostering. *Journal of Scheduling*, 7(6):441–499, 2004.

E. K. Burke, T. Curtois, G. Post, R. Qu, and B. Veltman. A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem. *European Journal of Operational Research*, 188(2):330 – 341, 2008.

R. Buttner. A classification structure for automated negotiations. In *Web Intelligence and Intelligent Agent Technology Workshops, 2006. WI-IAT 2006 Workshops. 2006 IEEE/WIC/ACM International Conference on*, pages 523 –530, dec. 2006.

M. Cadoli and A. Schaerf. Compiling problem specifications into SAT. *Artificial Intelligence*, 162(1-2):89–120, Feb. 2005.

E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 11(1), Sept. 1971.

S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, STOC '71, pages 151–158, 1971.

M.-C. Côté, B. Gendron, C.-G. Quimper, and L.-M. Rousseau. Formal languages for integer programming modeling of shift scheduling problems. *Constraints*, 16(1):54–76, 2011.

P. I. Cowling, D. Ouelhadj, and S. Petrovic. A multi-agent architecture for dynamic scheduling of steel hot rolling. *Journal of Intelligent Manufacturing*, 14(5):457–470, 2003.

P. Cramton, Y. Shoham, and R. Steinberg, editors. *Combinatorial Auctions*. MIT Press, 2006.

P. De Causmaecker and G. Vanden Berghe. A categorisation of nurse rostering problems. *Journal of Scheduling*, 14:3–16, 2011.

P. De Causmaecker and G. Vanden Berghe. Towards a reference model for timetabling and rostering. *Annals of Operations Research*, 194:167–176, 2012.

P. Demeester, W. Souffriau, P. De Causmaecker, and G. Vanden Berghe. A hybrid tabu search algorithm for automatically assigning patients to beds. *Artificial Intelligence in Medicine*, 48(1):61 – 70, 2010.

L. Di Gaspero, S. Mizzaro, and A. Schaerf. A multiagent architecture for distributed course timetabling. In *Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT-2004)*, August 2004.

T. M. Dias, D. F. Ferber, C. C. De Souza, and A. V. Moura. Constructing nurse schedules at large hospitals. *International Transactions in Operational Research*, 10(3):245–265, 2003.

L. Duan, M. K. Doğru, U. Özen, and J. C. Beck. A negotiation framework for linked combinatorial optimization problems. *Autonomous Agents and Multi-Agent Systems*, Apr. 2011.

A. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *European Journal Of Operational Research*, 153(1):3–27, 2004.

M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

P. Faratin, C. Sierra, and N. R. Jennings. Using similarity criteria to make issue trade-offs in automated negotiations. *Artificial Intelligence*, 142(2):205–237, 2002.

S. Fatima, M. Wooldridge, and N. Jennings. Multi-issue negotiation with deadlines. *Journal of Artificial Intelligence Research*, 27(1):381–417, 2006.

S. Fatima, M. Wooldridge, and N. R. Jennings. An analysis of feasible solutions for multi-issue negotiation involving nonlinear utility functions. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '09, pages 1041–1048, 2009.

S. S. Fatima, M. Wooldridge, and N. R. Jennings. An agenda-based framework for multi-issue negotiation. *Artif. Intell.*, 152(1):1–45, 2004.

D. E. Ferguson. Fibonaccian searching. *Commun. ACM*, 3(12):648–, Dec. 1960.

C. A. Glass and R. A. Knight. The nurse rostering problem: A critical appraisal of the problem structure. *European Journal of Operational Research*, 202(2): 379–389, 2010.

M. Gobert, L. Alvarez-Irusta, G. Berckmans, M. Coêffé, O. Dardenne, S. Ghysselinckx, T. Van Durme, D. Myny, D. Debergh, F. Gossiaux, T. Vaes, T. Vandenbrande, P. Van Pelt, W. Ver Heyen, and M. Lamberts. Optimalisatie van het roosterbeleid en de mobiele equipe voor onmiddellijke vervanging: naar meer gezonde en voorspelbare roosters door de efficiënte inzet van de mobiele equipe, 2009.

T. Groves. Incentives in Teams. *Econometrica*, 41(4):617–631, 1973.

R. H. Guttman, A. G. Moukas, and P. Maes. Agent-mediated electronic commerce: a survey. *Knowl. Eng. Rev.*, 13(2):147–159, July 1998.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11: 10–18, November 2009.

A. K. Hutzschenreuter, P. A. N. Bosman, and J. A. La Poutré. Evolutionary Multiobjective Optimization For Dynamic Hospital Resource Management. In M. Ehrgott, C. M. Fonseca, X. Gandibleux, J.-K. Hao, and M. Sevaux, editors, *Proceedings of Evolutionary Multi-Criterion Optimization 2009*, volume 5467 of *Lecture Notes in Computer Science*, pages 320–334. Springer, 2009.

INRC2010. First international nurse rostering competition website. URL: `http://www.kuleuven-kortrijk.be/nrpcompetition`, 2010.

T. Ito, M. Klein, and H. Hattori. A multi-issue negotiation protocol among agents with nonlinear utility functions. *Multiagent Grid Syst.*, 4(1):67–83, 2008.

M. O. Jackson. Mechanism theory, 2001.

N. Jennings, P. Faratin, A. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge. Automated Negotiation: Prospects, Methods and Challenges. *International Journal of Group Decision and Negotiation*, 10(2):199–215, 2001.

C. Jonker, V. Robu, and J. Treur. An agent architecture for multi-attribute negotiation using incomplete preference information. *Autonomous Agents and Multi-Agent Systems*, 15(2):221–252, Jan. 2007.

R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

M. Klein, P. Faratin, H. Sayama, and Y. Bar-Yam. Protocols for negotiating complex contracts. *Intelligent Systems, IEEE*, 18(6):32–38, 2003.

D. E. Knuth. *The art of computer programming, volume 3: (2nd ed.) sorting and searching*. 1998.

S. Koenig, C. Tovey, M. Lagoudakis, V. Markakis, and D. Kempe. The power of sequential single-item auctions for agent coordination. In *Proceedings of the National Conference on Artificial Intelligence*, 2006.

S. Kraus. *Strategic negotiation in multiagent environments*. MIT Press, 2001.

G. Lai and K. Sycara. A Generic Framework for Automated Multi-attribute Negotiation. *Group Decision and Negotiation*, 18(2):169–187, 2009.

G. Lai, C. Li, K. Sycara, and J. Giampapa. Literature review on multi-attribute negotiations. Technical report, 2004.

H. C. Lau, S. F. Cheng, T. Y. Leong, J. H. Park, and Z. Zhao. Multi-Period Combinatorial Auction Mechanism for Distributed Resource Allocation and Scheduling. In *Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, IAT '07, pages 407–411, 2007.

R. Y. K. Lau, M. Tang, and O. Wong. Towards genetically optimised responsive negotiation agents. In *Intelligent Agent Technology, 2004. (IAT 2004). Proceedings. IEEE/WIC/ACM International Conference on*, pages 295–301, 2004.

K. Leyton-Brown, E. Nudelman, and Y. Shoham. Learning the empirical hardness of optimization problems: The case of combinatorial auctions. In P. Van Hentenryck, editor, *Principles and Practice of Constraint Programming - CP 2002*, volume 2470 of *Lecture Notes in Computer Science*, pages 91–100. Springer Berlin / Heidelberg, 2006.

M. Lopez-Carmona, I. Marsa-Maestre, M. Klein, and T. Ito. Addressing stability issues in mediated complex contract negotiations for constraint-based, non-monotonic utility spaces. *Autonomous Agents and Multi-Agent Systems*, pages 1–51, 2010.

M. Lopez-Carmona, I. Marsa-Maestre, E. De La Hoz, and J. Velasco. A region-based multi-issue negotiation protocol for nonmonotonic utility spaces. *Computational Intelligence*, 27(2):166–217, 2011.

X. Luo, N. R. Jennings, N. Shadbolt, H. Leung, and J. Lee. A fuzzy constraint based model for bilateral, multi-issue negotiations in semi-competitive environments. *Artificial Intelligence*, 148(1-2):53–102, Aug. 2003.

B. Maenhout and M. Vanhoucke. An evolutionary approach for the nurse rerostering problem. *Computers & Operations Research*, 38(10):1400 – 1411, 2011.

I. Marsa-Maestre, M. Lopez-Carmona, and J. Velasco. Improving trade-offs in bilateral negotiations under complete and incomplete information settings. In T. Bui, T. Ho, and Q. Ha, editors, *Intelligent Agents and Multi-Agent Systems*, volume 5357 of *Lecture Notes in Computer Science*, pages 275–286. Springer Berlin / Heidelberg, 2008.

I. Marsa-Maestre, M. Lopez-Carmona, J. A. Carral, and G. Ibanez. *A Recursive Protocol for Negotiating Contracts Under Non-monotonic Preference Structures*. June 2011.

B. McCollum. A perspective on bridging the gap between theory and practice in university timetabling. In E. Burke and H. Rudová, editors, *Practice and Theory of Automated Timetabling VI*, volume 3867 of *Lecture Notes in Computer Science*, pages 3–23. Springer Berlin / Heidelberg, 2007.

B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A. J. Parkes, L. Di Gaspero, R. Qu, and E. K. Burke. Setting the Research Agenda in Automated Timetabling: The Second International Timetabling Competition. *INFORMS JOURNAL ON COMPUTING*, 2009.

P. Milgrom. Putting auction theory to work: The simultaneous ascending auction. *Journal of Political Economy*, 108(2):245–272, 2000.

M. Moz and M. Vaz Pato. An integer multicommodity flow model applied to the rerostering of nurse schedules. *Annals of Operations Research*, 119: 285–301, 2003.

M. Moz and M. Vaz Pato. A genetic algorithm approach to a nurse rerostering problem. *Computers & Operations Research*, 34(3):667 – 691, 2007.

D. Myny, T. Defloor, L. Alvarez-Irusta, D. Annys, F. Demeyere, I. DeVreese, V. Proenca, M. Vandermolen, K. Vanderwee, A. Van Hecke, and M. Gobert. Eindrapport welame, 2010.

E. Nudelman, K. Leyton-Brown, H. Hoos, A. Devkar, and Y. Shoham. Understanding random sat: Beyond the clauses-to-variables ratio. In M. Wallace, editor, *Principles and Practice of Constraint Programming - CP 2004*, volume 3258 of *Lecture Notes in Computer Science*, pages 438–452. Springer Berlin / Heidelberg, 2004.

T. Osogami and H. Imai. Classification of various neighborhood operations for the nurse scheduling problem. Technical Report 135, The Institute of Statistical Mathematics, 2000.

S. Petrovic and G. Vanden Berghe. A comparison of two approaches to nurse rostering problems. *Annals of Operations Research*, 194:365–384, 2012.

A. Ragone, T. Noia, E. Sciascio, and F. M. Donini. Logic-based automated multi-issue bilateral negotiation in peer-to-peer e-marketplaces. *Autonomous Agents and Multi-Agent Systems*, 16(3):249–270, Mar. 2008.

H. Raiffa. *The Art and Science of Negotiation*. Harvard University Press, 1982.

A. Riise and E. Burke. Local search for the surgery admission planning problem. *Journal of Heuristics*, 17:389–414, 2011.

J. S. Rosenschein and G. Zlotkin. *Rules of Encounter - Designing Conventions for Automated Negotiation among Computers*. MIT Press, 1994.

A. Rubinstein. Perfect equilibrium in a bargaining model. *Econometrica*, 50 (1):pp. 97–109, 1982.

A. Schaerf and L. Di Gaspero. Measurability and reproducibility in university timetabling research: Discussion and proposals. In E. Burke and H. Rudová, editors, *Practice and Theory of Automated Timetabling VI*, volume 3867 of *Lecture Notes in Computer Science*, pages 40–49. Springer Berlin / Heidelberg, 2007.

W. Shen, Q. Hao, H. J. Yoon, and D. H. Norrie. Applications of agent-based systems in intelligent manufacturing: An updated review. *Advanced Engineering Informatics*, 20(4):415 – 431, 2006.

C. Sinz. Towards an optimal cnf encoding of boolean cardinality constraints. In *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming (CP 2005)*, pages 827–831, 2005.

P. Smet, B. Bilgin, P. De Causmaecker, and G. Vanden Berghe. Modelling and evaluation issues in nurse rostering. *Annals of Operations Research*, pages 1–24, 2012a.

P. Smet, S. Martin, D. Ouelhadj, E. Özcan, and G. Vanden Berghe. Investigation of fairness measures for nurse rostering. In *Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling*, 2012b.

R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Trans. Comput.*, 29(12):1104–1113, Dec. 1980.

V. Tamma, S. Phelps, I. Dickinson, and M. Wooldridge. Ontologies for supporting negotiation in e-commerce. *Engineering Applications of Artificial Intelligence*, 18(2):223 – 236, 2005.

P. Valckenaers, Hadeli, B. Saint Germain, P. Verstraete, and H. Van Brussel. Emergent short-term forecasting through ant colony engineering in coordination and control systems. *Advanced Engineering Informatics*, 20 (3):261 – 278, 2006.

C. Valouxis, C. Gogos, G. Goulas, P. Alefragis, and E. Housos. A systematic two phase approach for the nurse rostering problem. *European Journal of Operational Research*, 219(2):425 – 433, 2012.

J. Vandeurzen. Werk maken van werk in de zorgsector. Actieplan ter bevordering van de werkgelegenheid in de zorgsector, 2010.

M. Vanhoucke and B. Maenhout. On the characterization and generation of nurse scheduling problem instances. *European Journal of Operational Research*, 196(2):457 – 467, 2009.

M. Vaz Pato and M. Moz. Solving a bi-objective nurse rerostering problem by using a utopic pareto genetic heuristic. *Journal of Heuristics*, 14:359–374, 2008.

W. Vickrey. Counterspeculation, Auctions, and Competitive Sealed Tenders. *The Journal of Finance*, 16(1):8–37, 1961.

J. Von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

D. M. Warner. Scheduling nursing personnel according to nursing preference: A mathematical programming approach. *Operations Research*, 24(5):842–856, 1976.

G. Weil, K. Heus, P. François, and M. Poujade. Constraint programming for nurse scheduling. *Engineering in Medicine and Biology Magazine, IEEE*, 14 (4):417 –422, 1995.

M. P. Wellman, W. E. Walsh, P. R. Wurman, and J. K. MacKie-Mason. Auction protocols for decentralized scheduling. *Games and Economic Behavior*, 35 (1-2):271 – 303, 2001.

M. P. Wellman, J. K. MacKie-Mason, D. M. Reeves, and S. Swaminathan. Exploring bidding strategies for market-based scheduling. In *Proceedings of the 4th ACM conference on Electronic commerce*, EC '03, pages 115–124, 2003.

M. J. Wooldridge. *An Introduction to MultiAgent Systems (2. ed.)*. Wiley, 2009.

M. Wu, M. de Weerdt, and H. La Poutré. Efficient Methods for Multi-agent Multi-issue Negotiation: Allocating Resources. In *Proceedings of the 12th International Conference on Principles of Practice in Multi-Agent Systems*, PRIMA '09, pages 97–112, 2009.

L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Satzilla: portfolio-based algorithm selection for sat. *J. Artif. Int. Res.*, 32(1):565–606, June 2008.

G. Zlotkin and J. S. Rosenschein. Mechanism design for automated negotiation, and its application to task oriented domains. *Artificial Intelligence*, 86(2): 195 – 244, 1996.

List of publications

International journal papers

- S. Haspeslagh, P. De Causmaecker, A. Schaerf and M. Stølevik, The first international nurse rostering competition 2010, Annals of Operations Research, pages 1-16, online first, doi: 10.1007/s10479-012-1062-0
- S. Haspeslagh and P. De Causmaecker, Bridging the gap between short and mid term nurse rostering through a negotiation protocol, submitted to Journal of Scheduling

International conference papers published in the conference proceedings

- S. Haspeslagh, P. De Causmaecker and G. Vanden Berghe, Distributed decision making in hospital wide nurse rostering problems, MISTA 2007, Proceedings of the 3th Multidisciplinary International Scheduling Conference: Theory and Applications, Paris, pages 192-199

Abstracts and presentations at international conferences

- S. Haspeslagh, P. De Causmaecker and G. Vanden Berghe, Framework for negotiation in Distributed Nurse Rostering Problems, PATAT 2006, Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling, Brno, pages 426-431
- B. Bilgin, P. De Causmaecker, S. Haspeslagh, T. Messelis and G. Vanden Berghe, Hardness studies for nurse rostering problems, LION 2009, Trento, pages 1-4

- S. Haspeslagh, P. De Causmaecker and G. Vanden Berghe, A multi-agent system handling personnel shortages in hospitals, MISTA 2009, Proceedings of the 4th Multidisciplinary International Scheduling Conference: Theory and Applications, Dublin, pages 693-695
- S. Haspeslagh, P. De Causmaecker, A. Schaerf and M. Stølevik, The First International Nurse Rostering Competition 2010, PATAT 2010, Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling, Dublin, pages 498-501
- S. Haspeslagh and P. De Causmaecker, Pareto optimal negotiation for nurse rostering problems, MISTA 2011, Proceedings of the 5th Multidisciplinary International Scheduling Conference: Theory and Applications, Phoenix, pages 1-5

Abstracts, papers and presentations at national conferences

- S. Haspeslagh and P. De Causmaecker, ORBEL 21, Benchmarks for the Nurse Rostering Problems, Luxembourg, 2007, pages 1-2
- T. Messelis, S. Haspeslagh, B. Bilgin, P. De Causmaecker and G. Vanden Berghe, Towards prediction of algorithm performance in real world optimisation problems, BNAIC 2009, Proceedings of the 21st Benelux Conference on Artificial Intelligence, Eindhoven, pages 177-183
- R. Lagatie, S. Haspeslagh and P. De Causmaecker, Negotiation protocols in distributed nurse rostering, BNAIC 2009, Proceedings of the 21st Benelux Conference on Artificial Intelligence, Eindhoven, pages 145-152
- T. Messelis, S. Haspeslagh, P. De Causmaecker, B. Bilgin and G. Vanden Berghe, Hardness studies for nurse rostering problems, ORBEL 23, Proceedings of the 23th Annual conference of the Belgian Operations Research Society, Leuven
- T. Messelis, S. Haspeslagh, P. De Causmaecker, On expressing nurse rostering constraints as propositional satisfiability problems, ORBEL 24, Proceedings of the 24th Annual conference of the Belgian Operations Research Society, Liège, pages 1-2
- P. De Causmaecker and S. Haspeslagh, Algorithm analysis and higher level decision support for nurse rostering, ORBEL 26, Proceedings of the 26th Annual conference of the Belgian Operations Research Society, Brussel, pages 1-2

Arenberg Doctoral School of Science, Engineering & Technology
Faculty of Science
Department of Computer Science
CODeS Research Group
Etienne Sabbelaan 53
B-8500 Kortrijk

KATHOLIEKE UNIVERSITEIT LEUVEN

ASSOCIATIE K.U.LEUVEN