Linköping Studies in Science and Technology. Dissertations.
No. 1421

Contributions within Two Topics in Integer Programming:

Nurse scheduling and Column Generation

Elina Rönnberg

Linköpings universitet
INSTITUTE OF TECHNOLOGY

Department of Mathematics, Division of Optimization
Linköping University, SE-581 83 Linköping, Sweden

Linköping 2012

Contributions within two topics in integer programming:
nurse scheduling and column generation
Elina Rönnberg

ISBN 978-91-7519-975-7 ISSN 0345-7524

Printed in Sweden by LiU-Tryck, 2012.

Abstract

Integer programming can be used to provide solutions to complex decision and planning problems occurring in a wide variety of situations. The application of integer programming to solve real world problems requires a modelling phase in which the problem at hand is translated into a mathematical description of the problem, and a solution phase that aims at developing methods for producing solutions to the mathematical formulation of the problem.

The first two papers of this thesis have their focus on the modelling phase, and the application of integer programming for solving nurse scheduling problems. Common to both papers is that the research has been conducted in collaboration with health care representatives, and that the models presented can be used for providing schedules that can be used by nurses. In the latter paper, a meta-heuristic approach is suggested for providing the schedules.

The last three papers address method development and specifically the design of column generation methods. The first of these papers presents optimality conditions that are useful in methods where columns are generated using dual solutions that are not necessarily optimal with respect to a linear programming relaxation, and the usefulness of these conditions are illustrated by examples from the literature.

Many applications of column generation yield master problems of a set partitioning type, and the fourth and fifth paper present methodologies for solving such problems. The characteristics of these methodologies are that all solutions derived are feasible and integral, where the preservation of integrality is a major distinction from other column generation methods presented in the literature.

Populärvetenskaplig sammanfattning

Bidrag inom heltalsoptimering: sköterskeschemaläggning och kolumngenereringsmetoder

Inom svensk vård och omsorg är schemaläggningen av sköterskor en stor utmaning, speciellt på de enheter som behöver bemanning dygnet runt under årets alla dagar. En schemaläggare behöver ta hänsyn till många olika och ofta motstridiga krav såsom:

- Verksamhetens behov av rätt kompetens vid rätt tidpunkt
- Arbetsrättslig lagstiftning som reglerar hur ett schema får se ut
- Att vissa schemakonstruktioner fungerar bättre än andra
- Varje medarbetares anställningsavtal, individuella behov och önskemål
- Rättvisa mellan medarbetarna

I dagsläget förbrukas mycket tid på att manuellt ta fram scheman, och dessa uppfyller sällan alla de krav som ställs. En dator har en mycket högre förmåga än en människa när det gäller att ta hänsyn till många aspekter samtidigt, och därför lämpar den sig bra för att lägga scheman. För att de scheman som en dator lägger ska kunna användas i praktiken krävs dock att den instrueras på rätt sätt, och däri består en stor utmaning – en dator kan på egen hand inte förstå vad som är ett bra schema.

Ena delen av denna avhandling handlar om hur man med hjälp av heltalsoptimering kan instruera en dator att lägga skräddarsydda och högkvalitativa scheman som kan användas av svenska vård- och omsorgsenheter. Resultatet av vår forskning är en programvara som används av företaget Schemagi AB.

Vid tillämpning av heltalsoptimering för att lösa problem av praktisk betydelse krävs dels att problemet kan formuleras matematiskt, dels att datorn givet denna matematiska formulering kan ta fram bra lösningar. Forskningen inom området påbörjades i mitten av 1900-talet och stora framsteg har sedan dess gjorts med hänsyn till vilka problem som kan lösas, men också med avseende på kvalitén på lösningarna. För att forskningen inom heltalsoptimering ska fortsätta framåt så krävs dels att heltalsoptimering tillämpas på verkliga problem, som är fallet med vårt arbete inom sköterskeschemaläggning, dels att förbättrade beräkningsmetoder tas fram. Den andra delen av avhandlingen är ett bidrag inom sådan metodutveckling.

Acknowledgements

Thanks to my supervisor Torbjörn Larsson for excellent guidance and support. I also thank my present and former colleagues at the Department of Mathematics, other people with whom I have collaborated, my family, and friends.

The research presented in Papers [III]–[V] of this thesis has been carried out with financial support from the Swedish Research Council.

Elina Rönnberg
Linköping, December 2011

Papers

The following papers are appended and will be referred to by their Roman numerals.

[I] E. Rönnberg and T. Larsson. Automating the self-scheduling process of nurses in Swedish healthcare: A pilot study. *Health Care Management Science*, 13: 35–53, 2010.

[II] E. Rönnberg and T. Larsson and A. Bertilsson. Automatic scheduling of nurses: What does it take in practice? To appear in *Systems Analysis Tools for Better Healthcare Delivery*, Editors P. Pardalos, P. Georgiev and P. Papajorgji, Springer Science + Business Media, New York, NY, 2012.

[III] E. Rönnberg and T. Larsson. Column generation using non-optimal dual solutions: Optimality conditions and over-generation. Manuscript submitted for publication.

[IV] E. Rönnberg and T. Larsson. Column generation in the integral simplex method. *European Journal of Operational Research*, 192:333–342, 2009.

[V] E. Rönnberg and T. Larsson. All-integer column generation: A meta-heuristic for set partitioning problems. Manuscript submitted for publication.

Contents

Introduction

1 Staff scheduling ... 3
2 An introduction to integer programming column generation ... 7
2.1 Linear programming column generation ... 7
2.2 Column generation for set partitioning problems ... 8
2.3 Further reading and historical perspective ... 12
3 On the set partitioning problem and quasi-integrality ... 15
3.1 The set partitioning problem ... 16
3.2 The quasi-integrality property ... 17
3.3 The integral simplex method ... 19
3.4 All-integer pivots ... 20
3.5 Concluding remarks ... 30
4 Thesis contributions and future work ... 31

Appended papers

I Automating the self-scheduling process of nurses in Swedish healthcare: A pilot study **41**

II Automatic scheduling of nurses: What does it take in practice? **75**

III Column generation using non-optimal dual solutions: Optimality conditions and over-generation **105**

IV Column generation in the integral simplex method **121**

V All-integer column generation: A meta-heuristic for set partitioning problems **139**

Introduction

Integer programming can be used to model and solve complex decision and planning problems occurring in a wide variety of situations, such as in logistics, infrastructure planning, and personnel and activity scheduling. One distinguishing characteristic of integer programming, which is one field within the area of optimisation, is that the decisions to be made are quantified by integers values, like 'build this factory (= 1) or not (= 0)' or 'the number of cars to be sent from A to B'.

This thesis is restricted to linear integer programs, which can be formulated as

$$\begin{aligned} z^* = \quad & \min \quad cx \\ & \text{s.t.} \quad Ax \geq b, \\ & \qquad x \geq 0 \text{ and integer}, \end{aligned}$$

where x is a column vector of n variables, c is a row vector of n elements, A is an n by m matrix, and b is a column vector of m elements. If only some of the variables are required to be integral, the problem is called a mixed integer program.

Applying integer programming to a real world application can be said to involve two phases; one – referred to as the modelling phase – that involves the interaction with the owners of an optimisation problem to be solved, and the other – referred to as the solution phase – that involves using computer software to produce a solution to the problem stated in the modelling phase.

In the modelling phase, all aspects that need to be considered to solve the problem must be stipulated and quantified in order to create a mathematical model describing the problem. When faced with a complex real world problem, it can be a real challenge to construct a solvable model that mirrors the reality and whose solutions are of practical interest. Considering this trade-off between the model being solvable and the amount of detail in the model, the aspects which should be included in the model, which should be discarded, and which should be replaced by some kind of approximation must be decided. It is also of crucial importance to resolve how to provide the model with input data of good enough quality for the solution to be sound.

The aim of the solution phase is to provide a solution to the problem described by the mathematical model constructed in the modelling phase. Nowadays, this can fairly often be done by the appropriate use of standard optimisation software. When this is not practically possible, the challenge is to develop new methods or to tailor already existing ones to solve the problem in an efficient way.

After obtaining one or several solutions to the problem at hand, it is wise to return to the modelling phase and present these suggestions to the problem owners in order to get their feedback on the validity and the quality of the solutions. If dealing with a complex problem, it is natural that the feedback obtained will result in changes in the model. This in turn can result in changes to the solution method, giving rise to an iterative process in which the interaction between modelling issues and implementation issues is often present. The strong relationship between how the model is formulated and which solution methods that can, or should, be used, often creates an inevitable dependency between the phases. After finishing this iterative process, the resulting model and solution method can be put to actual use, for example as a human-interactive decision-support tool, or as a tool that repeatedly and automatically provides and uses solutions to problem instances generated in real time.

Taking a closer look at the solution phase, the level of novelty required to solve a problem can range from combining well known solution strategies in a new situation, to applying an idea that creates a completely new area of research. In integer programming research, progress needs to be made within both the modelling phase and the solution phase, which as indicated above, offer challenges of completely different types.

Relating the above description to the contributions of this thesis, Papers [I] and [II] describe the application of integer programing for solving real world nurse scheduling problems and involve both the modelling and the solution phases, with an emphasis on the former. Papers [III], [IV], and [V] address the solution phase only, and contribute to the development of solution strategies that can be applied to problem structures that arise fairly often in large scale integer programming.

Outline of the thesis introduction

Chapter 1 gives a short introduction to the application of integer programming to staff scheduling problems. The purpose of this chapter is to provide a background to reading Papers [I] and [II], which themselves give a short introduction to the area of nurse scheduling.

Chapter 2 introduces the fundamentals of integer programming column generation. Papers [III], [IV], and [V] contribute to the area of column generation, and the purpose of Chapter 2 is to relate our contributions to other research conducted in the area.

Chapter 3 gives a background that facilitates the understanding of the methods presented in Papers [IV] and [V]. Both these papers suggest column generation methods for solving the set partitioning problem, and these methods utilise a certain property of the set partitioning problem. Chapter 3 describes this property and presents results from the literature in a new way in an attempt to make them easier to understand. Some additional insights originating from our work and that have not been included in the papers are also included in this chapter.

Chapter 4 summarises the contributions of this thesis.

1 Staff scheduling

The term staff scheduling refers to the process of determining when the members of a staff shall carry out their duties and what tasks they shall perform in order to meet the service demand that the employer wants to fulfil. A thorough introduction to the area of staff scheduling is found in Ernst et al. [11], and the essence of the brief overview to be presented here originates from that paper. See also Ernst et al. [10] for a bibliography of over 700 papers on staff scheduling.

The earliest operations research paper to address a staff scheduling issue was Edie [9] in 1954, and since then, several types staff scheduling applications have been presented in the literature. The following application areas are considered to be the most important ones:

- **Transportation systems** In the transportation market (airlines, railways etc.) staff scheduling is referred to as crew scheduling. Common to crew scheduling problems is that all tasks are determined from a timetable and are associated with a time and a place. Because of its economic importance, airline crew scheduling is probably the most well-studied application within staff scheduling.

- **Call centres** Typical for a call centre is that the service demand varies both during a day and from one day to another. In order to meet such demand in a cost efficient way, the start times and shift lengths need to vary. It is also often the case that both some under staffing and over coverage of the demand needs to be accepted.

- **Health care systems** The most common staff scheduling application within health care is nurse scheduling. The service demand in nurse scheduling is induced by the number of patients in the ward, and it is typical that these problems are often over-constrained due to all the regulations and other requirements on the schedules.

- **Protection and emergency services** The demand for police, ambulance, fire, and security services is often specified in terms of response times to attend incidents with properly trained officers. The acceptable patterns of shift work are often controlled by tight regulations.

- **Civic services and utilities** This application of staff scheduling includes, for example, government agencies dealing with claims for pensions, companies dealing with insurance claims, postal processing, and libraries. These types of services are often labour intensive and their challenges with respect to scheduling have been subject to some research.

- **Venue management** This type of scheduling includes for example ground operations at an airport, cargo terminals, casinos and sporting venues. Common to these applications is that a variety of skills are required to perform a task.

- **Financial services and retail** Common to staffing in financial services and in the retail industry is that the demand for services is variable over the day as is the case for call centres. There are not many papers published on this kind of scheduling.

- **Hospitality and tourism** The most common type of scheduling addressed within this area concerns hotel staff, and it is a type of scheduling that is similar to the one in financial services and retail. A difference is, however, that there are often more kinds of skills to take into account in hotel staff scheduling.

- **Manufacturing** The challenge of this type of scheduling is to have the right staffing levels in each production period in order to meet the production levels established. This type of scheduling can also be integrated with project task scheduling.

As is indicated by the above description, different staff scheduling applications can include different aspects that need to be taken into account. A consequence of these differences is that, in the current state of the art of staff scheduling, different mathematical models and algorithms are used for different applications in order to manage to provide solutions in an efficient way.

Since a staff scheduling problem can include the whole process from determining the demand levels for a complete planning period down to determining what tasks each member of the staff shall carry out on a day-to-day basis, it is usually not practically possible to deal with the whole problem at once. Ernst et al. [11] suggest a modularisation of a general staff scheduling problem to be used as a reference when decomposing the problem into manageable sub-problems. For each of the modules, different mathematical formulations might be required for different applications, and based on what formulations are used, different ways of decomposing the problem can be profitable. In short, the suggested modules are as follows.

- **Demand modelling** The aim of this module is to determine the number of staff needed at different times over a planning period. The demand can be *task based* (obtained from a list of tasks to be performed, as for example in transport applications), *flexible* (the demand is not well known and is instead predicted by some forecasting technique, as for example in call centres and for police services), or *shift based* (the number of staff required during each shift is specified in advance, as for example in nurse scheduling).

- **Days-off scheduling** For each line of work, this module is used to determine how rest days are to be interspersed between work days.

- **Shift scheduling** If there is a large pool of shifts from which some shifts shall be chosen to cover the demand, this module is used to determine the shifts to be chosen and the number of staff for each of them. This module is redundant when the demand is shift-based.

- **Line of work construction** This module is used to construct lines of work that are useful both with respect to the rules that prescribe what schedules are feasible and with respect to the service demand.
- **Task assignment** In some applications, each task is assigned to be carried out during a certain shift. The shift chosen for the task must be within a line of work that is assigned to a member of the staff that has the right skills for this task.
- **Staff assignment** The purpose of this module is to assign each member of the staff to a line of work.

In some applications, it is not possible to separate the problem into these modules, both for the reason that not all are of interest, and because some of them coincide.

Since staff scheduling problems are often highly constrained, it is difficult to find good solutions, and of course even more difficult to find optimal ones. The most commonly used approaches for solving staff scheduling problems are mathematical programming and meta-heuristic ones. The meta-heuristic approaches are known for being relatively robust and they are suitable for handling the messy objectives and constraints of the real world. A drawback of using a meta-heuristic is of course, that there is no guarantee for the quality of the solution. Pure mathematical programming formulations are limiting in the sense of what constraints and objectives can be expressed easily, and such formulations are more common when dealing with simplified versions of real world problems.

The airline crew scheduling problem has attracted a lot of attention within the research community, and a very popular approach for solving such problems is by column generation. When applying column generation, a lot of the complexity is hidden within the columns, making the subproblem a real challenge to solve and it is preferabe that this is done by applying some heuristic. For these types of scheduling problems, the column generation master problems are of set partitioning type, and an introduction to column generation techniques for such problems is found in the next chapter.

2 An introduction to integer programming column generation

This chapter gives an introduction to the fundamentals of integer programming column generation. The description has a strong bias towards set partitioning problems, partly because these are common, well studied, and a topic of this thesis, but also because such a limitation makes it possible to reduce the amount of technical details.

Even if column generation has provided many success stories in integer programming, it is originally a linear programming method, and therefore this introduction will use linear programming as its point of departure.

2.1 Linear programming column generation Assume that the following problem is to be solved by the simplex method for linear programs,

$$[LP-MP] \qquad \begin{aligned} z^* \;=\; & \min \quad \sum_{j\in\mathcal{N}} c_j\lambda_j \\ & \text{s.t.} \quad \sum_{j\in\mathcal{N}} a_j\lambda_j \geq b, \\ & \qquad \lambda_j \geq 0, \quad j\in\mathcal{N}, \end{aligned}$$

where $\mathcal{N}$ is the set of indices for the non-negative continuous variables λ_j, $j \in \mathcal{N}$. Let u denote the vector of dual variables associated with the linear constraints. When the pricing step of a simplex iteration is performed, a non-basic variable $\lambda_{j'}$ such that

$$j' = \arg\min\{\bar{c}_j = c_j - u^{\mathrm{T}}a_j :\ j\in\mathcal{N}\}$$

is chosen to enter the basis if $\bar{c}_{j'} < 0$, and if $\bar{c}_{j'} \geq 0$, an optimal solution has already been found and the algorithm is terminated.

In a column generation setting, the set $\mathcal{N}$ is assumed to be huge, making it practically challenging or even impossible to store all columns explicitly. Instead, only a subset of the columns $N \subseteq \mathcal{N}$ is assumed to be at hand, forming a restricted master problem $LP-RMP$, which throughout this chapter will be assumed to contain a feasible solution.

Let $\bar{\lambda}$ and $\bar{u}$ respectively be an optimal primal solution and a corresponding complementary dual solution to $LP-RMP$. In the pricing step, instead of an explicit search among the variables $j \in \mathcal{N}\setminus N$, a subproblem is solved over a set P, which implicitly describes the feasible columns by using constraints. The cost of the column is assumed to be calculated as $c_j(a_j)$, $j\in\mathcal{N}$. The objective of the subproblem is to find

$$[LP-SP] \qquad \bar{c}^* = \min\left\{c(a) - u^{\mathrm{T}}a :\ a\in P\right\}.$$

Typical for many applications of column generation is that a column represents an object that is easy to interpret in the application at hand, for example a path in a network or an assignment of jobs to a machine.

The problem formulation using $LP-MP$ and $LP-SP$ introduced above, is called an extensive formulation of a problem. This extensive formulation can either be the original formulation, or it can be the result of applying Dantzig-Wolfe decomposition, see Dantzig and Wolfe [8], on a corresponding compact formulation of the problem.

Given a compact formulation of a linear program of the form

$$[LP-COMP] \qquad \begin{array}{rll} z^* = & \min & c^{\mathrm{T}}x \\ & \text{s.t.} & Ax \geq b, \\ & & Dx \geq d, \\ & & x \geq 0, \end{array}$$

Dantzig-Wolfe decomposition can be performed as follows. Let

$$P = \{x:\ Dx \geq d,\ x \geq 0\} \neq \emptyset.$$

Assuming that P is bounded, each $x \in P$ can be written as a convex combination of extreme points $\{p_j\}_{j\in\mathcal{N}}$, where the set $\mathcal{N}$ is finite. To obtain the extensive formulation of $LP-COMP$, x is replaced by

$$x = \sum_{j\in\mathcal{N}} p_j\lambda_j, \quad \sum_{j\in\mathcal{N}} \lambda_j = 1,\ \lambda_j \geq 0, j \in \mathcal{N}.$$

To mimic the notation used in $LP-MP$, form $c_j = c^{\mathrm{T}}p_j$ and $a_j = Ap_j$, $j \in \mathcal{N}$; the resulting extensive formulation then becomes

$$[LP-MP_{D-W}] \qquad \begin{array}{rll} z^* = & \min & \displaystyle\sum_{j\in\mathcal{N}} c_j\lambda_j \\ & \text{s.t.} & \displaystyle\sum_{j\in\mathcal{N}} a_j\lambda_j \geq b, \\ & & \displaystyle\sum_{j\in\mathcal{N}} \lambda_j = 1, \\ & & \lambda_j \geq 0, \quad j \in \mathcal{N}, \end{array}$$

and

$$[LP-SP_{D-W}] \qquad \min\left\{(c^{\mathrm{T}} - u^{\mathrm{T}}A)x:\ Dx \geq d,\ x \geq 0\right\}.$$

The extensive and the compact formulation of a problem are equivalent in the sense that they have the same optimal objective value, but the respective polyhedrons of $LP-MP_{D-W}$ and $LP-COMP$ are not combinatorial equivalent, see comments in [18]. For a description of the more general case when P is not bounded, see [18].

2.2 Column generation for set partitioning problems A natural and the most common way to obtain integer solutions in a column generation setting, is to apply the branch-and-price strategy, which is to embed linear programming column generation in a linear-programming-based branch-and-bound scheme. This might at a first glance seem very straightforward, but that is however not the case, firstly since the columns generated during the solution of the linear programming relaxation are typically not sufficient to solve the integer program, and secondly, since the branching strategies normally used in branch-and-bound are likely to perform very poorly.

This section introduces the basic ideas used in integer programming column generation by describing them for the case when the master problem is a set partitioning problem. The restriction to set partitioning problems makes it easier to convey the general ideas without having to focus too much on technical details.

2.2.1 Model formulation The modelling framework presented with a compact and an extensive formulation respectively can also be used in integer programming column generation, for details see [18]. An extensive formulation can be obtained from a compact formulation by performing Dantzig-Wolfe decomposition, and vice versa, if only the extensive formulation is known Villeneuve et al. [28] have shown how to constructively obtain a corresponding compact formulation, if some very mild assumptions hold.

For the set partitioning problem the following formulations will be used. In the compact model, introduce the decision variables

$$x_{ik} = \begin{cases} 1 \text{ if element } i \text{ is assigned to subset } k, \\ 0 \text{ otherwise} \end{cases} \quad i \in M,\ k \in K.$$

Let S_k be the finite set of feasible assignments of elements for subset k, $k \in K$, also, let c_{ik} be the cost of assigning element i to subset k, $i \in M$, $k \in K$. The compact model can then be formulated as,

$$\begin{aligned} [COMP] \qquad z^* \ = \quad \min \quad & \sum_{k\in K}\sum_{i\in M} c_{ik}x_{ik} \\ \text{s.t.} \quad & \sum_{k\in K} x_{ik} = 1, \quad i \in M, \\ & (x_{ik})_{i\in M} \in S_k \subseteq \{0,1\}^{|M|},\ k \in K. \end{aligned}$$

Introduce $\left(y_j^k\right)_{j\in\mathcal{N}_k}$ as the feasible points of S_k, $k \in K$, and for $i \in M$, let $y_{ij}^k = 1$ for the elements i assigned to subset k in y_j^k and let $y_{ij}^k = 0$ otherwise. Let c_j^k be the cost of column $j \in \mathcal{N}_k$ for subproblem $k \in K$ and denote the master problem variables by

$$\lambda_j^k = \begin{cases} 1 \text{ if assignment } j \text{ of subset } k \text{ is used}, \\ 0 \text{ otherwise} \end{cases} \quad j \in \mathcal{N}_k,\ k \in K,$$

the master problem then becomes

$$\begin{aligned} [MP] \qquad z^* \ = \quad \min \quad & \sum_{k\in K}\sum_{j\in\mathcal{N}_k} c_j^k\lambda_j^k \\ \text{s.t.} \quad & \sum_{k\in K}\sum_{j\in\mathcal{N}_k} y_{ij}^k\lambda_j^k = 1, \quad i \in M, \\ & \sum_{j\in\mathcal{N}_k} \lambda_j^k = 1, \quad k \in K, \\ & \lambda_j^k \in \{0,1\}, \quad j \in \mathcal{N}_k,\ k \in K. \end{aligned}$$

Denote the linear programming relaxation of MP by MP_{LP} when the binary requirement on λ_j^k is replaced by a non-negativity requirement. Let u_i, $i \in M$ be the complementary dual variables for the partitioning constraint of MP_{LP}. For each subset $k \in K$, the subproblem of the extensive formulation becomes

$$[SP_k] \qquad \min\left\{\sum_{i\in M}(c_{ik} - u_i)y_i^k : \ \left(y_i^k\right)_{i\in M} \in S_k\right\}.$$

When discussing branch-and-price in the next section, it is of special interest to study the case when all subsets have identical requirements, since such problems suffer from inherent symmetries that cause branch-and-bound approaches on the compact formulation to perform poorly.

When the requirements on the subset are identical, introduce $S = S_1 = \cdots = S_k$, and for $j \in \mathcal{N}$, let y_{ij}, $i \in M$ and c_j denote the elements of the feasible points of S and their respective costs. The extensive model can be obtained by aggregating the master problem variables as $\lambda_j = \sum_{k \in K} \lambda_j^k$, $j \in \mathcal{N} = \mathcal{N}_1 = \cdots = \mathcal{N}_K$, which in turn makes it possible to combine the convexity constraints into one constraint, resulting in the following master problem formulation.

$$
\begin{aligned}
[MP^{ID}] \qquad z^* \; = \quad & \min \quad && \sum_{j \in \mathcal{N}} c_j \lambda_j \\
& \text{s.t.} && \sum_{j \in \mathcal{N}} y_{ij} \lambda_j = 1, \quad i \in M, \\
& && \sum_{j \in \mathcal{N}} \lambda_j = K, \\
& && \lambda_j \in \{0,1\}, \quad j \in \mathcal{N}.
\end{aligned}
$$

The common subproblem becomes

$$
[SP^{ID}] \qquad \min \left\{ \sum_{i \in M} (c_i - u_i) y_i : \; (y_i)_{i \in M} \in S \right\},
$$

where $c_i = c_{i1} = \cdots = c_{iK}$.

The models introduced above will be used when the concept of branch-and-price is introduced in the next section.

2.2.2 Branch-and-price The core mechanism of branch-and-price is to apply linear-programming-based branch-and-bound on MP of the extensive formulation, using SP_k, $k \in K$ for generating new columns to MP. Since the columns generated by solving the linear programming relaxation of MP at the root node of the tree are typically not those needed in a solution to the integer program, it is likely that more columns need to be generated to solve the problems in the nodes throughout the tree.

To apply a standard strategy for branching in branch-and-bound would be to branch on fractional valued λ_j^k, $j \in \mathcal{N}_k$, $k \in K$, by creating one branch $\lambda_j^k = 1$ and one branch $\lambda_j^k = 0$. The 1-branch forces one column to be used in the solution, and this causes no immediate trouble, but the 0-branch does, because it forces one particular column not to be used in a solution to MP. A column is a solution to a subproblem, and the 0-branch therefore means that this particular solution is omitted from the subproblem. It is, however, quite likely that the optimal solution to the subproblem is this column, and if this is the case, the second best solution should be used instead. At level l in the tree, there might occur a situation where all the $l - 1$ best solutions should be omitted. The practical drawbacks of using

this branching strategy make it likely that the over-all performance of the solution strategy will be unsatisfactory.

An alternative to branching on the MP-variables is to instead branch on the compact variables x_{ik}, $i \in M$, $k \in K$. The $x_{ik} = 1$-branch is obtained by forcing element i to be assigned to subset k in subproblem SP_k by letting $y_i^k = 1$, and by excluding all columns in MP where this does not hold. In the $x_{ik} = 0$-branch element i is permanently excluded from subset k in subproblem SP_k by letting $y_i^k = 0$, and all columns in MP where this does not hold are excluded from MP.

The strategy of branching on the compact variables is viable for a problem setting with distinct subproblems, but when there are identical subproblems, as is the case for the formulations MP^{ID} and SP^{ID} above, this strategy will not perform well. The inefficiency is caused by symmetries, due to the fact that there is an exponential number of solutions that differ only by the names of the subsets, hence, by swapping the assignment of elements between two subsets, the solutions are the same, but the values of the compact variables differ. If one solution is excluded, it can reappear several times elsewhere in the tree, creating several alternative optima and causing weak bounds.

A remedy to this difficulty is to apply the well known branching strategy from Ryan and Foster [20], which is here applied to MP^{ID}. Introduce $J(r,s) = \{j \in \mathcal{N} : y_{rj} = y_{sj} = 1\}$ to denote the set of indices for the columns covering both rows r and s, where $r,\ s \in M$. Choose (r,s) such that

$$0 < \sum_{j \in J(r,s)} \lambda_j < 1,$$

(see [6] for a proof that such pair of rows always exists for a fractional solution to MP) and create the branches

$$\text{(i)} \quad \sum_{j \in J(r,s)} \lambda_j = 1, \qquad \text{and} \qquad \text{(ii)} \quad \sum_{j \in J(r,s)} \lambda_j = 0.$$

In branch (i), rows r and s shall always be covered by the same column, which can be achieved by combining the two rows into one and exclude all columns of MP^{ID} for which this is not possible. Branch (ii) can be obtained by restricting the columns to be such that they cannot cover both rows r and s and eliminating from MP^{ID} all columns which cover both rows.

The branching strategy from Ryan and Foster [20] can also be applied to MP and SP_k, $k \in K$, where the subsets are not identical. If in such a case r is chosen among the rows of the partitioning constraints, and s is chosen among the convexity constraints, this branching strategy is equivalent to branching on the original variables since it (i) forces element r to be assigned to subset s, and (ii) prohibits this assignment.

This introduction to integer programming column generation does not cover branching strategies for general mixed integer programming problems, and the reader is referred to the following papers. In the case of subsets with different restrictions,

branching can be performed on the variables of the compact formulation, see description in [6] which refers to Johnson [17]. General branching strategies that avoid difficulties related to symmetries were independently suggested by Vanderbeck and Wolsey [25] and Barnhart et al. [5], but see also [26], [27], [18], [6], and [29].

2.2.3 Implementation issues The introduction above merely provides the basic ideas of how column generation can be applied to a set partitioning problem. From an implementation point of view, there are many additional aspects to take into account, and apart for some general remarks that will now follow, implementation issues will not be covered in this introduction.

Since the dual solution has an important influence on which column will be generated when a subproblem is solved, it is of great importance that the sequence of dual solutions generated during the course of the algorithm will give rise to useful columns. To achieve this however, is not a straightforward task, due to the following circumstances.

Because of primal degeneracy, which is usually massive for set partitioning problems, a primal solution's complementary dual solution does not need to be unique; to the contrary, it usually constitutes a face of the dual feasible polyhedron. The extreme points of this dual face can give rise to subproblem solutions which differ a lot, and because of this, which of the dual solutions of the face that is used matters, and whether it is an extreme point or an interior point. Means for controlling the choices of dual solutions is an important area within column generation research.

Another common inherent property of the master problem is that the columns useful for solving the linear programming relaxation are of little or no use in an optimal integer solution, and vice versa, having the columns of an optimal integer solution at hand when solving the linear programming relaxation might be inefficient.

Linear programming column generation suffers from the so-called tailing-off effect; meaning that a near optimal solution can be obtained quite fast, but that it then takes a comparatively long time to obtain and verify optimality. When solving the linear programming relaxations of the restricted master problem throughout the branch-and-price tree, the tailing-off effect can be avoided by settling for near-optimal solutions.

Since the means for obtaining integer solutions is to apply a branching scheme, it is of great importance how the branching strategies are designed and implemented. A lot of research has been conducted to derive efficient branching schemes, which commonly relies on branching on the variables of the compact formulation.

2.2.4 Alternative strategies for obtaining integrality There is very little literature on alternatives to branch-and-price for obtaining integer solutions in a column generation setting. An introduction to the only such approach that we are familiar with is found in Paper [III]. For the original references, see Baldacci et al. [3], Baldacci et al. [4], and Sweeney and Murphy [22].

2.3 Further reading and historical perspective The essence of this introduction to integer programming column generation is taken from Lübbecke and Desrosiers [18] and Barnhart et al. [6], if no other reference is given. Together, these papers provide an excellent survey of the area and I recommend them as a starting

point for further reading. A third survey containing several detailed examples has been written by Wilhelm [29].

The column generation strategy originates from a paper by Ford and Fulkerson [12] in 1958, who suggested that the variables of a multicommodity flow problem should be dealt with only implicitly. This idea was further developed in Dantzig and Wolfe [8], 1960, where the Dantzig-Wolfe decomposition principle was introduced. The first implementation of a column generation strategy is found in Gilmore and Gomory [13] and Gilmore and Gomory [14], from 1961 and 1963 respectively. The challenges of combining column generation and linear-programming-based branch-and-bound were first described in Appelgren [1] in year 1969. Together, these pioneering papers set the scene for the development of efficient column generation strategies for solving integer programming problems.

3 On the set partitioning problem and quasi-integrality

The set partitioning problem as well as its close relatives, the set covering and the set packing problem, arise in a wide variety of situations, which all aim to distribute some resource in order to fulfil some demand or without causing any conflicts. The structures of these types of problems can be illustrated by the following example.

Example 1 *Consider six boxes, each assigned a value and some letters according to Figure 3.1.*

a. *An example of a* set partitioning problem *is to choose a combination of boxes with the lowest total value and such that the letters included are* **exactly** *the ones needed for forming the word* OPTIMISATION. *In such formulation, boxes that include a letter which is not part of the word* OPTIMISATION *shall be discarded. The two feasible solutions to this problem are* $\{I, V\}$*, with profit 8, and* $\{III, VI\}$*, with profit 7, making the latter solution the optimal one.*

b. *An example of a* set covering problem *is to choose a combination of boxes with the lowest total value and such that the letters included are* **at least** *the ones needed for forming the word* OPERATIONS RESEARCH. *The two feasible solutions to this problem are* $\{I,\ II,\ IV,\ V\}$*, with profit 13, and* $\{II,\ ,III,\ IV,\ VI\}$*, with profit 12, making the latter solution the optimal one.*

c. *An example of a* set packing problem *is to choose a combination of boxes with the highest total value and such that the letters included appear* **at most** *once. The boxes I, II, III, and V can never be part of a solution since they individually contain some letter twice. The boxes left are IV and VI, which cannot be picked together, and hence the optimal solution is to pick box IV.*

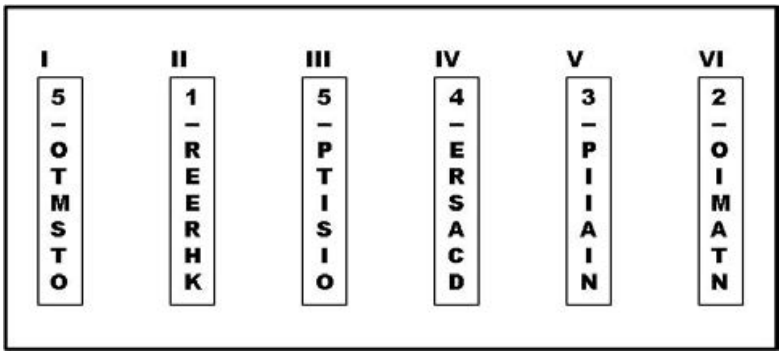


Figure 3.1: Six boxes of letters and each box is assigned a profit.

The more practically useful examples to follow are from Hoffman and Padberg [16].

One of the most well-known set partitioning formulations originates from solving cockpit crew scheduling problems, where each flight leg of an airline must be assigned one crew. Another example is the political districting problem where a geographical region should be divided into districts such that each citizen is assigned to one district. This type of problem structure is also common in scheduling, routing, delivery, and location applications, but the distribution of the resource does then not necessarily need to be a partitioning. It can either suffice that some demand is covered, and then the set covering formulation can be used, or it can suffice that that the distribution is made without conflicts, and then the set packing formulation can be used.

Thanks to the importance of the set partitioning problem, the properties of its set of feasible solutions have been subject to extensive studies in order to obtain strong formulations; for references to such studies, see Hoffman and Padberg [16]. This chapter is devoted to one particular property of the polytope of feasible solutions to the linear programming relaxation of the set partitioning problem — *the quasi-integrality property* — and strategies for solving the set partitioning problem facilitated by this property.

The quasi-integrality property implies that there exists a path between every pair of feasible integer points, consisting only of such edges of the polytope that connects feasible integer points by simplex pivots on one-entries. The existence of this path enables the use of linear programming techniques for finding improved integer solutions. This possibility was first explored by Trubin [24] in the sixties, and was then more thoroughly described and named *the integral simplex method* by Yemelichev et al. [31]. The integral simplex method only acts as a local search method, but complemented with a branching strategy, as in Thompson [23], the finding of an optimal solution can be ensured.

This chapter contains both results from the literature, which are presented in a new way with the purpose of making them easier to understand, and some new insights into how the quasi-integrality property can be used when solving the set partitioning problem. These new insights lead to the definition of *all-integer pivots* which can be described as a well motivated version of the integral simplex method.

3.1 The set partitioning problem Throughout this chapter, the following notation will be used to describe the set partitioning problem.

$$
\begin{aligned}
[SPP] \qquad z^* \;=\; & \min \quad \sum_{j\in N} c_j x_j \\
& \text{s.t.} \quad \sum_{j\in N} a_{ij} x_j = e_i, \quad i \in M \\
& \qquad\quad x_j \in \{0,1\}, \quad j \in N.
\end{aligned}
$$

Here, $N = \{1,\dots,n\}$ and $M = \{1,\dots,m\}$ are the sets of indices for the variables and constraints respectively. The costs, c_j, $j \in N$, are integers, a_{ij}, $i \in M$, $j \in N$, are zeros and ones, and $e_i = 1$, $i \in M$. We assume that $n \geq m$ and that the problem is feasible. Let $a_j = (a_{1j},\dots,a_{mj})^{\mathrm{T}}$, $j \in N$, and assume, without loss of generality, that the matrix $(a_1,\dots,a_n)$ has no zero column and full rank.

Let SPP^{LP} be the linear programming relaxation of SPP when $x_j \in \{0,1\}$ is replaced by $x_j \geq 0$, $j \in N$. Because the feasible set of SPP^{LP} is contained within the unit hypercube, all feasible integer points will be extreme points of this set. For an extreme point to SPP^{LP}, denote a basis by $B \subseteq \{a_1, \ldots, a_n\}$. If an extreme point is degenerate, it is associated with more than one basis. For a given basis, let I and J denote the corresponding index sets of basic and non-basic columns, respectively, and let $u^{\mathrm{T}} = c_B^{\mathrm{T}} B^{-1}$, where $c_B = (c_j)_{j \in I}$, be the complementary dual basic solution. For this basis, let $\bar{e} = B^{-1}e$, where $e = (e_1, \ldots, e_m)^{\mathrm{T}}$, and $\bar{a}_j = B^{-1}a_j$, $j \in N$, be the updated right-hand-side and constraint columns, respectively, and $\bar{c}_j = c_j - u^{\mathrm{T}} a_j$, $j \in J$, be the reduced costs.

Two bases are called *adjacent* if they differ in exactly one column, and two extreme points are called adjacent if there exists a basis belonging to one of the extreme points that is adjacent to a basis belonging to the other extreme point.

3.2 The quasi-integrality property This section formally introduces the quasi-integrality property and presents a proof that the set partitioning problem has this property.

Definition 1 *Let X be a polytope and X_I its set of integer points. The polytope X is called* quasi-integral *if every edge of the convex hull of X_I is also an edge of X.*

The first to show that the polytope of feasible solutions to SPP^{LP} has this property was Trubin [24]. Given below is what in our opinion is the more accessible proof given by Yemelichev et al. [31], though with some minor rephrasings.

Theorem 1 *The feasible polytope of SPP^{LP} is quasi-integral.*

The main argument used in the proof is that a sufficient condition for the quasi-integrality property to hold, is that each pair of integer extreme points belongs to an integral face of the polytope describing the set of feasible solutions to the linear programming relaxation of the integer problem. The term integral face refers to a face that has the integrality property, which holds if the face can be described by constraints of which the matrix is totally unimodular.

For the proof we need the following two lemmas; the first is the mentioned sufficient condition for a polytope to be quasi-integral and it is taken from [31].

Lemma 1 *If, given any two integral vertices of X there is an integral face containing them, then X is quasi-integral.*

The second lemma gives a sufficient condition for a matrix to be totally unimodular, and it is taken from Wolsey [30], page 39.

Lemma 2 *A matrix A with elements a_{ij} is totally unimodular if*

(i) $a_{ij} \in \{0, 1, -1\}, \quad \forall i, j.$

(ii) Each column contains at most two nonzero coefficients.

(iii) There exists a partition (M_1, M_2) of the set of rows such that each column j containing two nonzero coefficients satisfies

$$\sum_{i \in M_1} a_{ij} - \sum_{i \in M_2} a_{ij} = 0.$$

The results needed to prove the results of Theorem 1 are now available, and the proof is as follows.

Proof of Theorem 1: According to Lemma 1, and because all feasible integer points of SPP are extreme points of the feasible polytope of SPP^{LP}, it suffices to show that any two feasible integer points, x^1 and x^2, of SPP belong to an integral face of the polytope of SPP^{LP}.

Partition the index set N into the three disjoint subsets

$$N_0 = \{j : x_j^1 = x_j^2 = 0\},\ N_1 = \{j : x_j^1 = x_j^2 = 1\},\ \text{and}\ N_2 = \{j : x_j^1 \neq x_j^2\}.$$

The set of feasible solutions of SPP^{LP} that satisfy $x_j = 0$, $j \in N_0$, and $x_j = 1$, $j \in N_1$, constitutes a face of the polytope of feasible solutions to SPP^{LP}. To show that this face is integral, it suffices to show that the matrix $A_2 = (a_j)_{j \in N_2}$ is totally unimodular. Since $x^0 = (x^1 + x^2)/2$ is a feasible solution to SPP^{LP}, it holds that

$$2 = \sum_{j \in N} a_{ij}(x^1 + x^2) \geq \sum_{j \in N_2} a_{ij}.$$

Hence, every row of A_2 contains at most two elements equal to one. The columns of A_2 can be partitioned into two disjoint sets (M_1, M_2), where the first contains the variables x_j such that $x_j^1 = 1$ and the second contains the variables x_j such that $x_j^2 = 1$. That is, the transpose of A_2 fulfils the conditions of Lemma 2 and is thereby totally unimodular. A matrix is totally unimodular if and only if its transpose is totally unimodular, see [30, page 39], so A_2 is totally unimodular and the face containing x^1 and x^2 is integral. ■

The following property of the set partitioning problem, which is closely related to the quasi-integrality property, was shown in Balas and Padberg [2].

Theorem 2 *Let x^1 be a feasible and integer solution to SPP^{LP}, associated with the basis B_1, and suppose that x^1 is not optimal in SPP. Denote by x^2 an optimal solution to SPP, and let B_2 be an associated basis in SPP^{LP}. Let J_1 and Q_2 be the index sets of the non-basic columns in x^1 and the one-valued basic variables in x^2, respectively. Then there exists a sequence of adjacent bases $B_{10}, B_{11}, B_{12}, \ldots, B_{1p}$ in SPP^{LP}, such that $B_{10} = B_1$, $B_{1p} = B_2$, and (a) the associated vertices $x^1 = x^{10}, x^{11}, x^{12}, \ldots, x^{1p} = x^2$, are all feasible and integer, (b) $\sum_{j \in N} c_j x_j^{10} \geq \sum_{j \in N} c_j x_j^{11} \geq \ldots \geq \sum_{j \in N} c_j x_j^{1p}$, and (c) $p = |J_1 \cap Q_2|$.*

The sequence of bases given in the theorem can be obtained by making simplex pivots, if the degenerate pivots are also allowed to be made on any non-zero entry. The construction of this sequence does however require knowing the optimal solution x^2.

Since any vertex can be made optimal by adjusting the objective function, it follows from the theorem that for *any* two integer feasible basic solutions to SPP^{LP}, x^1 and x^2, there exists a sequence of adjacent bases of length $p = |J_1 \cap Q_2|$, such that the associated vertices are all feasible and integer. Note that a consequence of Theorem 2 is that the Hirsch conjecture, see [19, page 441], is true for integer solutions to SPP^{LP}.

Examples of other problems that have the quasi-integrality property are *the simple plant location problem*, see [31], *the uncapacitated network design problem*, see [15], and *the one and two facility, one commodity, network design problem*, see [21].

3.3 The integral simplex method As long ago as in his paper from the sixties, Trubin [24] outlined the idea of a method, the integral simplex method, that utilises the fact that the polytope of SPP^{LP} has the quasi-integrality property. The same idea was later also described in Yemelichev et al. [31], but neither of them specify an actual algorithm for how to implement the integral simplex method. The only implementation of the integral simplex method that we are familiar with is by Thompson [23]. In his realisation, the method is started from a purely artificial basis, and then simplex pivots on one-entries are performed. These pivots are referred to as *pivots-on-one*, and because of their importance, we make the following definition.

Definition 2 *Given a basis associated with a feasible solution to* SPP^{LP} *and a* $s \in J$ *such that* $\bar{c}_s < 0$*, then*

(i) a non-degenerate pivot-on-one *is a pivot operation on an entry* $\bar{a}_{rs} = 1$ *such that*

$$\min_{i \in M} \left\{ \frac{\bar{e}_i}{\bar{a}_{is}} \;\middle|\; \bar{a}_{is} > 0 \right\} = \frac{\bar{e}_r}{\bar{a}_{rs}} = 1, \textit{and}$$

(ii) a degenerate pivot-on-one *is a pivot operation on an entry* $\bar{a}_{rs} = 1$ *such that*

$$\min_{i \in M} \left\{ \frac{\bar{e}_i}{\bar{a}_{is}} \;\middle|\; \bar{a}_{is} > 0 \right\} = \frac{\bar{e}_r}{\bar{a}_{rs}} = 0.$$

Both (i) and (ii) are referred to as a pivot-on-one.

An advantage of the integral simplex method is that all the solutions encountered are integral, but there are also some major challenges and drawbacks associated with the method. Firstly, due to the fact that set partitioning problems are often highly degenerate, it can be difficult to find a basis that enables a non-degenerate pivot, and there is a risk of a lot of time being spent on making degenerate pivots which will yield no improvement of the objective function.

Secondly, and also because of degeneracy, the prevention of cycling is an important issue. If the integral simplex method is to be used for solving problems of practical interest, a rule for efficient prevention of cycling needs to be constructed, and to the best of our knowledge, no pivoting rule that is guaranteed to prevent cycling has been published.

A third drawback of the integral simplex method, that follows from the restriction to pivots-on-one, is that one might reach a basis associated with a non-optimal extreme point where no further pivot-on-one is possible. For this reason, the integral simplex method can be regarded as a local search method that can get trapped at a local optimum, as defined below.

Definition 3 *A basis associated with an integer extreme point to* SPP^{LP} *is called a* local optimum *if it is not possible to make a pivot-on-one.*

By complementing the integral simplex method with implicit enumeration, optimality can eventually be obtained, and such a strategy has been suggested by Thompson [23]. In his method, the feasible set is partitioned through a branching, when the method is trapped at a local optimum. In his branching strategy, a subproblem is created for each variable with a negative reduced cost, and in the subproblem, this variable is forced to take the value one. The fixation of a variable enables a reduction of the original problem, and each of the reduced problems is solved by the integral simplex method, with further branchings performed if necessary. Thompson has applied his method to randomly generated problem instances and problems arising in crew scheduling, and he reports that the results are promising.

3.4 All-integer pivots The development of the integral simplex method has its point of departure in how to restrict the simplex method to maintain integrality. When studying the old results presented above, we raised some other questions: 'What pivots is it possible to make if integrality should be guaranteed, and how can we understand the mechanisms of such pivots?'

The purpose of this section is to convey the understanding we gained during this work, and the conclusions we drew that helped extend the integral simplex method to all-integer pivots. We also provide arguments for why one should be careful if the set of possible pivots is extended beyond the all-integer pivots.

To make the description more easily accessible, the following example will be used throughout this section.

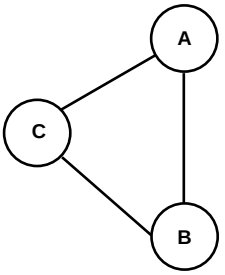
$$
\begin{array}{lrll}
[SPP_{EX}] & z^* = & \min & \begin{pmatrix} c_1 & c_2 & c_3 & c_4 & c_5 \end{pmatrix} x \\
 & & \text{s.t.} & \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix} x = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \\
 & & & x \in \{0,1\}^5.
\end{array}
$$

Because the example will be used to illustrate the quasi-integrality property which is a property of the feasible polytope of SPP^{LP}, there is no need to specify an objective function.

There are three extreme points associated with SPP^{LP}_{EX}, and these are $x^A = (1,0,0,0,0)$, $x^B = (0,1,0,0,1)$, and $x^C = (0,1/2,1/2,1/2,0)$. In Figure 3.2 the polytope describing the feasible set of SPP^{LP}_{EX} is illustrated by a graph, where the extreme points are represented by vertices, and the edges of the polytope are represented by edges in the graph.

Because of degeneracy, an extreme point can be associated with more than one basis. For x^A, there are five possible sets of basic variables, namely $\{x_1, x_2, x_3\}$, $\{x_1, x_2, x_4\}$, $\{x_1, x_3, x_4\}$, $\{x_1, x_3, x_5\}$, and $\{x_1, x_4, x_5\}$. For x^B there are two possible sets of basic variables, $\{x_2, x_3, x_5\}$ and $\{x_2, x_4, x_5\}$, and for x^C there is only one, $\{x_2, x_3, x_4\}$. In Figure 3.3, all these possible bases for each extreme point are illustrated as vertices within the super-vertex representing the extreme point. As in Figure 3.2, each edge of the graph represents the fact that there exists an edge of the polytope between the two extreme points.

So far, quasi-integrality has been described as a geometric property of the polytope of feasible solutions of SPP^{LP}, but there is also a corresponding algebraic property that can be described within the context of the simplex method. The simplex method for linear programs is a search method in which pivots between adjacent bases of the polytope of feasible solutions are performed. When carrying out a simplex pivot, a variable x_s, $s \in J$, such that $\bar{c}_s < 0$, is chosen to enter the basis, and a variable x_r,



$$\begin{aligned} x^A &= (1,0,0,0,0) \\ x^B &= (0,1,0,0,1) \\ x^C &= (0,1/2,1/2,1/2,0) \end{aligned}$$

Figure 3.2: A graph where the vertices and edges are representing the extreme points of SPP^{LP}_{EX} and the polytope edges that connects them, respectively.

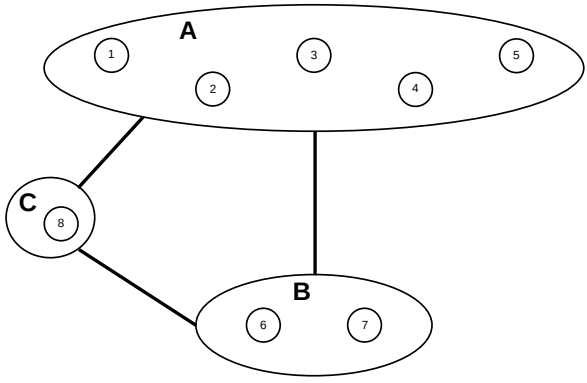


Figure 3.3: A graph where the vertices within a super-vertex represent the possible bases for each extreme point. Each edge shows that there exists an edge of the polytope between the two extreme points.

$r \in I$, such that

$$\min_{i \in M} \left\{ \frac{\bar{e}_i}{\bar{a}_{is}} \;\middle|\; \bar{a}_{is} > 0 \right\} = \frac{\bar{e}_r}{\bar{a}_{rs}}$$

is chosen to leave the basis. In Figure 3.4, the feasible bases of SPP_{EX}^{LP} are represented by vertices the same way as in Figure 3.3, and all possible simplex pivots between them are represented by edges in the graph. Within the theory of degeneracy graphs, the graph in Figure 3.4 is called a representation graph. For further reading on degeneracy graphs, see Zörnig [32].

3.4.1 Sufficient conditions for integrality When solving the set partitioning problem, we are only interested in moving between bases representing integer extreme points, and derived below are sufficient conditions for integrality.

When working with integer solutions to a linear program, the determinant of the basis is of special interest, and for future reference the following definition is given.

Definition 4 *A linear programming basis B is called* unimodular *if $|det(B)| = 1$.*

The importance of unimodular bases is motivated as follows. For $j \in I$, let x_{j_i} denote the ith basic variable. The values of x_{j_i}, $i \in M$, can be calculated by using Cramer's rule, that is,

$$x_{j_i} = \frac{det(B^i)}{det(B)}, \qquad i \in M,$$

where B^i is the matrix obtained when replacing the ith-column vector of B by the column vector e. The value of $det(B^i)$ is integral because the matrix B^i is integral, hence, if $|det(B)| = 1$, then all x_{j_i}, $i \in M$, are integral and the following lemma holds.

Lemma 3 *If the basis B is unimodular, then the corresponding extreme point is integral.*

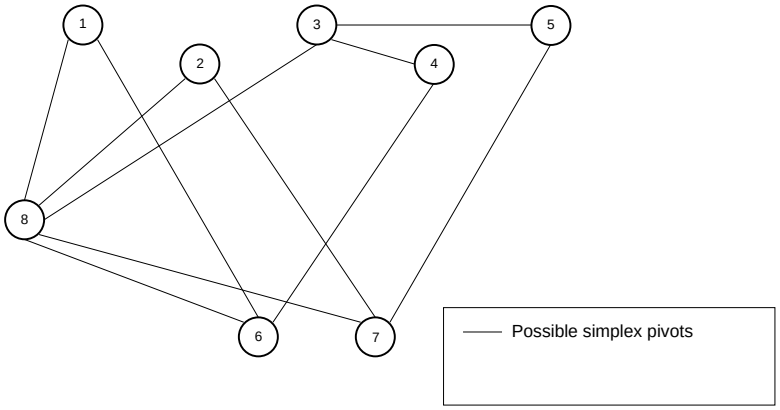


Figure 3.4: The possible simplex pivots between the bases of SPP_{EX}^{LP}

In the simplex method, the pivot operation that transforms the current basis B into the new basis B_{new} by letting the variable x_s enter the basis and the variable x_r leave the basis, can be represented by a pivot matrix, P, see Murty [19], page 141. The pivot matrix P is a unit matrix of order m with column r replaced by the eta vector,

$$\eta = \left(\frac{-\bar{a}_{1s}}{\bar{a}_{rs}}, \ldots, \frac{-\bar{a}_{r-1,s}}{\bar{a}_{rs}}, \frac{1}{\bar{a}_{rs}}, \frac{-\bar{a}_{r+1,s}}{\bar{a}_{rs}}, \ldots, \frac{-\bar{a}_{ms}}{\bar{a}_{rs}} \right)^T.$$

The relation between the current basis and the new one is that $B_{new}^{-1} = PB^{-1}$.

If we start from a unimodular basis, integrality can be maintained by allowing only such pivots that lead to a new unimodular basis. The relation between the determinants of the current basis and the new one is that

$$det(B_{new}) = \frac{det(B)}{det(P)}$$

where the determinant of P can be calculated by using expansion by minors on the rth row, yielding

$$det(P) = \frac{(-1)^{r+s}}{\bar{a}_{rs}}.$$

Then, if $|det(B)| = 1$, it holds that

$$|det(B_{new})| = \left| \frac{1}{det(P)} \right| |det(B)| = \left| \frac{1}{\bar{a}_{rs}} \right|,$$

which leads to the following lemma.

Lemma 4 *If a pivot is performed on an entry $\bar{a}_{rs} = \pm 1$ from a basis B such that $|det(B)| = 1$, then the pivot yields a new basis B_{new} such that $|det(B_{new})| = 1$.*

For a pivot to be between two distinct extreme points, it needs to be non-degenerate, and if the current solution is feasible, a non-degenerate pivot needs to be made on a positive entry $\bar{a}_{rs}$ in order for the new solution to be feasible. This is not required in the degenerate case, however, since the pivot is made between bases of the same solution. This observation leads to the our definition of all-integer pivots, and to a theorem providing a sufficient condition for a sequence of pivots to be between integer extreme points only.

Definition 5 *Given a unimodular basis associated with a feasible solution to SPP^{LP} and a $s \in J$ such that $\bar{c}_s < 0$. An* all-integer pivot *is,*

(i) in the non-degenerate *case a pivot made on an entry $\bar{a}_{rs} = 1$ such that*

$$\min_{i \in M} \left\{ \frac{\bar{e}_i}{\bar{a}_{is}} \;\middle|\; \bar{a}_{is} > 0 \right\} = \frac{\bar{e}_r}{\bar{a}_{rs}} = 1;$$

(ii) in the degenerate *case a pivot made on an entry $\bar{a}_{rs} = \pm 1$ such that $\bar{e}_r = 0$.*

Theorem 3 *Starting from a feasible unimodular basis for SPP^{LP}, a sufficient condition for a sequence of pivots to yield a path of integer extreme points is that the pivots are all-integer.*

Proof: From Lemma 3 it follows that it suffices to show that the all-integer pivots preserve feasibility in SPP^{LP} and unimodularity of the bases. That each basis in the pivoting sequence will be unimodular follows from Lemma 4. Since the choice of pivot entry in the non-degenerate case complies with the regular pivoting rule used in the simplex method, the pivot leads to a feasible solution. In the degenerate case, feasibility is maintained since the pivot is made within the same solution. ■

The sufficient condition of Lemma 3 for an extreme point to be integral is however not necessary, as shown by the following counter-example.

Example 2 *Consider a set partitioning problem with the constraint matrix*

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}.$$

The linear programming relaxation of this problem has three extreme points, $(0,0,0,1,0,0)$, $(0,0,1,0,0,1)$ *and* $(0,1,0,0,1,0)$, *which are associated with respectively four, two, and two bases. All these bases have* $|det(B)| = 2$.

In the above example, none of the integer extreme points is associated with a unimodular basis, which causes problems for both the all-integer pivots and the integral simplex method.

A common way to find an initial basis in the simplex method is to introduce artificial variables, and it is also profitable to do so in order to initiate the all-integer pivots. From the next theorem it follows that if a set partitioning problem contains a full set of artificial variables, then each integer extreme point is associated with at least one unimodular basis.

Theorem 4 *If the constraint matrix of SPP contains the columns of an identity matrix of order m, then there exists a unimodular basis for each integer extreme point of* SPP_N^{LP}.

Proof: Let x be any integer extreme point and form the matrix $R = (a_j)_{j \in Q}$ where $Q = \{j : \ x_j = 1, \ j \in I\}$. The following procedure can then be used to create a unimodular basis B at x. Note that the matrix R has exactly one element with the value one in each row, and let n_j be the number of rows covered by column $j \in Q$ in R.

1. Let $q = 1$.

2. Pick the column $a_p \in R$ that covers row q and sort the rows of R such that a_p will cover rows $q, \ldots, q + n_p - 1$.

3. Let column $q + n_p - 1$ of B to be a_p, and if $n_p > 1$, let columns $i = q, \ldots, q + n_p - 2$ of B be the i:th identity matrix column.

4. If $q + n_p - 1 < |M|$, let $q = q + n_p$ and go to 2.

The resulting matrix B will be upper triangular with all diagonal elements equal to one. Hence, $det(B) = 1$ and the basis is unimodular. ■

To preserve integrality when pivoting, a sufficient condition is to restrict the pivots to be between unimodular bases only. The following example shows that a non-degenerate pivot-on-one from a basis which is not unimodular can lead to a non-integer extreme point.

Example 3 *This example uses the same constraint matrix as in Example 2 but it is here augmented with an identity matrix such that the variables* 1 *to* 6 *are associated with the columns of the original matrix, and the variables* 7 *to* 10 *are associated with the columns of an identity matrix. Consider the basis that is constituted by the columns 1, 2, 4, and 6; this basis has* $det(B) = 2$*, and is a basis for the integer extreme point* $x = (0, 0, 0, 1, 0, 0, 0, 0, 0, 0)$*. Making a non-degenerate pivot-on-one where* x_9 *enters the basis and* x_4 *leaves the basis, will lead to the basis consisting of the columns 1, 2, 6, and 9, which is associated with the non-integer extreme point* $x = (1/2, 1/2, 0, 0, 0, 1/2, 0, 0, 1, 0)$.

This example is also valid for both the integral simplex method and the all-integer pivots.

3.4.2 An illustration of the all-integer pivots Below, the problem SPP_{EX} will be used to illustrate the all-integer pivots, but we begin by returning to the pivots that are possible in the integral simplex method. In the example SPP_{EX}, all bases for the integer extreme points are unimodular.

To illustrate how to determine which of the pivots in Figure 3.3 are pivots-one-one, the simplex tableaus for vertices 3 and 4 of the example are given in Figure 3.5, and the associated movements in the graph are found in Figure 3.6.

Vertex 3:

		$-\bar{c}_2$			$-\bar{c}_5$	$\bar{z}$
x_1	1	2	0	0	-1	1
x_3	0	-1	1	0	1	0
x_4	0	-1	0	1	1	0

Vertex 4:

		$-\bar{c}_2$		$-\bar{c}_4$		$\bar{z}$
x_1	1	1	0	1	0	1
x_3	0	0	1	-1	0	0
x_5	0	-1	0	1	1	0

Figure 3.5: The simplex tableaus corresponding to vertices 3 and 4 in the example.

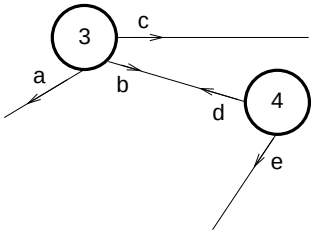
There are three possible pivots associated with leaving vertex 3.

- If x_2 enters and x_1 leaves the basis, the pivot will be non-degenerate, and not on a one-entry, hence the new solution will be the fractional extreme point x^C.
- If x_5 enters the basis and x_4 leaves the basis, the pivot is a degenerate pivot-on-one, and the solution will still be x^A.
- If x_5 enters the basis and x_3 leaves the basis, it is also a degenerate pivot-on-one, and the solution will still be x^A.

For vertex 4 there are two possible pivots.

- If x_4 enters and x_5 leaves the basis, it is a degenerate pivot-on-one, and the solution will still be x^A.
- If x_2 enters and x_1 leaves the basis, it is a non-degenerate pivot-on-one leading to x^B.

In Figure 3.7, all the pivots between the bases of SPP_{EX}^{LP} are categorised in the same manner as for the pivots associated with vertices 3 and 4.



vertex	movement	entering variable	leaving variable
3	a to vertex 8	x_2	x_1
3	b to vertex 4	x_5	x_4
3	c to vertex 5	x_5	x_3
4	d to vertex 3	x_4	x_5
4	e to vertex 6	x_2	x_1

Figure 3.6: The possible simplex pivots for leaving vertices 3 and 4.

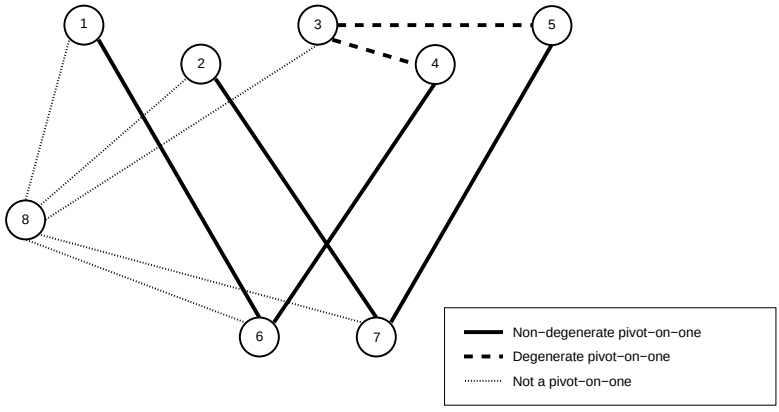


Figure 3.7: A categorisation of the possible simplex pivots between the bases of SPP_{EX}^{LP}.

The quasi-integrality property guarantees that there exists a non-degenerate pivot-on-one between every pair of adjacent integer solutions, but it says nothing about how to find it. Assume that we do not have an objective function and, for some unknown reason, would like to pivot from vertex 3 to vertex 1. It is then possible to do so by pivoting as shown by the arrows in Figure 3.8.

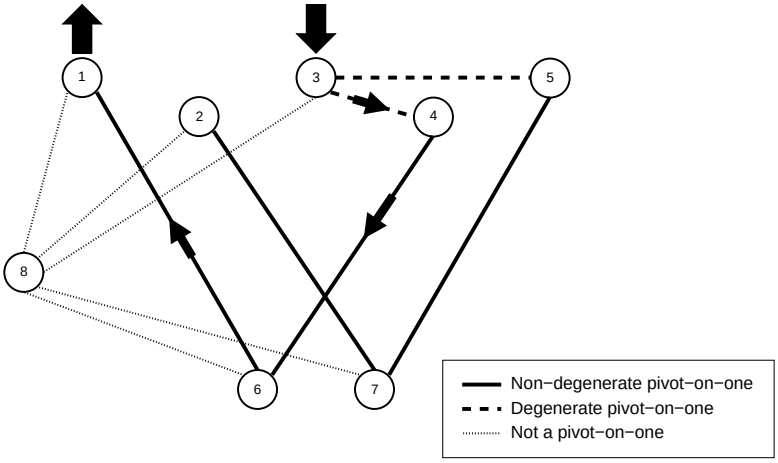


Figure 3.8: A possible path of pivots-on-one from vertex 3 to vertex 1.

At vertex 3, a choice is made between pivoting to vertex 4 or to vertex 5, and this choice is made without any knowledge of the consequences with respect to forthcoming pivots. Assuming that instead of choosing vertex 4 to follow on vertex 3, vertex 5 is chosen, then the path would be as shown by the arrows in Figure 3.9.

As can be seen from the figure, it is not possible to continue with a pivot-on-one from vertex 2, and therefore vertex 2 is a local optimum with respect to the integral simplex method. A possibility could of course be to reverse the pivots that led to this local optimum and choose another path, and if restricted to pivots-on-one, the only way to get to vertex 1 from vertex 2 is by reversing the pivots back to vertex 3 and there choose to pivot to vertex 4 instead. If this type of approach is used, it remains a true challenge how to develop efficient strategies for how to reverse pivots.

The possible pivots presented in Figure 3.7 only include the pivots-on-one and do not allow the possibility to perform a degenerate pivot on a minus one entry. If studying the same example from an all-integer perspective, the pivots between the following pairs of bases become possible, $(1,2)$, $(1,3)$, $(1,4)$, $(2,3)$, $(2,5)$, and $(4,5)$. In particular, the extension to all-integer pivots facilitates a degenerate pivot on a minus one entry at vertex 2, by letting x_3 become basic instead of x_4, leading to vertex 1. This pivot corresponds to the movement illustrated in Figure 3.10, where vertex 1 is reached from vertex 2 and vertex 2 is no longer a local optimum.

To summarise, the extension to all-integer pivots increases the number of possible pivots compared to the integral simplex method, and could therefore decrease the risk of getting trapped at a local optimum.

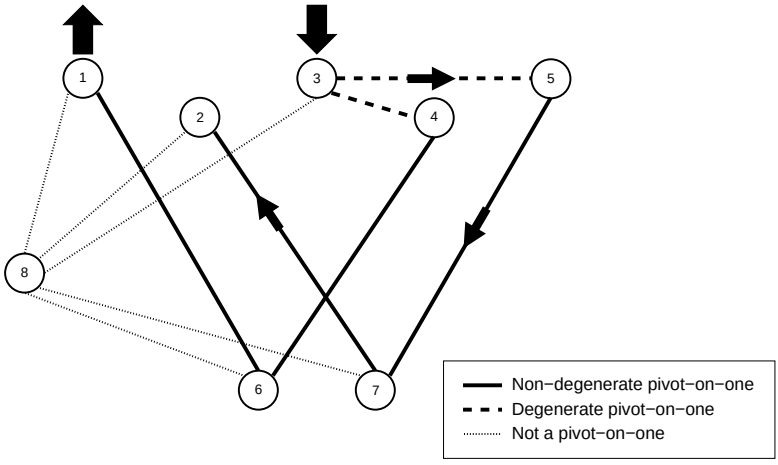


Figure 3.9: A path of pivots-on-one from vertex 3, not leading to vertex 1.

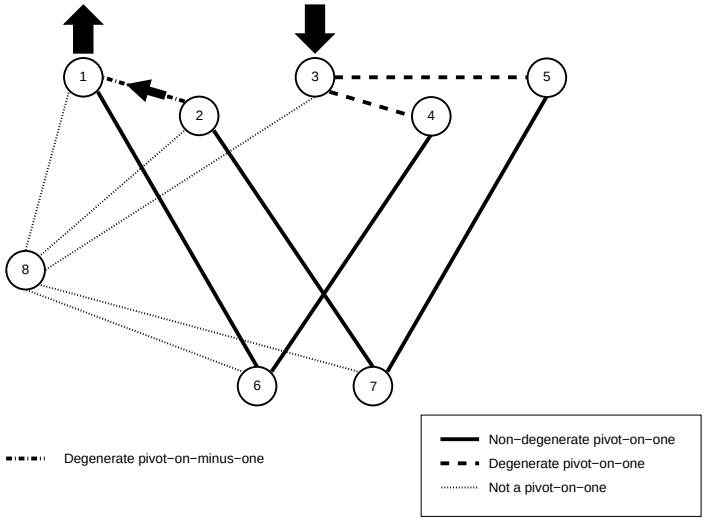


Figure 3.10: Vertex 1 can be reached from vertex 2 by a degenerate pivot on a minus one entry.

3.4.3 Basing a solution method on all-integer pivots The properties of a solution method using all-integer pivots are essentially the same as those of the integral simplex method, see Section 3.3, with one difference being that the risk of getting trapped at a local optimum is be decreased because there are more possible pivots.

A solution method based on all-integer pivots can be designed as follows.

1. Start from a purely artificial basis.

2. If possible, perform a non-degenerate all-integer pivot where the pivot entry is chosen according to some pivoting rule, and then go to 1.

3. If possible, perform a degenerate all-integer pivot where the pivot entry is chosen according to some pivoting rule, and then go to 1.

4. No all-integer pivot is possible. Stop.

An important question in the design of a method based on all-integer pivots is how to prevent cycling. As commented on in Section 3.3, we do not know of an anti-cycling rule for the integral simplex method. In a paper by Zörnig [33] it is claimed that it is possible to construct examples when the integral simplex method cycles using the following pivoting rules.

- The most negative reduced cost rule for the entering variable and the smallest (minimum) ratio rule for the leaving variable, where ties are broken according to the least index rule. (Danzig's rule)

- The most negative reduced cost rule for the entering variable and the smallest ratio rule for the leaving variable, where ties are broken according to the largest coefficient rule with respect to the pivot element.

- The entering variable is the one with the most negative ratio of reduced cost to the length of the vector corresponding to a unit change in the non-basic variable.

It is not clear to which version of the integral simplex method paper [33] refers however, and for this reason his results need to be investigated further before any conclusions can be drawn.

We have only briefly addressed the challenge of developing an anti-cycling rule for the all-integer pivots. The contributions we have made so far are examples of pivoting rules that do not prevent cycling, and the rules studied are the following.

- Danzig's rule, as described above, but with the modification that ratio comparison in the degenerate case is made with respect to the absolute values.

- Bland's rule, whereby an ordering of the variables is chosen and the eligible variable with the lowest index according to this ordering is selected, both when the tie is with respect to the entering and leaving variable.

The cycles we have constructed are presented in the following example.

Example 4 *Study the following instance of the set partitioning problem.*

$$z^* = \min \quad \begin{pmatrix} 9 & 9 & 9 & 9 & 5 & 1 & 3 & 2 & 1 & 1 & 1 & 1 \end{pmatrix} x$$

$$s.t. \quad \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} x = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

$$x \in \{0,1\}^{12}.$$

The following simplex tableau is obtained when the basic variables are x_6, x_{12}, x_8, and x_{10}.

	-7	-10	-7	-9	-3	0	-2	0	2	0	3	0	3
x_6	1	0	1	-1	0	1	1	0	2	0	1	0	1
x_{12}	-1	1	0	0	-1	0	1	0	0	0	-1	1	0
x_8	1	-1	1	0	1	0	0	1	1	0	2	0	1
x_{10}	0	0	-1	1	1	0	-1	0	-1	1	0	0	0

From this tableau, no non-degenerate all-integer pivot is possible, but degenerate ones are. If the pivot entry is chosen according to Danzig's rule, x_{11} shall enter the basis and x_{12} shall leave the basis, yielding the following tableau.

	-10	-7	-7	-9	-6	0	1	0	2	0	0	3	3
x_6	0	1	1	-1	-1	1	2	0	2	0	0	1	1
x_{11}	1	-1	0	0	1	0	-1	0	0	0	1	-1	0
x_8	-1	1	1	0	-1	0	2	1	1	0	0	2	1
x_{10}	0	0	-1	1	1	0	-1	0	-1	1	0	0	0

Again, no non-degenerate all-integer pivot is possible. If Danzig's rule for choosing the pivot entry is applied, x_{12} shall enter the basis and x_{11} shall leave the basis, and the first tableau is obtained again and a cycle is created.

It can be shown that if Bland's rule is used on the given example, a cycle between the following bases is obtained, $I_1 = \{6, 12, 8, 9\}$, $I_2 = \{6, 12, 8, 10\}$, and $I_3 = I_1 = \{6, 12, 8, 9\}$.

A comment to conclude this section is that we believe that it is a challenge to develop a pivoting rule for the all-integer pivots which both prevents cycling and yields a method that efficiently progresses to better solutions.

3.5 Concluding remarks The purpose of this chapter was to convey an understanding of the possibilities and challenges associated with utilising the quasi-integrality property for solving the set partitioning problem. This background was developed when preparing the work presented in Paper [IV] and Paper [V].

4 Thesis contributions and future work

Keeping in mind the background in the previous chapters, the purpose of this chapter is to summarise the contributions of the thesis and relate them to other research conducted in the respective areas. I will also comment on the possibilities for future research originating from our work.

Paper [I] and Paper [II] When we started the project on nurse scheduling, it was clear from the literature that there is a gap between real world nurse scheduling and the kind of model components that have been included in most of the research published on nurse scheduling. Quoting a paragraph of the conclusions in Burke et al. [7], *The state of the art of nurse rostering*, 2004:

> 'However, one key point that can be drawn from an analysis of the literature over the years is that very few of the developed approaches are suitable for directly solving difficult real world problems. Many of the models that have been presented and discussed are too simple to be directly applied to hospital wards. This point is even more pronounced if we are to consider modern problems in large and busy hospitals. There is a definite gap between much of the current state of the art in nurse scheduling research and the demanding and challenging requirements of today's hospital environments. A crucial scientific goal for nurse scheduling research in the future is to fully address the needs and requirements of the real world.'

On the basis of their conclusion, we initiated our project by addressing the question, 'What does it take to produce a high-quality and ready-to-use schedule for a regular Swedish nursing ward?' The contribution of Paper [I] is essentially an answer to this question, and presents a mathematical model tailored for a nursing ward with which we collaborated during the project. This mathematical model was solved by Cplex in order to produce schedules to be evaluated at the nursing ward. The promising results of the work leading to Paper [I] encouraged us to continue along this chosen path.

During the work with Paper [I], two challenges became especially clear, one associated with the modelling phase and the other associated with the solution phase. The modelling challenge arose from the awareness of the complexity of a model tailored for one particular ward and reads, 'How can the model be made generic enough to be useful on most Swedish nursing wards?'. For practical reasons, we concluded that it would be preferable to design the model in a way that the adaptation to a particular ward is made only by providing the appropriate data to the model, and not by making actual changes in the model. Moreover, such a model needs to be of a multi-objective type, where the importance of the constituent objectives can be changed between wards.

The second challenge was how to produce solutions to the more generic model, and again quoting Burke et al. [7],

> 'Exact methods represented some of the early approaches to solving the problem but they cannot cope with the enormous search spaces that are

> represented by real problems (at least on their own). There is evidence to suggest that there is research scope in hybridising exact methods with heuristic approaches. Although it is the case that exact methods may be able to aid heuristics it is clear that some kind of (hybrid) heuristic method offers the only realistic way of tackling such difficult and challenging problems in the foreseeable future. However, heuristic methods alone cannot cope with the dynamic and uncertain nature of modern problems (see below). The current state of the art is represented by interactive approaches which incorporate problem specific methods and heuristics (that are derived from specific knowledge of the problem and the required constraints) with powerful modern metaheuristics, constraint based approaches and other search methods.'

Compared to the tailored model in Paper [I], the generic model was going to be more complex and should be capable of handling larger wards. Because of that, we concluded that using CPLEX or any other standard integer programming software was not an option. Instead, we needed to design a solution method that exploits the structure of the problem.

Both of the challenges described above were addressed in the work leading to Paper [II]. The main contribution of the paper however, is the two case studies that involve delivering schedules to be used by two quite different nursing wards. The purpose of the paper is to show what it takes to do deliver high-quality and ready-to-use schedules in practice.

Our nurse scheduling project turned out to be very successful, and the result was that our developed model and solution method were very useful for delivering schedules to a variety of nursing wards. Our conclusion from the project was that if our software could be put to actual use in Swedish health care, it could contribute to improving the working conditions for nurses. We also realised that the use of our software has the potential to save time, since nowadays the scheduling is made manually, which is a process that can take up to 900 hours per year on a regular ward of 30 nurses. If instead our software is used to create the schedules, a ward can reallocate this time to be used for patient care, and hence improve the quality of the service provided.

At the time writing this thesis introduction, the software designed in the project has been put to use in the form of a scheduling service provided by the company SCHEMAGI AB, and the case studies presented in Paper [II] were actually conducted in collaboration with this company when it was in a start-up phase. Since the software is now in commercial use, the details of the design of both the model and the solution method must be kept a secret, and are therefore not published. Paper [II] instead gives an overview of the modelling framework and some comments on the design of the solution method.

Future work originating from this project could involve improvements to the software as well as the development of other optimisation-based software useful in connection with the scheduling, for example decision support tools for activity scheduling or for strategic planning.

Paper [III] From a chronological point of view, Paper [III] is the last contribution to this thesis. The inspiration for this work comes from the development of optimality conditions to be used in the methodology suggested in Paper [V]. In brief, when working on Paper [V], we needed optimality conditions for a column generation setting where the dual solution is typically not optimal with respect to a linear programming relaxation of the restricted master problem, which in the case of Paper [V] is a set partitioning problem.

When studying the literature, we found that this type of optimality conditions has been derived for the case when a general binary master problem contains convexity constraints, see Sweeney and Murphy [22], which is a paper from the late seventies. Independently of Sweeney and Murphy [22], similar but weaker such conditions have been suggested for master problems of set partitioning type with cardinality constraints, and these conditions have been used as a vital component in methods presented in Baldacci et al. [3] and Baldacci et al. [4].

The recent paper by Baldacci et al. [4] presents excellent computational results, and this led us to the following motives for deriving more generic optimality conditions for binary programming column generation using non-optimal dual solutions. Firstly, since the approach used in Baldacci et al. [4] has proved to be successful, it could be interesting to apply a similar approach to other problem structures, and such an attempt requires that the optimality conditions can be transferred to the problem structure at hand. Secondly, with inspiration deriving from Sweeney and Murphy [22], it was clear to us how the optimality conditions used in Baldacci et al. [4] could be improved upon.

The contribution of Paper [III] is the generic optimality conditions, together with a general discussion of how they can be applied in the design of a solution method. The paper does not contain computational results produced by us, but instead, refers to the examples that are already known from the literature and explains how these can be derived as special cases of a general methodology facilitated by the more generic optimality conditions.

I consider the contribution of Paper [III] to provide a specific piece to the column generation research puzzle. The results of the paper connect previous work presented in [22], [3], and [4] and facilitate future work along the lines already suggested in those papers.

Paper [IV] and Paper [V] The project resulting in Papers [IV] and [V] addresses column generation methodologies for solving the set partitioning problem when it has a huge number of columns. Common to both papers is that the quasi-integrality property is utilised in the method design, see Chapter 3.

Our suggested methodology unites two successful ways of solving large scale integer programs, namely combinatorial meta-heuristics and column generation. Characteristic for the meta-heuristics that I refer to as being combinatorial, is that the integrality requirements on the variables are never relaxed; instead it is common to relax the feasibility with respect to some well chosen constraints. An advantage of not relaxing integrality is that each solution encountered during the course of an algorithm represents a solution to the problem that can be interpreted in terms of the decisions possible, even if it is not necessarily feasible. This is especially useful when

solving real world problems where the importance of fulfilling various constraints can differ.

When solving integer programs by column generation, the most commonly used approach is branch-and-price, see Chapter 2, in which the integrality requirements are relaxed during the course of an algorithm. A crucial advantage of using column generation on a problem with a huge number of variables is that only a relatively small subset of the variables is explicitly included in the model, and that the rest of the variables are taken into account only implicitly.

The methodology developed in our project does not relax the integrality requirements and does, like in column generation, work on a restricted master problem which is successively augmented with profitable columns. The combination of these two properties makes our approach fundamentally different from other methods. Since the method relies on utilising the quasi-integrality property, it can of course only be applied to problems possessing this property. However, since it is common that the master problems in a column generation setting has a set partitioning structure, the applicability is still wide.

The relation between Paper [IV], that was written first, and Paper [V], is as follows. Paper [IV] introduces the theoretical cornerstones for how the quasi-integrality can be utilised in a solution method, and proposes an exact method to show that such a method exists. The contribution of Paper [IV] is the introduction of a novel idea solution concept for the set partitioning problem, and we believe that the proposed method is mainly of theoretical interest.

Based on the cornerstones introduced in Paper [IV], the purpose of Paper [V] is to get one step closer to a method that can be of practical use for solving large scale set partitioning problems. In Paper [V], this is done by extending the basic method components introduced in Paper [IV] with components that facilitate the design of a meta-heuristic type of method. Paper [V] also reports the results of a very simple implementation of a method within the suggested framework, in order to show that the behaviour is as expected and as desired.

What remains as important future work within this line of research is to make a full implementation of a method within the proposed framework in order to evaluate it against other methods. This is however a challenging task that requires both wise choices in the design of the method and a huge amount of implementation work. An attractive property of the method is that during the course of the algorithm, feasible and successively improved solutions will be obtained. The method would then both be capable of handle a huge number of variables and possess the same type of advantages as the combinatorial types of meta-heuristics. For practical reasons, it has not been possible to include a full implementation of a method of this type in this thesis project.

Publication status

Paper [I] has been published in *Health Care Management Science*.

Paper [II] is to be published in the book *Systems analysis tools for better healthcare delivery*, Editors P. Pardalos, P. Georgiev and P. Papajorgji, Springer Science + Business Media.

Paper [III] is submitted for publication.

Paper [IV] has been published in *European Journal of Operational Research*.

Paper [V] is submitted for publication.

Conference presentations

Operational research peripatetic post-graduate program, EURO conference for young OR researchers, Cadiz, Spain, 2011. Presented parts of Chapter 3, Paper [IV], and Paper [V].

The 20th International Symposium of Mathematical Programming, Chicago, Illinois, USA, 2009. Presented parts of Chapter 3, Paper [IV], and Paper [V].

Operations research in healthcare, Svenska operationsanalysföreningens seminarie om operationsanalys inom hälso- och sjukvård, 2009. Presented parts of Paper [I] and Paper [II].

Third Nordic Optimization Symposium, Stockholm, 2009. Presented parts of Chapter 3, Paper [IV], and Paper [V].

Column generation 2008, workshop in Aussois, France. Presented parts of Chapter 3, Paper [IV], and Paper [V].

Second Nordic Optimization Symposium, Oslo, 2007. Presented parts of Chapter 3, Paper [IV].

EURO XXII, Prague, 2007. Presented parts of Paper [I].

References

[1] L. H. Appelgren. A column generation algorithm for a ship scheduling problem. *Transportation Science*, 3:53–68, 1969.

[2] E. Balas and M. W. Padberg. On the set-covering problem. *Operations Research*, 20:1152–1161, 1972.

[3] R. Baldacci, L. Bodin, and A. Mingozzi. The multiple disposal facilities and multiple inventory locations rollon-rolloff vehicle routing problem. *Computers and Operations Research*, 33:2667–2702, 2006.

[4] R. Baldacci, N. Christofides, and A. Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming, series A*, 115:351–385, 2008.

[5] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. Vance. Branch-and-price: Column generation for solving huge integer programs (abbreviated version). In J. R. Birge and K. G. Murty, editors, *Mathematical Programming, State of the Art*, pages 186–207. Braun- Brumfield, 1994.

[6] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1998.

[7] E. K. Burke, P. de Causmaecker, G. van den Berghe, and H. van Landeghem. The state of the art of nurse rostering. *Journal of Scheduling*, 7:441–499, 2004.

[8] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

[9] L. C. Edie. Traffic delays at toll booths. *Journal of the Operations Research Society of America*, 2:107–138, 1954.

[10] A. T. Ernst, H. Jiang, M. Krishnamoorthy, B. Owens, and D. Sier. An annotated bibliography of personnel scheduling and rostering. *Annals of Operations Research*, 127:21–144, 2004.

[11] A. T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153:3–27, 2004.

[12] L. R. Ford and D. R. Fulkerson. A suggested computation for maximal multi-commodity network flows. *Management Science*, 5:97–101, 1958.

[13] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9:849–859, 1961.

[14] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting-stock problem-part II. *Operations Research*, 11:863–888, 1963.

[15] J. Hellstrand, T. Larsson, and A. Migdalas. A characterization of the uncapacitated network desing polytope. *Operations Research Letters*, 12:159–163, 1992.

[16] K. Hoffman and M. Padberg. Set covering, packing and partitioning problems. In C. A. Floudas and P. M. Pardalos, editors, *Encyclopedia of optimization*. Springer-Verlag, Berlin, second edition, 2008.

[17] E. Johnson. Modelling and strong linear programs for mixed integer programming. In S. W. Wallace, editor, *Algorithms and model formulations in mathematical programming*, NATO ASI Series 51, pages 1–41, 1989.

[18] M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53:1007–1023, 2005.

[19] K. G. Murty. *Linear programming*. John Wiley & Sons, New York, NY, 1983.

[20] D. M. Ryan and B. A Foster. An integer programming approach to scheduling. In A. Wren, editor, *Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling*, pages 269–280, Amsterdam, 1981. North-Holland.

[21] T. Sastry. One and two facility network design revisited. *Annals of Operations Research*, 108:19–31, 2001.

[22] D. J. Sweeney and R. A. Murphy. A method of decomposition for integer programs. *Operations Research*, 27:1128–1141, 1979.

[23] G. L. Thompson. An integral simplex algorithm for solving combinatorial optimization problems. *Computational Optimization and Applications*, 22:351–367, 2002.

[24] V. A. Trubin. On a method of solution of integer linear programming problems of a special kind. Translated by V. Hall. *Soviet Mathematics Doklady*, 10: 1544–1546, 1969.

[25] F. Vanderbeck and L. A. Wolsey. An exact algorithm for IP column generation. *Operations Research Letters*, 19:151–159, 1996.

[26] F. Vanderbeck. On dantzig-wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, 2000.

[27] F. Vanderbeck. Branching in branch-and-price: a generic scheme. *Mathematical Programming, series A*, 2010. Online first, DOI: 10.1007/s10107-009-0334-1.

[28] D. Villeneuve, J. Desrosiers, M. E. Lübbecke, and F. Soumis. On compact formulations for integer programs solved by column generation. *Annals of operations research*, 2005.

[29] W. E. Wilhelm. A technical review of column generation in integer programming. *Optimization and Engineering*, 2:159–200, 2001.

[30] L. A. Wolsey. *Integer programming.* John Wiley & Sons, New York, NY, 1998.

[31] V. A. Yemelichev, M. M. Kovalev, and M. K. Kravtsov. *Polytopes, graphs and optimisation.* Translated by G. H. Lawden. Cambridge University Press, Cambridge, 1984.

[32] P. Zörnig. *Degeneracy graphs and simplex cycling.* Springer-Verlag, Berlin, 1991.

[33] P. Zörnig. Systematic construction of examples for cycling in the simplex method. *Computers & Operations Research*, 33:2247–2262, 2006.