Solving Large-scale
Open Pit Mining Production Scheduling Problems
by Integer Programming

Solving Large-scale Open Pit Mining Production Scheduling Problems by Integer Programming

Diplomarbeit bei
Prof. Dr. Rolf H. Möhring

vorgelegt von
Ambros M. Gleixner

am Institut für Mathematik
Fakultät II – Mathematik und Naturwissenschaften
Technische Universität Berlin

Juni 2008

Abstract

Since the initial application of mathematical optimisation methods to mine planning in 1965, the Lerchs-Grossmann algorithm for computing the ultimate pit limit, operations researchers have worked on a variety of challenging problems in the area of open pit mining. This thesis focuses on the open pit mining production scheduling problem: Given the discretisation of an orebody as a block model, determine the sequence in which the blocks should be removed from the pit, over the lifespan of the mine, such that the net present value of the mining operation is maximised.

In practise, when some material has been removed from the pit, it must be processed further in order to extract the valuable elements contained therein. If the concentration of valuable elements is not sufficiently high, the material is discarded as waste or stockpiled. Realistically-sized block models can contain hundreds of thousands of blocks. A common approach to render these problem instances computationally tractable is the aggregation of blocks to larger scheduling units.

The thrust of this thesis is the investigation of a new mixed-integer programming formulation for the open pit mining production scheduling problem, which allows for processing decisions to be made at block level, while the actual mining schedule is still computed at aggregate level. A drawback of this model in its full form is the large number of additional variables needed to model the processing decisions. One main result of this thesis shows how these processing variables can be aggregated efficiently to reduce the problem size significantly, while practically incurring no loss in net present value.

The second focus is on the application of lagrangean relaxation to the resource constraints. Using a result of Möhring et al. [41] for project scheduling, the lagrangean relaxation can be solved efficiently via minimum cut computations in a weighted digraph. Experiments with a bundle algorithm implementation by Helmberg [25] showed how the lagrangean dual can be solved within a small fraction of the time required by standard linear programming algorithms, while yielding practically the same dual bound.

Finally, several problem-specific heuristics are presented together with computational results: two greedy sub-MIP start heuristics and a large neighbourhood search heuristic. A combination of a lagrangean-based start heuristic followed by a large neighbourhood search proved to be effective in generating solutions with objective values within a 0.05% gap of the optimum.

Zusammenfassung

Seit 1965 Lerchs und Grossmann die ersten mathematischen Algorithmen zur Erstellung optimierter Abbaupläne im Tagebau entwarfen, wurde an einer Vielzahl anspruchsvoller Optimierungsprobleme in diesem Anwendungsbereich geforscht. Die vorliegende Arbeit widmet sich dem Problem der Produktionsplanung im Tagebau ("open pit mining production scheduling problem"): Ausgehend von einer Diskretisierung des Erzvorkommens, einem sogenannten Blockmodell, besteht die Aufgabe darin, eine zeitliche Abbaureihenfolge für die Blöcke zu bestimmen, die den diskontierten Ertrag der Ausgrabungen maximiert.

In der Praxis muss abgebautes Material weiterverarbeitet werden, um Erz von minderwertigem Erdreich zu trennen. Dieser Vorgang ist allein bei ausreichend hohem Erzgehalt rentabel, Material von zu niedrigem Erzgehalt wird entsorgt oder bevorratet. Blockmodelle realistischer Größe bestehen aus hunderttausenden von Blöcken. Ein üblicher Ansatz zur Lösung von Problemen dieser Größenordnung ist die Aggregation von Blöcken zu größeren Einheiten.

Ein Schwerpunkt dieser Arbeit besteht in der Untersuchung einer neuartigen Formulierung des Problems als gemischt-ganzzahliges Programm, welches eine selektive Weiterverarbeitung des abgebauten Materials auf Blockebene ermöglicht, den Abbau selbst jedoch auf Aggregatsebene belässt. Ein Nachteil des neuen Modells besteht in der großen Anzahl zusätzlicher Variablen zur Steuerung der Weiterverarbeitung. Als Hauptresultat präsentiert die Arbeit eine erfolgreiche Methode zur Aggregation dieser Variablen, um den zusätzlichen Rechenaufwand zu begrenzen.

Der zweite Schwerpunkt liegt auf der Anwendung der Lagrange-Relaxation auf die Ressourcenbeschränkungen des Tagebauproduktionsplanungsproblems. Mithilfe eines Resultats von Möhring et al. [41] für Projektplanungsprobleme kann das verbleibende Programm effizient durch die Berechnung minimaler Schnitte in einem Digraphen gelöst werden. Ergebnisse von Experimenten mit einem Bündelalgorithmus in einer Implementierung von Helmberg [25] zeigen, dass das der Lagrange-Relaxation entsprechende duale Problem in einem Bruchteil der Zeit gelöst werden kann, die zur direkten Lösung der LP-Relaxation benötigt wird, und dennoch im wesentlichen dieselbe duale Schranke liefert.

Zum Abschluss werden verschiedene problemspezifische Heuristiken vorgestellt und anhand von Ergebnissen aus Experimenten bewertet: zwei Startheuristiken sowie eine Verbesserungsheuristik gemäß dem "large neighbourhood search"-Paradigma. Die Kombination einer Lagrange-basierten Startheuristik mit anschließendem Verbesserungsschritt stellt sich dabei als besonders effektiv heraus und erzeugt auf einer Reihe realistischer Testinstanzen hochwertige Lösungen mit Zielfunktionswerten innerhalb 0.05% des Optimums.

Acknowledgements

I was introduced to the field of optimisation in open pit mining by Gary Froyland, who was my supervisor during my honours year at the University of New South Wales in Sydney, Australia. The mixed-integer programming model investigated in this thesis is his brainchild. Many thanks for the guidance and support I received during this inspiring year go to him, as well as to Irina Dumitrescu and Natashia Boland – working together with you was as fruitful as it was pleasant. I want to thank the Mine Optimisation Group at BHP Billiton Pty. Ltd. in Melbourne, in particular Peter Stone, Mark Zuckerberg and Merab Menabde, for being able to present and discuss my work and for providing me with the data that made it possible to evaluate the approaches developed on real-world open pit mines.

In Berlin, I am indebted to my supervisor Rolf Möhring, especially that he understood and agreed when I intended to adjust the topic of this thesis. Thanks to Marco Lübbecke for his guidance in laying out this thesis. Many thanks go to the optimisation group at the Konrad Zuse Institute Berlin, in particular to Tobias Achterberg, Timo Berthold and Kati Wolter for taking their time to introduce me to SCIP. Equally much I want to thank Andreas Bley and Marc Pfetsch for their helpful and clarifying comments on my work.

Finally, special thanks go to the Egerváry Research Group of András Frank at the Eötvös Loránd University in Budapest, Hungary, where I had the opportunity to study for three months. Exchanging mathematical as well as non-mathematical thoughts with all of you and learning from your culture and language was a very enriching experience. In particular, I want to thank László Végh for his remarks on the complexity proofs.

Contents

Chapter 1

Introduction

The history of mathematical optimisation in the field of mine planning dates back as far as 1965, when Lerchs and Grossmann [37] considered the problem of determining the ultimate pit limit of an open pit mine, i.e. the part of the orebody the mining of which will return the highest profit, not considering the course of time. To this end, they proposed two algorithms for the maximum closure problem[1] on a directed acyclic graph with node weights: a dynamical programming approach as well as a graph-theoretical algorithm. The latter has become widely known as the *Lerchs-Grossmann algorithm* and has over years been the benchmark algorithm for the maximum closure problem.[2]

Computing the ultimate pit limit is only a first step in open pit mine planning and this field provides many more applications for mathematical optimisation. One particular challenge posed by all of these problems is the large-scale nature of realistically-sized mine models. The special focus of this thesis will be on the problem of determining an actual mining schedule over a time span of several years such that the net present value is maximised, the so-called *open pit mining production scheduling problem*.

Section 1.1 introduces basic terminology and modelling assumptions appearing in open pit mining. An outline of the thesis is provided in Section 1.2, while Section 1.3 gives details on the data sets, hardware and software used for experiments.

1.1 A brief introduction to open pit mining

Fricke [18, pp. 2] gives an extensive description of terminology and techniques appearing in mining applications. The following will provide the basic notions.

[1] Given a directed acyclic graph with node weights, the maximum closure problem is to find a subset of nodes with maximum weight such that no arc leaves the subset.

[2] Only over the past 15 years, a network flow approach has become more efficient due to the advances in network flow algorithms, see e.g. Caccetta et al. [8] and Hochbaum and Chen [28].

A *mineral* is a natural resource that is located in the earth and can be extracted. *Ore* specifically is a naturally appearing aggregation of – one or more – solid minerals valuable enough to be mined. The *grade* of some material is the percentage of ore contained therein, the rest being waste. *Base metals* are special types of ore, such as copper or zinc, which are roughly understood to be metals that are sold in pure form, where "impurities" can be disregarded.

Base metals are mostly found close to the surface and can be extracted by the method of *open pit mining*: Waste on the top is removed first, until valuable material can be extracted in small pieces (called *mining*). As a second procedure, the mined material has to be refined to the final product (called *processing*). Mined material will only be processed if the prospective profit exceeds the cost. The grade of ore at which refinement is no longer profitable (and the material discarded as waste) is termed *cutoff grade*.

For the planning of open pit mining operations, the orebody is generally approximated by subdividing it into small mining units called *blocks*. This is essentially a discretisation of the orebody into a three-dimensional array of mostly regular blocks, called the *block model*, and thus well-suited for adoption in a mathematical model. A major assumption here, as in many applications found in the literature, is the complete knowledge of the geological properties of each block which are essential for mine planning.[3] For open pit mining of base metals it is often sufficient to know the rock and ore tonnage.

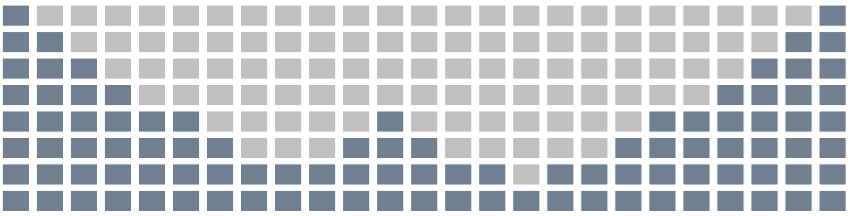

Figure 1.1: Example of a two-dimensional block model with ultimate pit limit

The brighter blocks form the so-called *ultimate pit limit* of the mine: Given fixed profits for each block, the ultimate pit limit is the contour of those blocks which can be feasibly mined so as to yield the maximum profit – not considering the course of time, i.e. the net present value objective. An optimal schedule for the open pit mining production scheduling problem can always be found within the ultimate pit limit, see e.g. [9].

This information is subsequently used to calculate the cost of mining and processing and to make an assumption about the return that selling the final product will yield. Amongst other things due to varying metal prices, this involves a considerable level of risk making best- and worst-case analysis crucial. Furthermore, since a mine is usually operated for several years, the time value

[3] An important area of research deals with methods taking into account the uncertainty of the available orebody data. It may be expected that advances in solving deterministic models will in turn be able to improve the solution of stochastic models.

of money has to be accounted for. By discounting the cash flows occurring during each time period to the present, one obtains the so-called *net present value*, which is the significant value to be considered for maximising the profit of a mining operation. Again, we will assume that this has been taken into account and hence fixed values can be assigned to mining costs and processing profits for each block.

Among the various problems appearing in mine planning, this thesis will focus on the *open pit mining production scheduling problem* (OPMPSP).[4] The standard assumptions made for this approach, most of which were already mentioned above, can be summarised as outlined by Froyland et al. [19]:

- A deterministic block model is given as input data,
- mining and processing costs, the selling price of the product and future discount rates are perfectly known into the future,
- the infrastructure is fixed (though not necessarily constant) throughout the life of the mine (e.g. mining and processing capacities),
- grade control is assumed to be perfect (i.e. once a block has been blasted, its content is precisely known) and
- the requirement to maintain safe wall slopes (and remove overlying material before the underlying) can be modelled as precedence relations between the blocks (i.e. for each block to be mined, a cone of predecessor blocks has to be extracted previously).

Then, the OPMPSP consists of determining a feasible order for the mining of blocks which will maximise the net present value. This can be interpreted as a constrained scheduling problem by identifying blocks with jobs, where mainly two types of constraints[5] are present:

- removal of overlying blocks before the underlying ones and maintenance of safe wall slopes (*precedence constraints*), and
- limitation on mining and processing activities (*resource constraints*).

One aspect which makes mine planning particularly interesting and challenging is the large-scale nature of realistically-sized open pit mines, which can consist of hundreds of thousands of blocks. Despite more than forty years

[4] For a general survey of the major applications of mathematical optimisation in the field of mining, see Fricke [18, pp. 45].

[5] There exist many variants of the OPMPSP which incorporate further constraints, such as limiting conditions on impurities, or which consider the additional possibility of stockpiling mined material for delayed processing, etc. Some of the methods developed here may also be applicable to those more specific models.

of mathematical research, as Fricke [18, p. 5] points out, "determination of methods to achieve this in time frames acceptable in industry for models of realistic size remains an open question in mine planning."

1.2 Mathematical prerequisites, thesis outline and contribution

1.2.1 Mathematical prerequisites

The thesis addresses readers with a good understanding of the fundamental concepts of linear and mixed-integer programming. The theory used includes duality for linear programmes, LP- and lagrangean relaxation for mixed-integer programmes and total unimodularity of matrices. For the complexity proofs, the reader should be familiar with polynomial time reducibility. We will follow Möhring et al. [41] with a solution approach for a lagrangean relaxation subproblem for which minimum s-t-cuts are computed in a weighted digraph. Basic knowledge in (algorithmic) graph theory is recommended for this section. Also recommended is an understanding of the fundamental components of LP-based branch-and-cut algorithms for general MIP-solving. We refer to the books by Nemhauser and Wolsey [42] and Wolsey [50].

Algorithms for solving the lagrangean dual and the theory of column aggregation for linear programmes are applied and will be described where needed.

1.2.2 Outline of the thesis

The thesis focuses on the problem of production scheduling for open pit mines, short OPMPSP. Chapter 2 introduces the problem by giving a precise description of the OPMPSP as understood in this thesis. We continue with a survey of prominent integer and mixed-integer programming formulations found in the literature. Subsequently, a new mixed-integer programming model *D-MIP* is presented, allowing for integrated cutoff grade optimisation. This formulation serves as the basis for the thesis – its advantages over other formulations found in the literature are pointed out and possible shortcomings are addressed. We highlight the connection to two closely related problems, the precedence-constrained knapsack problem and the resource-constrained project scheduling problem. Finally, we prove that optimising the new mixed-integer programming model introduced is NP-hard.

Before devising and evaluating specialised methods computationally, Chapter 3 provides the theoretical background and analyses basic structural properties of programme *D-MIP*: We show how the standard LP-relaxation can be reduced in size and describe an application of lagrangean relaxation by resource constraints. The structure yielded by the knapsack constraints on the processing variables in *D-MIP* is analysed and appears prominently in Chapter 5. We also give the somewhat complementary result that the feasible

region without resource constraints is integral. Finally, we mention how to integrate valid inequalities into the reduced LP-relaxations and the lagrangean approach under a reasonable condition.

Our central aim is naturally to obtain high quality OPMPSP-solutions. To this end, heuristic methods are presented and tested in Chapter 6. Before, however, Chapter 4 is concerned with the in some sense complementary question: How can the quality of primal solutions be evaluated most efficiently by computing dual bounds? Being able to obtain good bounds on the optimal objective value within reasonable running times is a key ingredient for many optimisation algorithms, such as for a classical branch-and-bound approach. The computation of dual bounds is not only of theoretical importance, though. It is essential also in practise in order to evaluate the quality of solutions and provide engineers with confidence in the mine plans developed.

Chapter 4 will give an overview over the methods commonly applied for solving the lagrangean dual – the problem of computing the best dual bound achievable by the lagrangean relaxation approach introduced in Chapter 3. On basis of computational results, we can compare the solution of the LP-relaxation by standard linear programming algorithms with the solution of the lagrangean dual by a bundle algorithm. The latter proves to outperform the LP-approach clearly with respect to the running time, while yielding practically the same dual bound. Solving the lagrangean dual also provides us rapidly with near-optimal dual multipliers associated with the resource constraints, which are utilised in the next chapter.

Chapter 5 draws upon the theory from Section 3.2 to devise methods for reducing the large number of additional processing variables in *D-MIP*. To this end, a specialised column aggregation approach is presented using so-called "binnings". Computational experiments show that this helps to reduce the problem size significantly, while incurring practically no loss in objective function value.

With efficient methods for computing dual bounds at hand and having reduced the problem size significantly, Chapter 6 proposes methods for obtaining high-quality primal solutions. We describe two start heuristic based on a common generic greedy scheme: one proceeding by time periods, another based on a solution of the lagrangean relaxation from Section 3.4. Computational results show the lagrangean-based heuristic to be superior to the time-based one, but both produce solutions of very reasonable to high quality. A method is discussed how these heuristics can be improved further by adding a more global perspective to the greedy steps. We also propose a simple, yet effective large neighbourhood search heuristic, which serves to improve the solutions obtained by the start heuristics even further.

To conclude, Section 6.3 describes how the primal heuristics can be brought together with the lagrangean dual bound computation from Chapter 4 in a

customised branch-and-bound algorithm. Details on preliminary experiments using SCIP [3] as a branch-and-bound framework are reported. Integration of valid inequalities into the lagrangean approach appears to be one promising direction for further research.

1.2.3 Contribution of the thesis

The major contribution of this thesis is the investigation of a new mixed-integer programming formulation for the OPMPSP, which allows for *integrated cutoff grade optimisation*. We highlight the knapsack structure on additional processing variables and show how their values in an optimal solution are distributed according to their oregrade. Using this insight, in Chapter 5 we can devise an efficient column aggregation scheme according to what we called "binnings". Computational results show how this leads to significant reduction of the number of variables, while incurring only negligible decrease in objective value. These results appear also in the paper by Boland et al. [6].

Also, the computational experiments conducted for Chapter 4 could establish the efficiency of a lagrangean approach which uses a bundle algorithm to solve the lagrangean dual and, following a result of Möhring et al. [41], efficient minimum cut computations in a weighted digraph for optimising the subproblems.

Furthermore, Chapter 6 proposes a series of integer programming heuristics, which proved very effective in computational experiments.

Preliminary experiments using SCIP [3] as a branch-and-bound framework indicated how the quick dual bound computation and the successful primal heuristics could be brought together to develop an OPMPSP-specific lagrangean-based branch-and-cut approach in further research.

1.3 Data sets, hardware and software used in the computational experiments

To evaluate the theoretical methods presented, the author conducted a series of experiments on data sets of three open pit mines provided by the research partner BHP Billiton Pty. Ltd.[6] All of the data sets are based on a simple block model, but because the number of blocks is usually too large to solve the OPMPSP, blocks are aggregated to bigger mining units, simply called "aggregates". (See Chapter 2 for a more detailed explanation.)

Rock and ore data is given for each single block and precedence relations between blocks are determined to maintain safe wall slopes. Precedence relations between the aggregates stem from the block precedence relations. Scheduling periods are time periods of one year each with a discount factor of

[6] `http://www.bhpbilliton.com/`

10% per year. Realistic values for mining costs and processing profits as well as for mining and processing capacities were chosen by BHP Billiton Pty. Ltd. The three data sets are as follows:

- Data set "marvin" is based on an artificially created block model provided with the Whittle 4X mine planning software [39]. It consists of 8513 blocks and was aggregated at three different resolutions, to 115, 296 and 1038 aggregates. The lifespan of this mine is 15 years, i.e. the profitable part of the orebody can be fully mined within that time frame. Each aggregate has on average 2.4, 3.1 and 5.0 immediate predecessor aggregates, respectively.[7]

- Data set "wa" is based on the block model of a real-world open pit mine in Western Australia, consisting of 96821 blocks aggregated to 125 aggregates. Here, the aggregates are so-called "panels", i.e. single layers of blocks without block-to-block precedence relations. Each aggregate has an average of 2.0 immediate predecessor aggregates. The lifespan of this mine is 25 years.

- Data set "ca" is based on the block model of a real-world open pit mine in Canada, consisting of 29266 blocks, which was aggregated to 121 aggregates. Each aggregate has on average 2.2 immediate predecessor aggregates. While for each of the models above, one fixed scenario for rock and ore content of the blocks is given, for this data set, 25 different scenarios for rock and ore tonnages were computed. The lifespan of this mine is 15 years.

This gives 29 problem instances, which the author used for computational evaluation of the methods presented. Experiments on the "marvin"-instances will highlight in particular the impact of the aggregation resolution when using the same data for fixed rock and ore tonnage. From a computational point of view, the instance marvin-1038-8513 is particularly difficult with the largest number of aggregates and precedence relations. In contrast, the "ca"-scenarios allow for tests on computationally more and less difficult instances based on the same precedence model. The "wa"-model stands out due to the large number of blocks within each aggregate, and will therefore be of particular interest for evaluation of the aggregation techniques presented in Chapter 5.

The algorithms were implemented in C++ and conducted on a personal computer with a 32-bit Intel Pentium 4 3 GHz processor and 2 GB RAM. All algorithms were run single-threaded. For solving linear and mixed-integer linear programmes, the state-of-the-art software CPLEX 11.0 [29] was used. The

[7] I.e. $\sum_{k=1}^{K} \bar{\mathcal{P}}(k)/K$ with the notation introduced in Section 2.1.

ConicBundle software [25] of Helmberg provided an implementation of a bundle algorithm for solving the lagrangean dual in Chapter 4. For the minimum cut computations used to solve the lagrangean relaxation, an implementation of the pseudoflow algorithm by Chandran and Hochbaum [12] was integrated. Finally, the author conducted preliminary experiments with a lagrangean-based branch-and-bound algorithm, which are described in Section 6.3. For these experiments, the constraint integer programming solver SCIP 1.05 [3] was embedded and used as a general branch-and-bound framework.

Chapter 2

Modelling the open pit mining production scheduling problem

In the previous chapter, we only gave a vague description of the open pit mining production scheduling problem. This chapter will make it precise and give various integer and mixed-integer programming formulations.

The first section specifies in detail our understanding of the OPMPSP and introduces important notation used throughout the thesis. Section 2.2 presents solution approaches found in the literature, in particular focusing on integer programming models. For these models, cutoff grades, i.e. ore grades below which mined material is discarded as waste, need to be determined prior to the optimisation process. In Section 2.3 we introduce a new mixed-integer programming formulation which will be the focus of the thesis. This model distinguishes itself from the other models found in the literature by allowing for *integrated cutoff grade optimisation*. The decision which blocks are to be processed is not made a priori, but within the model and thus subject to optimisation itself. Section 2.4 highlights the connection to two closely related problems, the *precedence-constrained knapsack problem* and the *resource-constrained project scheduling problem*. The last section gives a proof of NP-hardness.

2.1 General model outline and notation

In this section we will specify the details of our OPMPSP-model. First, we need to clarify what is understood by a feasible mine schedule. Given the block model of an open pit mine, our goal is to determine the order in which the blocks should be mined such as to maximise profit in a net present value sense. In classical machine scheduling, one would want to determine the exact order

of blocks to achieve this goal. However, note that changes in the order on small time scales have only a very minor effect on the net present value objective. Hence, determining the exact order of blocks appears to be unnecessary and would likely increase the computational effort disproportionally.

Instead we subdivide the lifespan of the mine into discrete, not necessarily uniform time periods $1, \dots, T$. Then, by a schedule we understand an assignment of blocks to one or more time periods during which the block is mined and processed. While in most scheduling applications every job must be scheduled, we do not require that every block is removed from the pit. Mining will be stopped as soon as no further increase in the net present value can be achieved, and the unmined material will remain in the pit.

One major condition on a feasible mine plan is that overlying material must be removed before the underlying while maintaining safe wall slopes. As outlined in Section 1.1, we make the standard assumption that this requirement can be expressed by means of precedence relations between blocks: For each block we are given a set of predecessor blocks which need to be mined beforehand. Mathematically, this is represented by a directed acyclic predecessor graph with blocks as nodes and arcs pointing from blocks to their predecessors. Without loss of generality we may assume that this predecessor graph

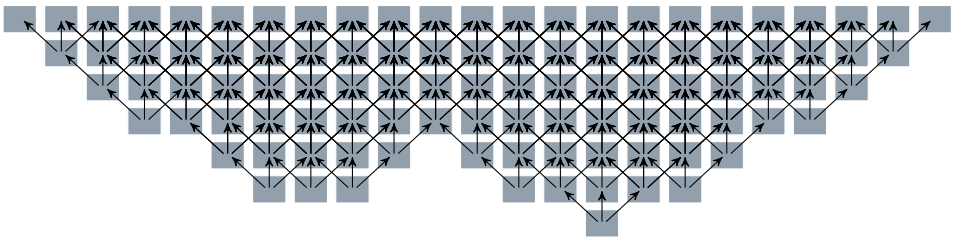

Figure 2.1: Example of a two-dimensional block model with transitively reduced precedence relations

Safe wall slopes are determined by an angle of 45°. Precedence arcs point from blocks to their immediate predecessors.

is transitively reduced and consider only so-called immediate predecessors for each block.[1]

In our model, several blocks may be scheduled during the same time period, even if one is the predecessor of another, but any feasible schedule has to obey the *precedence constraints* that

- a block can only be mined during some time period $t \in \{1, \dots, T\}$ if all of its predecessor blocks have been mined completely before or during time period t,

[1] A directed acyclic digraph $D = (V, A)$ is transitively reduced, if for all pairs of arcs (u, v), $(v, w) \in A$, the arc (u, w) is not also contained in A. By deleting such "superfluous" arcs, any directed acyclic digraph can be transitively reduced.

and the *resource constraints* demanding that

- during each time period the amount of rock mined and processed must not exceed the mining and processing capacities, respectively.

High resolution block models of realistically-sized open pit mines usually contain too many blocks for the OPMPSP to be solved directly. Therefore, a common approach in mine planning is to aggregate the block model, i.e. to partition the set of blocks into subsets, which we will call *(mining) aggregates*, and schedule the fewer number of aggregates instead of the blocks. The block precedence relations naturally induce precedence relations between aggregates: Aggregate $\mathcal{K}_1$ is a predecessor of aggregate $\mathcal{K}_2$ if and only if $\mathcal{K}_1$ contains a predecessor block of some block in $\mathcal{K}_2$. In the block precedence graph, this corresponds to contracting the node set formed by each of the aggregates.

There are different methods for aggregating block models, for details we refer to the literature. One recently proposed method is for example the aggregation according to so-called *fundamental trees* by Johnson, Dagdelen and Ramazan [30, 45]. They define a fundamental tree as a minimal group of blocks that have positive undiscounted value in total and obey the precedence constraints between the blocks in the group. In the experiments conducted, the data sets used were already provided with aggregated block models.

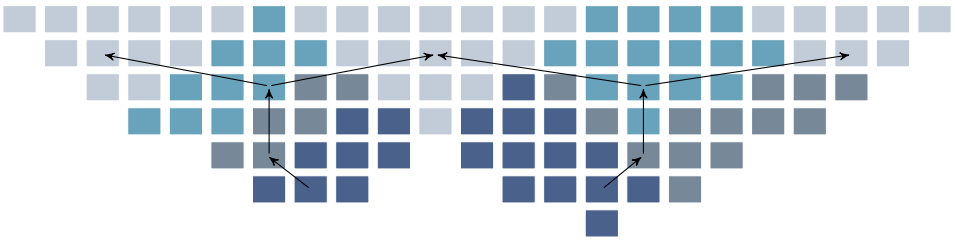

Figure 2.2: Example of a two-dimensional aggregated block model

Precedence relations between aggregates are computed on basis of block precedence relations and subsequently transitively reduced.

An aggregated block model can theoretically be viewed as a block model at lower resolution. However, an essential feature of an aggregate is the underlying block structure. While a block is assumed to have a homogeneous distribution of ore, an aggregate may, and usually will contain blocks of different oregrades. In this context, it needs to be emphasised again that the mining operations consist of two procedures: the removal of rock material from the pit, which we will call *mining*, and the subsequent refinement of the mined material, referred to as *processing*. Even if the mining schedule is determined at aggregate level, making processing decisions at aggregate level appears to be highly suboptimal. Processing decisions clearly depend on the oregrade,

thus – unless an aggregate shows a completely homogeneous distribution of ore – they should not be made at aggregate level, but at a higher resolution, ideally at block level.

By default, the OPMPSP assumes a simple block model or aggregated block model as input. On basis of the above discussion, we will consider a basic block model together with an aggregation as input. To summarise, an instance of the OPMPSP is given by the following data:

- A *block model* of an open pit mine with precedence relations between the blocks: Let N be the total number of blocks and $\mathcal{N} = \{1, \dots, N\}$ denote the set of blocks (or block indices, to be exact). For each block $i \in \mathcal{N}$, we denote by $\mathcal{P}(i) \subseteq \mathcal{N}$ the set of its immediate predecessors, i.e. those neighbouring blocks which have to be fully mined before the mining of block i can be started.

- An aggregation of blocks into *(mining) aggregates*: Let K be the total number of aggregates, then all aggregates $\mathcal{K}_1, \dots, \mathcal{K}_K \subseteq \mathcal{N}$ form a partition of the set of blocks. The precedence relations at block level naturally determine precedence relations for the aggregates: Denote by $\bar{\mathcal{P}}(k) \subseteq \{1, \dots, K\}$ the (indices of) immediate predecessor aggregates of $\mathcal{K}_k$, for $k \in \{1, \dots, K\}$.

- A *discretisation of the mine's lifespan* with discount rates for each time period: Let T denote the number of (not necessarily uniform) time periods and $r_1, \dots, r_T$ denote the discount rates for each time period. Then the net present value of a cash flow C occurring in some time period $t \in \{1, \dots, T\}$ is $\prod_{s=1}^{t} \frac{C}{1+r_s}$.

- *Mining costs* and *processing profits*: For block $i \in \mathcal{N}$ and time period $t \in \{1, \dots, T\}$, let $c_{i,t}$ denote the cost of mining and let $p_{i,t}$ denote the profit from processing block i in time period t, already in the net present value sense. This yields mining costs and processing profits for the aggregates: $\bar{c}_{k,t} = \sum_{i \in \mathcal{K}_k} c_{i,t}$ and $\bar{p}_{k,t} = \sum_{i \in \mathcal{K}_k} p_{i,t}$ for all $k \in \{1, \dots, K\}$ and $t \in \{1, \dots, T\}$.

- *Rock tonnages* and *mining and processing capacities*: Let $a_i > 0$ denote the rock tonnage of block $i \in \mathcal{N}$ and define $\bar{a}_k = \sum_{i \in \mathcal{K}_k} a_i$ as the rock tonnage of aggregate $\mathcal{K}_k$, $k \in \{1, \dots, K\}$. The total amount of rock being mined and processed per time period is limited: Let U_t^{m} and U_t^{p} denote the capacities for mining and processing during time period $t \in \{1, \dots, T\}$, respectively.

2.2 Previous work

2.2.1 Heuristic approaches and dynamic programming

The main approaches to solving the OPMPSP found in the literature are heuristics, dynamic programming and integer programming. One well-known heuristic approach, which has also been used in commercial mine planning software, for instance the Whittle software package [39], is based on the Lerchs-Grossmann algorithm for determining the ultimate pit, see introduction to Chapter 1. Here, a series of nested ultimate pits is created by decreasing the sale price of ore from its true value step by step. The schedule is then defined by excavating these nested pits in order from smallest to largest pit.

A dynamic programming approach to the OPMPSP seems natural and is described by Onur and Dowd [43]. However, they noted that a purely dynamic programming approach is unlikely to be able to solve realistically-sized problem instances due to the large size of the state space.

The signicant improvements to mixed-integer programming solvers in recent years allow larger and larger problems to be solved optimally or near-optimally. This prompts further investigation of integer and mixed-integer programming techniques to solving the OPMPSP and the remainder of this thesis will focus on this approach. In the following sections, we present several typical formulations found in the literature. Similar models are increasingly used in commercial mine planning software, for instance the Whittle software package [39]. Finally, we present a novel mixed-integer programming model allowing for integrated cutoff grade optimisation, which will be the subject of the following chapters.

2.2.2 Integer programming formulations

Among the integer programming formulations found in the literature, we want to highlight a relatively recent one given by Caccetta and Hill [9] in the framework of a specialised branch-and-cut algorithm. Their formulation assumes a basic block model with homogeneous ore distribution within each block.

The set of blocks $\mathcal{N}$ is partitioned into one set of high-value ore blocks $\mathcal{O}$ to be processed and another set of waste blocks $\mathcal{W}$. This way, a fixed value $\hat{p}_{i,t}$ can be assigned to each block $i \in \mathcal{N}$ for the profit gained from mining this block in time period $t \in \{1, \dots, T\}$, in the net present value sense. With the notation introduced in Section 2.1 we can write

$$\hat{p}_{i,t} = \begin{cases} p_{i,t} - c_{i,t} & \text{if } i \in \mathcal{O}, \\ -c_{i,t} & \text{if } i \in \mathcal{W}. \end{cases} \tag{2.1}$$

The formulation uses the binary decision variables

$$x_{i,t} = \begin{cases} 1 & \text{if block } i \text{ is mined during} \\ & \text{one of the time periods } 1, \dots, t, \\ 0 & \text{otherwise,} \end{cases}$$

for all $i \in \mathcal{N}$, $t \in \{1, \dots, T\}$.

For simplicity of notation, we add variables $x_{i,0} = 0$ for all $i \in \mathcal{N}$. In this model, the mining of a block $i \in \mathcal{N}$ cannot be spread out over several time periods: $x_{i,t} - x_{i,t-1}$ is 1 if and only if block i is mined during time period $t \in \{1, \dots, T\}$, 0 otherwise. They also include continuous reporting variables m_t for the rock tonnage of ore blocks mined in time period $t \in \{1, \dots, T\}$. The Caccetta-Hill formulation reads

maximise

$$\sum_{i=1}^{N} \sum_{t=1}^{T} \hat{p}_{i,t} \left(x_{i,t} - x_{i,t-1} \right) \tag{2.2}$$

subject to

$$x_{i,t-1} - x_{i,t} \leqslant 0 \qquad \text{for all } i \in \mathcal{N},\ t \in \{2, \dots, T\}, \tag{2.2a}$$

$$x_{i,t} - x_{j,t} \leqslant 0 \qquad \text{for all } i \in \mathcal{N},\ j \in \mathcal{P}(i),\ t \in \{1, \dots, T\}, \tag{2.2b}$$

$$\sum_{i \in \mathcal{O}} a_i (x_{i,t} - x_{i,t-1}) - m_t = 0 \qquad \text{for all } t \in \{1, \dots, T\}, \tag{2.2c}$$

$$L_t^{\mathcal{O}} \leqslant m_t \leqslant U_t^{\mathcal{O}} \qquad \text{for all } t \in \{1, \dots, T\}, \tag{2.2d}$$

$$\sum_{i \in \mathcal{W}} a_i (x_{i,t} - x_{i,t-1}) \leqslant U_t^{\mathcal{W}} \qquad \text{for all } t \in \{1, \dots, T\}, \tag{2.2e}$$

$$x_{i,t} \in \{0, 1\} \qquad \text{for all } i \in \mathcal{N},\ t \in \{1, \dots, T\}, \tag{2.2f}$$

$$x_{i,0} = 0 \qquad \text{for all } i \in \mathcal{N}. \tag{2.2g}$$

Constraints (2.2a) let each block be mined at most once. (2.2b) ensures the precedence constraints. (2.2c) and (2.2d) give maximal and minimal amounts for the total rock tonnage of ore blocks mined per time period. The total rock tonnage of waste blocks mined per time period is bounded above by constraint (2.2e). Despite the continuous $m_1, \dots, m_T$, this is not a "real" mixed-integer programming formulation, since reporting variables can easily be substituted, yielding a pure binary integer programme.

The Caccetta-Hill-formulation is also the basis for much of the work of Fricke [18]. He generalises the partition into ore and waste blocks by considering arbitrarily many so-called attributes $r \in \mathcal{R}$ such as ore, waste, impurities, etc. For each of these attributes, q_i^r denotes the total tonnage of attribute r

in block $i \in \mathcal{N}$. The total tonnage of each attribute mined per period is constrained by an upper bound.

As in the Caccetta-Hill-formulation, an a priori decision on which blocks to process and which blocks to discard as waste after mining must be made. Then as in (2.1) a fixed value $\hat{p}_{i,t}$ can be assigned to each block $i \in \mathcal{N}$ for the profit gained from mining this block in time period $t \in \{1, \dots, T\}$, in the net present value sense.

Fricke's generalised integer programming formulation for the OPMPSP (see [18, pp. 97]) reads

maximise

$$\sum_{i=1}^{N} \sum_{t=1}^{T} \hat{p}_{i,t} \left(x_{i,t} - x_{i,t-1} \right) \tag{2.3}$$

subject to

$$x_{i,t-1} - x_{i,t} \leqslant 0 \qquad \text{for all } i \in \mathcal{N},\ t \in \{2, \dots, T\}, \tag{2.3a}$$

$$x_{i,t} - x_{j,t} \leqslant 0 \qquad \text{for all } i \in \mathcal{N},\ j \in \mathcal{P}(i),\ t \in \{1, \dots, T\}, \tag{2.3b}$$

$$\sum_{i=1}^{N} q_i^r (x_{i,t} - x_{i,t-1}) \leqslant U_t^r \qquad \text{for all } t \in \{1, \dots, T\},\ r \in \mathcal{R}, \tag{2.3c}$$

$$x_{i,t} \in \{0, 1\} \qquad \text{for all } i \in \mathcal{N},\ t \in \{1, \dots, T\}, \tag{2.3d}$$

$$x_{i,0} = 0 \qquad \text{for all } i \in \mathcal{N}. \tag{2.3e}$$

Constraints (2.3a) ensure that each block is mined at most once and are sometimes called *reserve constraints*. (2.3b) are the *precedence constraints*. Constraints like (2.3c) limiting certain "activities" of the mining operations are often called *production* or *resource constraints*.

Typically, the total amount of rock which can be mined and processed per time period is limited by the mine's infrastructure. This can be expressed by including two attributes in the model. For block $i \in \mathcal{N}$, define q_i^0 as its rock tonnage a_i, and

$$q_i^1 = \begin{cases} a_i & \text{if block } i \text{ will be processed after mining,} \\ 0 & \text{otherwise.} \end{cases}$$

Then $\sum_{i=1}^{N} q_i^0 (x_{i,t} - x_{i,t-1})$ is the total rock tonnage mined, $\sum_{i=1}^{N} q_i^1 (x_{i,t} - x_{i,t-1})$ is the total rock tonnage processed during time period $t \in \{1, \dots, T\}$. These amounts can be limited by a mining capacity U_t^0 and a processing capacity U_t^1 for each time period $t \in \{1, \dots, T\}$.

2.2.3 Mixed-integer programming formulations

The binary integer programming formulations presented above restrict themselves to schedules that force each block to be mined completely during one period. These models usually cannot be solved on the most highly resolved block models, but only on aggregated block models as described in Section 2.1. Therefore, this restriction may become more and more unrealistic as the dimensions of the open pit mine and the sizes of blocks respectively aggregates grow – the mining of a block or aggregate is rather a continuous process than a discrete one.

Smith [48] presents a qualitative description of a mixed-integer programming formulation for the OPMPSP. His model allows for blocks to be mined fractionally, but no explicit formulation is given. Within the framework of stochastic mine planning, Menabde et al. [40] also present a mixed-integer programming formulation allowing for fractions of blocks to be mined. Fricke [18, pp. 103] simplifies this to a formulation for the deterministic OPMPSP and gives a generalised version for arbitrarily many attributes as in the integer programming formulation (2.3). We have binary decision variables

$$x_{i,t} = \begin{cases} 1 & \text{if block } i \text{ may be mined} \\ & \text{during time periods } t, \dots, T, \\ 0 & \text{otherwise,} \end{cases}$$

and continuous variables

$$y_{i,t} \in [0,1] \text{ as the fraction of block } i \text{ mined in time period } t$$

for $i \in \mathcal{N}$ and $t \in \{1, \dots, T\}$. With the same notation as in (2.3), we get

maximise

$$\sum_{i=1}^{N} \sum_{t=1}^{T} \hat{p}_{i,t}\, y_{i,t} \tag{2.4}$$

subject to

$$x_{i,t-1} - x_{i,t} \leqslant 0 \qquad \text{for all } i \in \mathcal{N},\ t \in \{2, \dots, T\}, \tag{2.4a}$$

$$\sum_{s=1}^{t} y_{i,s} - x_{i,t} \leqslant 0 \qquad \text{for all } i \in \mathcal{N},\ t \in \{1, \dots, T\}, \tag{2.4b}$$

$$x_{i,t} - \sum_{s=1}^{t} y_{j,s} \leqslant 0 \qquad \text{for all } i \in \mathcal{N},\ j \in \mathcal{P}(i),\ t \in \{1, \dots, T\}, \tag{2.4c}$$

$$\sum_{i=1}^{N} q_i^r\, y_{i,t} \leqslant U_t^r \qquad \text{for all } t \in \{1, \dots, T\},\ r \in \mathcal{R}, \tag{2.4d}$$

$$x_{i,t} \in \{0,1\} \qquad \text{for all } i \in \mathcal{N},\ t \in \{1, \dots, T\}, \tag{2.4e}$$

$$0 \leqslant y_{i,t} \leqslant 1 \qquad \text{for all } i \in \mathcal{N},\ t \in \{1, \dots, T\}. \tag{2.4f}$$

A block $i \in \mathcal{N}$ with $x_{i,t} = 1$ for some time period $t \in \{1, \dots, T\}$ can now be

mined in fractions spread out over the time periods from $\min\{t \mid x_{i,t} = 1\}$ to T. The precedence constraints are ensured by (2.4b) and (2.4c) together. Both the objective function and the resource constraints (2.4d) now contain only the continuous variables.

2.3 A new mixed-integer programming formulation with integrated cutoff grade optimisation

2.3.1 The model

The formulations from the previous section all require an a priori decision about cutoff grades, i.e. the oregrade above which mined material is worthwhile to be processed. A lowest cutoff grade is always given by the oregrade at which the profit returned from selling the final product equals the processing costs. However, note that due to the net present value objective and limited processing capacities, the optimal cutoff grade typically varies over time. Profit made during early time periods pays off more, making it profitable to reach high value material as early as possible. Often the material of high oregrade is located towards the bottom of the pit and the overlying material might have to be mined faster than it can be processed due to the limited infrastructure. Therefore, even material with an oregrade sufficiently high to allow for profitable processing might be discarded as waste in early time periods.[2] This typically yields cutoff grades which are decreasing over time.

The formulations from the previous section all take a block model with homogeneous oregrade distribution as input. In practise, however, they cannot be solved on the most highly resolved block models, but only on aggregated block models as introduced in Section 2.1. While the mining decisions have to be made at aggregate level, processing decisions depend on the oregrade and should therefore take into account the heterogeneous oregrade distribution within the aggregates. This can be achieved by introducing continuous variables for the processing of each block. The mixed-integer programming formulation presented in the following was introduced by Boland et al. [6].

To model the mining process, we use three sets of variables. For the mining process, we have the binary decision variables

$$x_{k,t} = \begin{cases} 1 & \text{if aggr. } \mathcal{K}_k \text{ may be mined} \\ & \text{during time periods } t, \ldots, T, \\ 0 & \text{otherwise,} \end{cases}$$

[2] In practise, this material is of course stockpiled and saved for processing during later periods when the cutoff grade is lower again.

and the continuous variables

$$y_{k,t} \in [0,1] \text{ as the fraction of aggreg. } \mathcal{K}_k \text{ mined during time period } t$$

for all $k \in \{1,\dots,K\}$ and $t \in \{1,\dots,T\}$. Additionally, we define for each block $i \in \mathcal{N}$ and $t \in \{1,\dots,T\}$

$$z_{i,t} \in [0,1] \text{ as the fraction of block } i \text{ processed during time period } t.$$

With the notation introduced in Section 2.1, this gives the following mixed-integer programming model, which we will later refer to as *D-MIP*:[3]

maximise

$$\sum_{k=1}^{K}\sum_{t=1}^{T} -\bar{c}_{k,t}\, y_{k,t} \;+\; \sum_{i=1}^{N}\sum_{t=1}^{T} p_{i,t}\, z_{i,t} \tag{2.5}$$

subject to

$$x_{k,t-1} - x_{k,t} \leqslant 0 \qquad \text{for all } k \in \{1,\dots,K\},\ t \in \{2,\dots,T\}, \tag{2.5a}$$

$$\sum_{s=1}^{t} y_{k,s} - x_{k,t} \leqslant 0 \qquad \text{for all } k \in \{1,\dots,K\},\ t \in \{1,\dots,T\}, \tag{2.5b}$$

$$x_{k,t} - \sum_{s=1}^{t} y_{\ell,s} \leqslant 0 \qquad \text{for all } k \in \{1,\dots,K\},\ \ell \in \bar{\mathcal{P}}(k),\ t \in \{1,\dots,T\}, \tag{2.5c}$$

$$z_{i,t} - y_{k,t} \leqslant 0 \qquad \text{for all } k \in \{1,\dots,K\},\ i \in \mathcal{K}_k,\ t \in \{1,\dots,T\}, \tag{2.5d}$$

$$\sum_{k=1}^{K} \bar{a}_k y_{k,t} \leqslant U_t^{\mathrm{m}} \qquad \text{for all } t \in \{1,\dots,T\}, \tag{2.5e}$$

$$\sum_{i=1}^{N} a_i z_{i,t} \leqslant U_t^{\mathrm{p}} \qquad \text{for all } t \in \{1,\dots,T\}, \tag{2.5f}$$

$$x_{k,t} \in \{0,1\} \qquad \text{for all } k \in \{1,\dots,K\},\ t \in \{1,\dots,T\}, \tag{2.5g}$$

$$0 \leqslant y_{k,t} \leqslant 1 \qquad \text{for all } k \in \{1,\dots,K\},\ t \in \{1,\dots,T\}, \tag{2.5h}$$

$$0 \leqslant z_{i,t} \leqslant 1 \qquad \text{for all } i \in \mathcal{N},\ t \in \{1,\dots,T\}. \tag{2.5i}$$

The first part of the objective function, $\sum_{k=1}^{K}\sum_{t=1}^{T} -\bar{c}_{k,t}\, y_{k,t}$, evaluates to the cost for the mining operations, which is always negative. The second part, $\sum_{i=1}^{N}\sum_{t=1}^{T} p_{i,t}\, z_{i,t}$, gives the profit returned from processing and will be nonnegative for any optimal schedule (z-variables with negative objective coefficient can always be set to their lower bound zero). Due to the reserve constraints (2.5a), the mining of every aggregate is started at most once.

[3] *D-MIP* stands for "Disaggregated MIP", which will become clear in the context of Chapter 5.

Constraints (2.5b) and (2.5c) together ensure precedence-feasibility. Note that instead of (2.5b) we could also write $y_{k,t} \leqslant x_{k,t}$. However, (2.5b) results in a stronger LP-relaxation. By (2.5d), only as much can be processed as has been mined already. (2.5e) and (2.5f) guarantee that the resource constraints on mining and processing activities are respected.

2.3.2 Discussion

The improvements in the new model presented over the ones found in the literature are two-fold: First, processing decisions can be made at a higher spatial resolution than the mining decision and thus results in a more accurate modelling of the real orebody. Second, the separation of processing and mining decisions is a crucial improvement in itself. The models found in the literature require an a priori determination of cutoff grades, which might be suboptimal. With the new model, optimal cutoff grades are automatically determined within the optimisation procedure itself.

Some points of criticism need to be addressed which might be aimed at the practicality of this model. A major assumption is hidden in constraints (2.5d). When a fraction of some aggregate has been mined in some time period, then it is implicitly assumed that all of the blocks in this aggregate have been mined by the same fraction and thus may be processed by this same amount. This is certainly unrealistic, since the mining proceeds block by block. When half of an aggregate, say, has been mined, then this should mean that half of the blocks have been mined completely instead of all of the blocks having been mined by half. This results in a violation of the precedence constraints, since material at the bottom of an aggregate may be accessed before all of the overlying has been removed.

In one type of aggregate model important in mine planning applications, blocks are aggregated to so-called “panels”, for instance in the data set “wa” described in Section 1.3. Panels are single layers of blocks and thus contain no block-to-block precedence relations. In this case, the model presented will not at all result in the violation of precedence constraints. For other aggregations, aggregates might consist of a small number of several consecutive layers. In an optimal solution, however, aggregates will typically be fully mined and processed within one or two consecutive time periods. Therefore, the impact of possibly violated precedence constraints may be expected to be minor.

Another possibility to overcome this drawback is simply to consider a version where each aggregate has to be mined completely within one time period. In this case, no block-to-block precedence constraints will be violated. This corresponds to setting $y_{k,t} = x_{k,t} - x_{k,t-1}$ for $t \in \{2, \dots, T\}$ and $y_{k,1} = x_{k,1}$.

Substituting the y-variables accordingly yields the programme *D-MIP'*,

maximise

$$\sum_{k=1}^{K}\sum_{t=1}^{T} -\bar{c}_{k,t}(x_{k,t}-x_{k,t-1}) \;+\; \sum_{i=1}^{N}\sum_{t=1}^{T} p_{i,t}\, z_{i,t} \tag{2.6}$$

subject to

$$x_{k,t-1} - x_{k,t} \leqslant 0 \qquad \text{for all } k \in \{1,\dots,K\},\ t \in \{2,\dots,T\}, \tag{2.6a}$$

$$x_{k,t} - x_{\ell,t} \leqslant 0 \qquad \text{for all } k \in \{1,\dots,K\},\ \ell \in \bar{\mathcal{P}}(k),\ t \in \{1,\dots,T\}, \tag{2.6b}$$

$$z_{i,t} - (x_{k,t} - x_{k,t-1}) \leqslant 0 \qquad \text{for all } k \in \{1,\dots,K\},\ i \in \mathcal{K}_k,\ t \in \{1,\dots,T\}, \tag{2.6c}$$

$$\sum_{k=1}^{K} \bar{a}_k(x_{k,t} - x_{k,t-1}) \leqslant U_t^{\mathrm{m}} \qquad \text{for all } t \in \{1,\dots,T\}, \tag{2.6d}$$

$$\sum_{i=1}^{N} a_i z_{i,t} \leqslant U_t^{\mathrm{p}} \qquad \text{for all } t \in \{1,\dots,T\}, \tag{2.6e}$$

$$x_{k,0} = 0 \qquad \text{for all } k \in \{1,\dots,K\}, \tag{2.6f}$$

$$x_{k,t} \in \{0,1\} \qquad \text{for all } k \in \{1,\dots,K\},\ t \in \{1,\dots,T\}, \tag{2.6g}$$

$$0 \leqslant z_{i,t} \leqslant 1 \qquad \text{for all } i \in \mathcal{N},\ t \in \{1,\dots,T\}. \tag{2.6h}$$

Because this programme results from *D-MIP* by mere addition of the constraints $x_{k,t} = \sum_{s=1}^{t} y_{k,s}$ for all $k \in \{1,\dots,K\}$, $t \in \{1,\dots,T\}$, *D-MIP* can be viewed as a relaxation of *D-MIP'*. All the theory and methods developed in the following chapters for *D-MIP* are fully applicable to *D-MIP'*. We consider *D-MIP* as a better approximation of the reality, just because it allows the continuous mining process to be performed across the borders given by the time discretisation.

From a computational point of view, *D-MIP* might seem inferior to the models from Section 2.2.2 because it contains the large number of additional z-variables for processing. One of the main results of this thesis is presented in Chapter 5 and addresses exactly this objection. Many of the processing variables can be aggregated to a single variable and thus only comparatively few of them remain. Computational results show that this reduction in size, although heuristic, does not come at the cost of objective value.

A more general point of criticism towards all the integer programming models presented here could be that a real-world mine plan must consider

many technical details which are not included in the constraints of these models: haul roads, minimal footprint for excavation equipment, etc. However, the claim is not that the solutions of the OPMPSP-models will be completely practical in every technical detail. Nevertheless, it is certainly possible to construct a realistic mine plan on the basis of OPMPSP-schedules. If we can compute optimal or near-optimal solutions of the OPMPSP, this will in turn yield superior real-world mine plans.

2.4 Closely related problems

In the following, we will briefly present two NP-hard optimisation problems, the precedence-constrained knapsack problem and the resource-constrained project scheduling problem, both of which are structurally closely related to the OPMPSP. The notation used is restricted to this section.

2.4.1 The precedence-constrained knapsack problem

Suppose a set of items $i \in \mathcal{N} = \{1, \dots, N\}$ with non-negative weights w_i and profits c_i is given. The problem of choosing a subset of these items of total weight that does not exceed a given capacity U ("packing a knapsack") such that the overall profit is maximised is called the *0-1 knapsack problem*.[4] It can be written as a binary integer programme

$$\begin{aligned} \text{maximise} \quad & \sum_{i=1}^{N} c_i x_i \\ \text{subject to} \quad & \sum_{i=1}^{N} w_i x_i \leqslant U, \\ & x_i \in \{0,1\} \qquad \text{for all } i \in \mathcal{N}, \end{aligned} \tag{KP}$$

where for any $i \in \mathcal{N}$

$$x_i = \begin{cases} 1 & \text{if item } i \text{ is included in the knapsack,} \\ 0 & \text{otherwise.} \end{cases}$$

Along with many variants, the 0-1 knapsack problem has been widely studied since the early days of mathematical optimisation as it is the simplest prototype of an integer programme. A thorough presentation of knapsack problems is given by Kellerer et al. [32].

Of particular interest in relation to the OPMPSP is the so-called *precedence-constrained knapsack problem*, a variant that is further complicated by additionally introducing precedence constraints on the items to be included in the knapsack. If we denote by $\mathcal{P}(i)$ the set of predecessors of item $i \in \mathcal{N}$, i.e. the

[4] Generally, the weights and profits are assumed to be integer, but many results hold equally for rational and real values.

set of items which need to be included before item i, then the precedence-constrained knapsack problem can be written as

$$\begin{aligned} \text{maximise} \quad & \sum_{i=1}^{N} c_i x_i \\ \text{subject to} \quad & \sum_{i=1}^{N} w_i x_i \leqslant U, \\ & x_i \leqslant x_j \qquad \text{for all } i \in \mathcal{N},\ j \in \mathcal{P}(i), \\ & x_i \in \{0,1\} \qquad \text{for all } i \in \mathcal{N}. \end{aligned} \tag{PCKP}$$

Both *KP* and *PCKP* yield NP-hard optimisation problems, as shown by Kellerer [32, pp. 483], for example.

The OPMPSP-formulations and *PCKP* are related in the sense that the OPMPSP-formulations "contain" *PCKP*-structure. Fricke's binary integer programme (2.3) from Section 2.2.2 transforms directly into *PCKP* for a single time period and one attribute. The *PCKP*-structure can also be found in the formulations *D-MIP* and *D-MIP'* from Section 2.3 and can for instance be used to derive valid inequalities, as will be shown in Chapter 3.

2.4.2 The resource-constrained project scheduling problem

The problem of sequencing or scheduling a number of tasks of a given project arises in many different areas of application, reaching from production planning to project management. *Project scheduling* is a generic term for a class of problems of scheduling a set of precedence-constrained jobs such as to optimise a given objective function subject to various constraints, which may include due dates, release dates or resource constraints. Different types of objectives are considered, such as minimising the duration of a schedule or maximising its net present value.

More formally, we are given a set $\mathcal{N}$ of jobs $1, \ldots, N$ with integral processing times $p_i \geqslant 0$, $i \in \mathcal{N}$, which have to be scheduled over time periods $1, \ldots, T$ of unit length. A schedule is represented by a vector $S = (S_1, \ldots, S_N)$ of the jobs' start times. Several jobs may be processed in one time period as long as they respect the other constraints present. All jobs must be scheduled during the time available.

Precedence constraints are modelled by a weighted digraph $G = (\mathcal{N}, \mathcal{E})$, where to each arc $(i,j) \in \mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ we assign a time lag $d_{i,j}$ of integral length. Then, $(i,j) \in \mathcal{E}$ means that job i cannot be started earlier than $d_{i,j}$ periods after start of job j, i.e. $S_i \geqslant S_j + d_{i,j}$. Ordinary precedence constraints could be modelled by letting $d_{i,j} = p_j$, if job j is a predecessor of job i.

For the OPMPSP, the variant of the *resource-constrained project scheduling problem* is of particular interest, where additionally a set of resources $\mathcal{R}$ is given and each job $i \in \mathcal{N}$ requires an amount q_i^r of resource $r \in \mathcal{R}$ per time

period of being processed. The availability of resource $r \in \mathcal{R}$ during time period $t \in \{1, \dots, T\}$ is limited by U_t^r.

A common integer programming formulation was first proposed in a paper of Pritsker et al. [44]. Define the binary decision variables

$$y_{i,t} = \begin{cases} 1 & \text{if job } i \text{ starts at time } t, \\ 0 & \text{otherwise,} \end{cases}$$

for all $i \in \mathcal{N}$ and $t \in \{1, \dots, T\}$.[5] This yields a start-time-dependent objective function. Pritsker et al. formulated the following integer programme *RCPSP*:

$$\text{maximise} \quad \sum_{i=1}^{N} \sum_{t=1}^{T} c_{i,t}\, y_{i,t} \tag{2.7}$$

$$\text{subject to}$$

$$\sum_{t=1}^{T} y_{i,t} = 1 \qquad \text{for all } i \in \mathcal{N}, \tag{2.7a}$$

$$\sum_{t=1}^{T} t(y_{i,t} - y_{j,t}) \geqslant d_{i,j} \qquad \text{for all } (i,j) \in \mathcal{E}, \tag{2.7b}$$

$$\sum_{i=1}^{N} \sum_{\substack{s=t-p_i+1 \\ s \geqslant 0}}^{t} q_i^r y_{i,s} \leqslant U_t^r \qquad \text{for all } r \in \mathcal{R},\ t \in \{1, \dots, T\}, \tag{2.7c}$$

$$y_{i,t} \in \{0,1\} \qquad \text{for all } i \in \mathcal{N},\ t \in \{1, \dots, T\}. \tag{2.7d}$$

Constraints (2.7a) demand that each job must be started exactly once during time periods $1, \dots, T$. (2.7b) models the precedence constraints between jobs. To ensure that all jobs are not only started but also completed at the latest in time period T, we can add an additional job to $\mathcal{N}$ with processing time 0 having all other jobs as predecessors. Finally, (2.7c) are the resource constraints for each time period and resource. For a more detailed explanation of the constraints, we refer to Pritsker et al. [44].

One apparent difference between resource-constrained project scheduling and the OPMPSP-models is that the latter requires all jobs to be scheduled, while we did not demand that all blocks need to be removed from the pit. Mathematically however, this is equivalent since we can introduce a "slack time period" $T+1$ to our OPMPSP-schedules with the corresponding objective coefficients all being zero. Then the blocks remaining in the pit can be "mined" during time period $T+1$ without change in the objective function value.

This way, Fricke's binary integer programme (2.3) from Section 2.2.2 (on T time periods) can be written in the form of *RCPSP* (on $T+1$ time periods):

[5] Strictly speaking, they introduced decision variables equal to 1 if job i is completed in time period t, 0 otherwise. However, the formulation with start times resembles more our OPMPSP-models and is essentially the same.

For the precedence graph define $\mathcal{E} = \{(i,j) \mid i \in \mathcal{N}, j \in \mathcal{P}(i)\}$ with zero time lags. Set the processing time p_i to 1 and the objective coefficients $c_{i,T+1}$ to 0 for each block $i \in \mathcal{N}$. Then the capacity U^r_{T+1} on attribute $r \in \mathcal{R}$ can be set to infinity (or any value greater than or equal to $\sum_{i=1}^{N} q^r_i$) to effectively remove the resource constraints for the slack time period $T+1$. This yields the programme

maximise

$$\sum_{i=1}^{N} \sum_{t=1}^{T} c_{i,t}\, y_{i,t}$$

subject to

$$\sum_{t=1}^{T+1} y_{i,t} = 1 \qquad \text{for all } i \in \mathcal{N},$$

$$\sum_{t=1}^{T+1} t\, y_{i,t} - \sum_{t=1}^{T+1} t\, y_{j,t} \geqslant 0 \qquad \text{for all } i \in \mathcal{N},\ j \in \mathcal{P}(i),$$

$$\sum_{i=1}^{N} q^r_i\, y_{i,t} \leqslant U^r_t \qquad \text{for all } r \in \mathcal{R},\ t \in \{1,\dots,T\},$$

$$y_{i,t} \in \{0,1\} \qquad \text{for all } i \in \mathcal{N},\ t \in \{1,\dots,T+1\}.$$

To see that this is equivalent to (2.3), it only remains to perform the substitution $y_{i,t} = x_{i,t} - x_{i,t-1}$ for all $i \in \mathcal{N}$ and $t \in \{1,\dots,T+1\}$. The nonnegativity of the y-variables gives the reserve constraints (2.3a). The resource constraints transform directly to (2.3c). Because of the integrality of the y-variables, the term $\sum_{t=1}^{T+1} t\, y_{i,t}$ evaluates exactly to the time period in which some block $i \in \mathcal{N}$ is scheduled. Thus it can be checked that for any $i \in \mathcal{N}$ and $j \in \mathcal{P}(i)$, the condition $\sum_{t=1}^{T+1} t\, y_{i,t} \geqslant \sum_{t=1}^{T+1} t\, y_{j,t}$ is equivalent to the precedence constraints (2.3b), $x_{i,t} \leqslant x_{j,t}$ for all time periods $t \in \{1,\dots,T\}$. The constraint $\sum_{t=1}^{T+1} y_{i,t} = 1$ translates into $x_{i,T+1} = 1$, and finally we can remove all variables $x_{i,T+1}$ together with the trivial constraints $x_{i,T} \leqslant x_{i,T+1}$ for all blocks $i \in \mathcal{N}$.

We will be able to exploit this insight later in Chapter 3 when inspecting the lagrangean relaxation of the resource constraints in *D-MIP*. Resource-constrained project scheduling itself is NP-hard, since it contains Fricke's OPMPSP-formulation (2.3), which in turn contains the precedence-constrained knapsack problem. However, without resource constraints, project scheduling problems can be solved in polynomial time. Möhring et al. [41] show how this is done efficiently by computing a minimum s-t-cut in a suitably constructed network. We will see that the same approach can be used to solve the subproblem remaining after relaxing the resource constraints of *D-MIP*.

2.5 Complexity analysis

We conclude Chapter 2 by demonstrating the computational difficulty posed also by the new OPMPSP-models:

Proposition 2.1 *The optimisation problems given by D-MIP and D-MIP' are NP-hard.*

Proof. We show that the precedence-constrained knapsack problem can be reduced to *D-MIP'* and to *D-MIP*. Since the precedence-constrained knapsack problem is NP-hard (see e.g. Kellerer et al. [32]), the other problems are as well.

PCKP $\rightarrow$ *D-MIP'*: To polynomially transform an instance of *PCKP* into an equivalent instance of *D-MIP'*, we need to map the items $1, \ldots, N$ in *PCKP* onto aggregates in *D-MIP'* and "forget" about blocks and z-variables in *D-MIP'*. In *D-MIP'*, let $T = 1$ and $K = N$; furthermore, set $\bar{c}_{k,1} = -c_k$, $p_{i,1} = 0$, $\bar{a}_k = w_k$ for all $k, i \in \{1, \ldots, N\}$, $U_1^{\mathrm{m}} = U$ and $U_1^{\mathrm{p}} = 0$. Because the z-variables have zero objective coefficients, they can be disregarded together with the processing constraint and what effectively remains is the original precedence-constrained knapsack problem.

PCKP $\rightarrow$ *D-MIP*: This reduction is slightly more complicated than the above ones. Let an instance of *PCKP* on N items be given as in Section 2.4.1. For each item $i \in \{1, \ldots, N\}$, define two aggregates $\mathcal{K}_i$ and $\mathcal{K}_{N+i}$ for *D-MIP* and let $T = 1$. The precedence relations on these aggregates are defined by $\bar{\mathcal{P}}(i) = \{N + i\}$ and $\bar{\mathcal{P}}(N + i) = \mathcal{P}(i)$ for all $i \in \{1, \ldots, N\}$, where $\mathcal{P}(i)$ is the predecessor set for item i in the *PCKP*-instance.

In the precedence graph (with arcs pointing from items to their predecessors), this would correspond to applying a node splitting technique: Take each node v in the precedence graph of *PCKP* and split it into two nodes v^+ and v^-. For any arc (u, w) in the original graph, draw an arc from u^+ to w^-, and add further arcs from v^- to v^+ for each node v of the original graph. Here, the "$-$"-nodes correspond to aggregate numbers $1, \ldots, N$ and the "$+$"-nodes correspond to aggregate numbers $N+1, \ldots, 2N$. The idea is now to define the knapsack constraint on the "$+$"-aggregates and assign the profits to the "$-$"-aggregates.

The z-variables and processing constraints can be removed as described above. In the objective function, set $\bar{c}_{i,1} = -c_i$ and $\bar{c}_{N+i,1} = 0$ for all $i \in \{1, \ldots, N\}$. In the mining constraints, set $\bar{a}_i = 0$ and $\bar{a}_{N+i} = w_i$ for

$i \in \{1, \dots, N\}$, $U_1^{\mathrm{m}} = U$. Then the corresponding *D-MIP*-instance reads

maximise

$$\sum_{i=1}^{N} c_i \, y_{i,1} \tag{2.8}$$

subject to

$$y_{k,1} - x_{k,1} \leqslant 0 \qquad \text{for all } k \in \{1, \dots, 2N\}, \tag{2.8a}$$

$$x_{k,1} - y_{\ell,1} \leqslant 0 \qquad \text{for all } k \in \{1, \dots, 2N\},\ \ell \in \bar{\mathcal{P}}(k), \tag{2.8b}$$

$$\sum_{i=1}^{N} w_i \, y_{N+i,1} \leqslant U \tag{2.8c}$$

$$x_{k,1} \in \{0, 1\} \qquad \text{for all } k \in \{1, \dots, 2N\}, \tag{2.8d}$$

$$0 \leqslant y_{k,1} \leqslant 1 \qquad \text{for all } k \in \{1, \dots, 2N\}. \tag{2.8e}$$

It remains to prove that this programme and the original instance of *PCKP* are equivalent. First we show that without changing the objective value we can choose the y-variables to be integral. The constraints (2.8b) can be split up into

$$x_{i,1} \leqslant y_{N+i,1} \quad \text{and} \quad x_{N+i,1} \leqslant y_{j,1}$$

for all $i \in \{1, \dots, N\}$ and $j \in \mathcal{P}(i)$. If in any feasible solution we decrease the value of the variables $y_{N+i,1}$ to $x_{i,1}$, all the constraints remain satisfied and the objective value does not change. Thus, we can assume that $y_{N+i,1} = x_{i,1}$ is integral for all $i \in \{1, \dots, N\}$.

Furthermore, the variables $y_{i,1}$, $i \in \{1, \dots, N\}$, are only bounded by integral values in constraints (2.8a) and (2.8b). Hence, if in a feasible solution one of these variables is fractional, both rounding up and down will preserve feasibility. Therefore, in any optimal solution, the variables $y_{i,t}$, $i \in \{1, \dots, N\}$, must be integral if c_i is non-zero; otherwise they could be rounded up or down to increase the objective value.

Then we may assume (without changing the objective value) that constraints (2.8a) hold with equality and $y_{N+i,1} = x_{i,1} = y_{i,1}$ for all $i \in \{1, \dots, N\}$. Substituting all the redundant variables, (2.8) becomes

$$\begin{aligned}
&\text{maximise} && \sum_{i=1}^{N} c_i \, y_{i,1} && \\
&\text{subject to} && y_{i,1} - y_{j,1} \leqslant 0 && \text{for all } i \in \{1, \dots, N\},\ j \in \mathcal{P}(i), \\
& && \sum_{i=1}^{N} w_i \, y_{i,1} \leqslant U, && \\
& && y_{i,1} \in \{0, 1\} && \text{for all } i \in \{1, \dots, N\}.
\end{aligned}$$

which is exactly the original *PCKP*. □

2.6 Conclusion

With Section 2.1, this chapter gave a precise description of the open pit mining production scheduling problem as understood in this thesis. In Section 2.2, we mentioned very briefly the main solution approaches taken in the literature – heuristics, dynamic programming and integer programming – to conclude that integer programming currently seems a most promising area of research for the OPMPSP. We presented classical integer and mixed-integer programming formulations from the literature, all of which required cutoff grades to be determined a priori, i.e. previous and hence not subject to the actual optimisation itself. We continued to describe a formulation overcoming this disadvantage, allowing for *integrated cutoff grade optimisation*. This model was first described and studied by Boland et al. [6], and will be the main subject of the following chapters. We addressed several points of criticism which could be directed towards this model and the integer programming approach in general.

Section 2.4 highlighted the important connection to two closely related problems, the precedence-constrained knapsack problem and the resource-constrained project scheduling problem. In the next chapter it will be shown in Section 3.4.3, how a result of Möhring et al. [41] for the latter can be applied to the OPMPSP-formulations under consideration in a straightforward way. To conclude, we used the connection to the precedence-constrained knapsack problem to prove that open pit mining production scheduling is NP-hard, not only for the standard integer programming formulations, but equally for the mixed-integer programming formulations considered here.

Chapter 3

Structural analysis

This chapter analyses structural properties of the OPMPSP-formulation *D-MIP* introduced in Section 2.3. The following chapters will largely be based on the insights presented here. In Section 3.1 we show how the corresponding LP-relaxation can be reduced in size to be solved faster by any of the known LP-algorithms. Section 3.2 highlights the structure given by the knapsack constraints together with its implications for optimal cutoff grades, Section 3.3 proves that the feasible region remaining when the knapsack constraints are removed is an integral polyhedron. This result also serves as a motivation to apply lagrangean relaxation to the resource constraints, which is the focus of Section 3.4. In Section 3.5, we briefly outline valid inequalities arising from the precedence-constrained knapsack problem structure and show how they can be added to the reduced LP-relaxation from Section 3.1 and the lagrangean relaxation from Section 3.4.

3.1 Redundancy in the LP-relaxation

A common method for solving mixed-integer programmes is a branch-and-bound approach. Within this, computation of dual bounds, for instance given by the LP-relaxation, is an essential part. Another motivation for looking closer at the LP-relaxation are results in the next section which interpret the dual multipliers associated with the processing constraints in relation to cutoff grades.

Removing the integrality condition on the x-variables of programme *D-MIP* from Section 2.3, i.e. replacing (2.5g) by

$$0 \leqslant x_{k,t} \leqslant 1 \quad \text{for all } k \in \{1, \dots, K\},\, t \in \{1, \dots, T\},$$

gives the standard LP-relaxation, which we will further refer to as *D-LP*.

Note that the x-variables have zero coefficients in the objective function of *D-LP*. Furthermore, if in any optimal solution we decrease their values to the lower bounds given by constraints (2.5b), the other constraints (2.5a) and

(2.5c) involving x-variables will also remain satisfied. Hence, without loss in the objective function value, we may assume that $x_{k,t} = \sum_{s=1}^{t} y_{k,s}$ holds for all $k \in \{1, \dots, K\}$ and $t \in \{1, \dots, T\}$. Substituting the x-variables accordingly yields the linear programme *y-D-LP*:

maximise

$$\sum_{k=1}^{K} \sum_{t=1}^{T} -\bar{c}_{k,t}\, y_{k,t} \;+\; \sum_{i=1}^{N} \sum_{t=1}^{T} p_{i,t}\, z_{i,t} \tag{3.1}$$

subject to

$$\sum_{t=1}^{T} y_{k,t} \leqslant 1 \qquad \text{for all } k \in \{1, \dots, K\}, \tag{3.1a}$$

$$\sum_{s=1}^{t} y_{k,s} - \sum_{s=1}^{t} y_{\ell,s} \leqslant 0 \qquad \text{for all } k \in \{1, \dots, K\},\ \ell \in \bar{\mathcal{P}}(k),\ t \in \{1, \dots, T\}, \tag{3.1b}$$

$$z_{i,t} - y_{k,t} \leqslant 0 \qquad \text{for all } k \in \{1, \dots, K\},\ i \in \mathcal{K}_k,\ t \in \{1, \dots, T\}, \tag{3.1c}$$

$$\sum_{k=1}^{K} \bar{a}_k y_{k,t} \leqslant U_t^{\mathrm{m}} \qquad \text{for all } t \in \{1, \dots, T\}, \tag{3.1d}$$

$$\sum_{i=1}^{N} a_i z_{i,t} \leqslant U_t^{\mathrm{p}} \qquad \text{for all } t \in \{1, \dots, T\}, \tag{3.1e}$$

$$0 \leqslant y_{k,t} \leqslant 1 \qquad \text{for all } k \in \{1, \dots, K\},\ t \in \{1, \dots, T\}, \tag{3.1f}$$

$$0 \leqslant z_{i,t} \leqslant 1 \qquad \text{for all } i \in \mathcal{N},\ t \in \{1, \dots, T\}. \tag{3.1g}$$

Alternatively, we can substitute the y-variables according to $y_{k,t} = x_{k,t} - x_{k,t-1}$ for all $k \in \{1, \dots, K\}$, $t \in \{1, \dots, T\}$, where $x_{k,0} = 0$ are auxiliary "variables" used for simplicity of notation. The resulting linear programme *x-D-LP* reads

maximise

$$\sum_{k=1}^{K} \sum_{t=1}^{T} -\bar{c}_{k,t}(x_{k,t} - x_{k,t-1}) \;+\; \sum_{i=1}^{N} \sum_{t=1}^{T} p_{i,t}\, z_{i,t} \tag{3.2}$$

subject to

$$x_{k,t-1} - x_{k,t} \leqslant 0 \qquad \text{for all } k \in \{1, \dots, K\},\ t \in \{2, \dots, T\}, \tag{3.2a}$$

$$x_{k,t} - x_{\ell,t} \leqslant 0 \qquad \text{for all } k \in \{1, \dots, K\},\ \ell \in \bar{\mathcal{P}}(k),\ t \in \{1, \dots, T\}, \tag{3.2b}$$

$$z_{i,t} - (x_{k,t} - x_{k,t-1}) \leqslant 0 \qquad \text{for all } k \in \{1, \dots, K\},\ i \in \mathcal{K}_k,\ t \in \{1, \dots, T\}, \tag{3.2c}$$

$$\sum_{k=1}^{K} \bar{a}_k(x_{k,t} - x_{k,t-1}) \leqslant U_t^{\mathrm{m}} \text{ for all } t \in \{1,\dots,T\}, \tag{3.2d}$$

$$\sum_{i=1}^{N} a_i z_{i,t} \leqslant U_t^{\mathrm{p}} \text{ for all } t \in \{1,\dots,T\}, \tag{3.2e}$$

$$x_{k,0} = 0 \quad \text{for all } k \in \{1,\dots,K\}, \tag{3.2f}$$

$$0 \leqslant x_{k,t} \leqslant 1 \quad \text{for all } k \in \{1,\dots,K\},\ t \in \{1,\dots,T\}, \tag{3.2g}$$

$$0 \leqslant z_{i,t} \leqslant 1 \quad \text{for all } i \in \mathcal{N},\ t \in \{1,\dots,T\}. \tag{3.2h}$$

Let the objective functions of *D-LP*, *x-D-LP* and *y-D-LP* be denoted by $f_{D\text{-}LP}$, $f_{x\text{-}D\text{-}LP}$ and $f_{y\text{-}D\text{-}LP}$, and the corresponding optimal objective values by $f^*_{D\text{-}LP}$, $f^*_{x\text{-}D\text{-}LP}$ and $f^*_{y\text{-}D\text{-}LP}$, respectively. The following proposition summarises the connection between the three programmes:

Proposition 3.1 *The linear programmes D-LP, x-D-LP and y-D-LP are equivalent in the following sense:*

(i) $f^*_{D\text{-}LP} = f^*_{x\text{-}D\text{-}LP} = f^*_{y\text{-}D\text{-}LP}$.

(ii) If $(x,y,z) \in \mathbb{R}^{K\times T} \times \mathbb{R}^{K\times T} \times \mathbb{R}^{N\times T}$ *is feasible for D-LP, then* (y,z) *is feasible for y-D-LP and* $(\tilde{x},z)$ *is feasible for x-D-LP with* $\tilde{x} \in \mathbb{R}^{K\times T}$ *given by*

$$\tilde{x}_{k,t} = \sum_{s=1}^{t} y_{k,s}$$

for $k \in \{1,\dots,K\}$, $t \in \{1,\dots,T\}$. *The respective objective function values are equal, i.e.* $f_{D\text{-}LP}(x,y,z) = f_{y\text{-}D\text{-}LP}(y,z) = f_{x\text{-}D\text{-}LP}(\tilde{x},z)$.

(iii) If $(x,z) \in \mathbb{R}^{K\times T} \times \mathbb{R}^{N\times T}$ *is feasible for x-D-LP, then* (x,y,z) *is feasible for D-LP with* $y \in \mathbb{R}^{K\times T}$ *given by*

$$y_{k,t} = \begin{cases} x_{k,t} - x_{k,t-1} & \text{for } t = 2,\dots,T, \\ x_{k,1} & \text{for } t = 1. \end{cases}$$

for $k \in \{1,\dots,K\}$, $t \in \{1,\dots,T\}$. *The respective objective function values are equal, i.e.* $f_{D\text{-}LP}(x,y,z) = f_{x\text{-}D\text{-}LP}(x,z)$.

(iv) If $(y,z) \in \mathbb{R}^{K\times T} \times \mathbb{R}^{N\times T}$ *is feasible for y-D-LP, then* (x,y,z) *is feasible for D-LP with* $x \in \mathbb{R}^{K\times T}$ *given by*

$$x_{k,t} = \sum_{s=1}^{t} y_{k,t}$$

for $k \in \{1,\dots,K\}$, $t \in \{1,\dots,T\}$. *The respective objective function values are equal, i.e.* $f_{D\text{-}LP}(x,y,z) = f_{y\text{-}D\text{-}LP}(y,z)$.

Proof. As explained above, without changing the objective function value we may assume that constraint (2.5b), i.e.

$$x_{k,t} \leqslant \sum_{s=1}^{t} y_{k,s} \quad \text{for all } k \in \{1,\dots,K\},\, t \in \{1,\dots,T\},$$

holds with equality in *D-LP*. □

In Section 2.3, beside *D-MIP*, we also introduced a variant *D-MIP'* which required every aggregate to be mined completely during one time period. Its LP-relaxation *D-LP'* is identical to *x-D-LP*, which gives the following corollary:

Corollary 3.2 *D-MIP is a relaxation of D-MIP' and the dual bounds given by the LP-relaxation of both programmes are identical, i.e.*

$$f^*_{\text{D-MIP'}} \leqslant f^*_{\text{D-MIP}} \leqslant f^*_{\text{D-LP}} = f^*_{\text{D-LP'}},$$

where $f^*_{\text{D-MIP'}}$, $f^*_{\text{D-MIP}}$, $f^*_{\text{D-LP}}$ *and* $f^*_{\text{D-LP'}}$ *denote the optimal objective values of the corresponding programmes.*

Proof. The first inequality holds since *D-MIP'* can be obtained from *D-MIP* by adding the constraints $x_{k,t} = \sum_{s=1}^{t} y_{k,s}$ for all $k \in \{1,\dots,K\}$ and $t \in \{1,\dots,T\}$. The last equality follows from $f^*_{\text{D-LP}} = f^*_{\text{x-D-LP}} = f^*_{\text{D-LP'}}$. □

This insight gives some reason to expect that, compared to *D-MIP'*, *D-MIP* will be computationally "easier" to solve (optimally or near-optimally) by an LP-based branch-and-bound algorithm, since the initial dual bound at the root node is smaller in general. Other aspects, however, might speak against it, especially the KT additional y-variables present in *D-MIP*. This will for instance have an effect on the running time of primal heuristics, and solving the LP-relaxations at each node, especially at the root node, will take longer in general. Theoretically, solving the LP-relaxation of *D-MIP* requires the same effort as for *D-MIP'*, since we may use the linear programme *x-D-LP* identical to *D-LP'*. However, when using an out-of-the-box MIP-solver one needs to be careful, because this redundancy might not be removed completely.

Remark 3.3 Reducing *D-LP* to *x-D-LP* or *y-D-LP* is basically a presolving technique. Later on, we will be interested in the values of optimal dual multipliers associated with the resource constraints. In general, presolving steps can affect the dual space. Here it is straightforward to check that the dual multipliers associated with the resource constraints are optimal for *D-LP* if and only if they are optimal for *x-D-LP* if and only if they are optimal for *y-D-LP*.

3.2 Knapsack structures

In this section we analyse optimal solutions of *D-MIP*, more precisely their z-components. Because the following discussion is independent of the integrality conditions in *D-MIP*, it is equally valid for its LP-relaxation *D-LP*. The results hold analogously for the case of *D-MIP'*, but for simplicity of the exposition, we only consider *D-MIP* and *D-LP*.

Let an optimal solution (x^*, y^*, z^*) of either *D-MIP* or *D-LP* be fixed. The z-variables are subject only to the constraints (2.5d), (2.5f) and (2.5i). Because of optimality, for any fixed time period $t \in \{1, \dots, T\}$, $\left(z^*_{i,t}\right)_{i \in \mathcal{N}} \in \mathbb{R}^N$ must form an optimal solution to the linear programme

$$\text{maximise} \quad \sum_{i=1}^{N} p_{i,t} z_{i,t} \tag{3.3}$$

$$\text{subject to} \quad \sum_{i=1}^{N} a_i z_{i,t} \leqslant U^{\mathrm{p}}_t, \tag{3.3a}$$

$$0 \leqslant z_{i,t} \leqslant y^*_{k,t} \qquad \text{for all } k \in \{1, \dots, K\},\ i \in \mathcal{K}_k. \tag{3.3b}$$

Problems of this form, i.e. with continuous variables ranging from 0 to an upper bound and one knapsack constraint are called *continuous bounded knapsack problems*. They can be viewed as the LP-relaxation of a *bounded knapsack problem*, which differs from the standard 0-1 knapsack problem (see Section 2.4.1) only because the variables are not binary, but take integer values ranging from zero to an upper bound.

An optimal solution of a continuous bounded knapsack problem can be determined by ordering items according to non-increasing cost per unit weight. More precisely, we can prove the following result:

Proposition 3.4 *Let $(x^*, y^*, z^*) \in \mathbb{R}^{K \times T} \times \mathbb{R}^{K \times T} \times \mathbb{R}^{N \times T}$ be an optimal solution of D-MIP or D-LP. Then there exists a sequence $\sigma = (\sigma_1, \dots, \sigma_T)$ of non-negative values such that*

$$z^*_{i,t} = \begin{cases} 0 & \text{if } \frac{p_{i,t}}{a_i} < \sigma_t, \\ y^*_{k,t} & \text{if } \frac{p_{i,t}}{a_i} > \sigma_t, \end{cases} \tag{3.4}$$

holds for all $i \in \mathcal{N}$ and $t \in \{1, \dots, T\}$.

Proof. As mentioned above, $\left(z^*_{i,t}\right)_{i \in \mathcal{N}} \in \mathbb{R}^N$ must be an optimal solution of (3.3) for each fixed time period $t \in \{1, \dots, T\}$. Since the programmes (3.3) are not related for different values of t, each time period can be treated separately. Therefore, it suffices to show that for any continuous bounded

knapsack problem

$$\begin{aligned} \text{maximise} \quad & \sum_{i=1}^{N} c_i x_i \\ \text{subject to} \quad & \sum_{i=1}^{N} w_i x_i \leqslant U, \\ & 0 \leqslant x_i \leqslant b_i \quad \text{for all } i \in \{1, \ldots, N\}, \end{aligned} \tag{CBKP}$$

where $c_i \in \mathbb{R}$, $w_i > 0$, $b_i \geqslant 0$ for $i \in \{1, \ldots, N\}$, there exists a value $\rho \geqslant 0$ such that

$$x_i = \begin{cases} 0 & \text{if } \frac{c_i}{w_i} < \rho, \\ b_i & \text{if } \frac{c_i}{w_i} > \rho, \end{cases}$$

holds for all $i \in \{1, \ldots, N\}$ in any optimal solution x.

Let the items be ordered by non-increasing profit per unit weight, i.e.

$$\frac{c_1}{w_1} \geqslant \frac{c_2}{w_2} \geqslant \ldots \geqslant \frac{c_N}{w_N}.$$

Without loss of generality we may further assume that profits are non-negative. It is always suboptimal to include items with negative profit, thus $x_i = 0$ holds in any optimal solution if $\frac{c_i}{w_i} < 0 \leqslant \rho$.

Now, if all items can be fully included in the knapsack, i.e. if $\sum_{i=1}^{N} w_i b_i \leqslant U$, then any optimal solution will fully contain all items of positive weight and we can choose $\rho = 0$. Therefore we may assume without loss of generality that $\sum_{i=1}^{N} w_i b_i > U$. In this case, there exists exactly one item $s \in \{1, \ldots, N\}$, called *split item*, with

$$\sum_{i=1}^{s-1} w_i b_i \leqslant U \quad \text{and} \quad \sum_{i=1}^{s} w_i b_i > U.$$

Now, an optimal solution of (CBKP) is given by

$$\begin{aligned} x_i^* &= b_i \quad \text{for all } i \in \{1, \ldots, s-1\}, \\ x_s^* &= \frac{1}{w_s} \left(U - \sum_{i=1}^{s-1} w_i b_i \right) \quad \text{and} \\ x_i^* &= 0 \quad \text{for all } i \in \{s+1, \ldots, N\}. \end{aligned}$$

A proof for unit bounds is given by Kellerer et al. [32, pp. 18] and holds analogously for the general case. From the proof it also follows that

$$x_i = \begin{cases} b_i & \text{if } \frac{c_i}{w_i} > \frac{c_s}{w_s}, \\ 0 & \text{if } \frac{c_i}{w_i} < \frac{c_s}{w_s}, \end{cases}$$

holds for all $i \in \{1, \ldots, N\}$ in any optimal solution x. Hence, the choice $\rho = \frac{c_s}{w_s}$ has the desired properties. □

Definition 3.5 *We call* $\sigma = (\sigma_1, \ldots, \sigma_T)$ *a sequence of* split ratios *for D-MIP or D-LP, if condition* (3.4) *is satisfied for some optimal solution* (x^*, y^*, z^*).

Corollary 3.6 *A sequence of split ratios exists for any instance of D-MIP and D-LP.*

Proof. For any instance of *D-MIP* or *D-LP*, the feasible region is not empty (the zero-solution is always feasible) and bounded. Hence, an optimal solution exists and Proposition 3.4 guarantees a sequence of split ratios. □

The following proposition explains how to compute split ratios in theory, given an optimal solution (x^*, y^*, z^*) of *D-MIP* or *D-LP*:

Proposition 3.7 *Let* $(x^*, y^*, z^*) \in \mathbb{R}^{K\times T} \times \mathbb{R}^{K\times T} \times \mathbb{R}^{N\times T}$ *be an optimal solution of D-MIP or D-LP. Split ratios* σ_t, $t \in \{1, \ldots, T\}$, *are given by any values* σ_t *in the interval* $[\underline{\sigma}_t, \overline{\sigma}_t]$, *where*

$$\underline{\sigma}_t = \max\left\{0, \max\left\{ \frac{p_{i,t}}{a_i} \;\middle|\; z^*_{i,t} < y^*_{k,t}, k \in \mathcal{K}, i \in \mathcal{K}_k \right\}\right\} \tag{3.5}$$

and

$$\overline{\sigma}_t = \min\left\{ \frac{p_{i,t}}{a_i} \;\middle|\; z^*_{i,t} > 0, i \in \mathcal{N} \right\}. \tag{3.6}$$

If (x^*, y^*, z^*) *is a unique solution to D-MIP or D-LP, respectively, then any sequence of split ratios is of this form.*

Proof. Let ρ be a sequence of split ratios as guaranteed by Proposition 3.4, and let $t \in \{1, \ldots, T\}$ be a fixed time period. It follows from (3.4) that for all $k \in \{1, \ldots, K\}$ and $i \in \mathcal{K}_k$,

$$\frac{p_{i,t}}{a_i} \begin{cases} \geqslant \rho_t & \text{if } z^*_{i,t} > 0, \\ \leqslant \rho_t & \text{if } z^*_{i,t} < y^*_{k,t}. \end{cases}$$

Hence, $0 \leqslant \underline{\sigma}_t \leqslant \rho_t \leqslant \overline{\sigma}_t$ and so the interval $[\underline{\sigma}_t, \overline{\sigma}_t]$ is not empty. By definition of $\underline{\sigma}_t$ and $\overline{\sigma}_t$, any choice of $\sigma_t \in [\underline{\sigma}_t, \overline{\sigma}_t]$ will satisfy (3.4) for (x^*, y^*, z^*).

Suppose (x^*, y^*, z^*) is the only optimal solution to *D-MIP*, then any sequence of split ratios σ must satisfy (3.4) for (x^*, y^*, z^*). It follows immediately that $\sigma_t \in [\underline{\sigma}_t, \overline{\sigma}_t]$ for all $t \in \{1, \ldots, T\}$, again by definition of $\underline{\sigma}_t$ and $\overline{\sigma}_t$. □

Remark 3.8 Proposition 3.7 also shows that in general split ratios are not uniquely determined. In some time period $t \in \{1, \ldots, T\}$, though, for which $0 < z^*_{i,t} < y^*_{k,t}$ holds for some $k \in \{1, \ldots, K\}$, $i \in \mathcal{K}_k$, the value $\frac{p_{i,t}}{a_i}$ is contained in the domain of both (3.5) and (3.6) and the interval $[\underline{\sigma}_t, \overline{\sigma}_t]$ shrinks to one point.

Note that Proposition 3.7 is only a theoretical result, since it already requires an optimal solution of *D-MIP*. The computational benefit from knowledge about split ratios is apparent, though. If we knew a sequence σ of split ratios for *D-MIP*, we could reduce the programme significantly in size: For all $i \in \mathcal{N}$, $t \in \{1, \dots, T\}$ with $\frac{p_{i,t}}{a_i} < \sigma_t$, replace $z_{i,t}$ by 0; if $\frac{p_{i,t}}{a_i} > \sigma_t$ then substitute $z_{i,t} = y_{k,t}$. However, the following proposition shows that computing split ratios is NP-hard. Therefore, Chapter 5 will pursue a slightly different approach using heuristic ideas.

Proposition 3.9 *Computing split ratios for D-MIP is NP-hard.*

Proof. The notion of split ratios as introduced in this section is equally valid for *D-MIP'* and all the results from above hold correspondingly. For clarity of exposition, we will first give the proof that computing split ratios for *D-MIP'* is NP-hard. For this we show how the precedence-constrained knapsack problem can be solved using split ratios to indicate whether some item is included in an optimal solution.

Let an instance of *PCKP* be given as in Section 2.4.2, with N items of positive integer weight and objective coefficient, say. Choose an arbitrary item $j \in \{1, \dots, N\}$, for which we want to determine whether it is included in some optimal solution. Construct an instance of *D-MIP'* with one time period, $T = 1$, and $K = N+1$ aggregates as follows.

Aggregates $\mathcal{K}_1, \dots, \mathcal{K}_N$ correspond to the knapsack items: Set $\bar{a}_k = w_k$, $\bar{\mathcal{P}}(k) = \mathcal{P}(k)$ and $\bar{c}_{k,1} = -c_k$ for all $k \in \{1, \dots, N\}$. Aggregate $\mathcal{K}_{N+1}$ has an auxiliary function and will be "mined" in any optimal solution due to its large objective coefficient and no predecessors: Set $\bar{a}_{N+1} = w_j$, $\bar{\mathcal{P}}(N+1) = \emptyset$ and $\bar{c}_{N+1,1} = -C^0$, where $C^0 = \sum_{i=1}^{N} c_i + 1$. All aggregates are singletons, say $\mathcal{K}_k = \{k\}$ and $a_k = \bar{a}_k$ for all $k \in \{1, \dots, N+1\}$. The objective coefficients of the z-variables are

$$p_{k,1} = \begin{cases} \frac{1}{2} & \text{for } k = j, \\ \frac{1}{3} & \text{for } k = N+1, \\ 0 & \text{otherwise.} \end{cases}$$

Finally, set the mining and processing capacities by $U_1^{\mathrm{m}} = U + w_j$ and $U_1^{\mathrm{p}} = \frac{w_j}{2}$.

All in all, this gives the *D-MIP'*-instance

maximise

$$\sum_{k=1}^{N} c_k x_{k,1} + C^0 x_{N+1,1} + \frac{1}{2} z_{j,1} + \frac{1}{3} z_{N+1,1} \tag{3.7}$$

subject to

$$x_{k,1} - x_{\ell,1} \leqslant 0 \qquad \text{for all } k \in \{1,\dots,N\},\ \ell \in \mathcal{P}(k), \tag{3.7a}$$

$$z_{k,1} - x_{k,1} \leqslant 0 \qquad \text{for all } k \in \{1,\dots,N+1\}, \tag{3.7b}$$

$$\sum_{k=1}^{N} w_k x_{k,1} + w_j x_{N+1,1} \leqslant U + w_j, \tag{3.7c}$$

$$\sum_{k=1}^{N} w_k z_{k,1} + w_j z_{N+1,1} \leqslant \frac{w_j}{2}, \tag{3.7d}$$

$$x_{k,1} \in \{0,1\} \quad \text{for all } k \in \{1,\dots,N+1\}, \tag{3.7e}$$

$$0 \leqslant z_{k,1} \leqslant 1 \qquad \text{for all } k \in \{1,\dots,N+1\}. \tag{3.7f}$$

Step 1: We show that if (x, z) is an optimal solution of *D-MIP'*, then the set of knapsack items $I = \{k \in \{1,\dots,N\} \mid x_{k,1} = 1\}$ gives an optimal solution of *PCKP*.

Let (x, z) be optimal for *D-MIP'*, then the items in I are precedence-feasible because of (3.7a). Since $x_{N+1,1}$ must be 1 for optimality in *D-MIP'*, the knapsack constraint in *PCKP* is satisfied due to (3.7c). Constraint (3.7d) ensures that $z_{j,1} + z_{N+1,1} \leqslant \frac{1}{2}$. Hence, if $f_{\textit{D-MIP'}}(x, z)$ is the objective value of (x, z) and $c(I)$ the objective value of the knapsack solution, then

$$f_{\textit{D-MIP'}}(x, z) - C^0 - \frac{1}{2} \leqslant c(I) \leqslant f_{\textit{D-MIP'}}(x, z) - C^0. \tag{3.8}$$

Now suppose that I is not an optimal knapsack, i.e. there is a feasible set of items $\tilde{I} \subseteq \{1,\dots,N\}$ with $c(\tilde{I}) > c(I)$. Because we assumed the objective coefficients to be integral, we know that even $c(\tilde{I}) \geqslant c(I) + 1$ holds. Defining

$$\tilde{x}_{k,1} = \begin{cases} 1 & \text{if } k \in \tilde{I} \cup \{N+1\}, \\ 0 & \text{otherwise,} \end{cases}$$

gives a feasible solution $(\tilde{x}, 0)$ of *D-MIP'*. But

$$\begin{aligned} f_{\textit{D-MIP'}}(\tilde{x}, 0) &= c(\tilde{I}) + C^0 \geqslant c(I) + 1 + C^0 \\ &> c(I) + C^0 + \frac{1}{2} \overset{(3.8)}{\geqslant} f_{\textit{D-MIP'}}(x, z), \end{aligned}$$

which would contradict the optimality of (x, z).

Step 2: Suppose an optimal split ratio σ_1 for *D-MIP'* is known, i.e. there is an optimal solution (x, z) of *D-MIP'* such that

$$z_{k,1} = \begin{cases} 0 & \text{if } \frac{p_{k,1}}{a_k} < \sigma_1, \\ x_{k,1} & \text{if } \frac{p_{k,1}}{a_k} > \sigma_1, \end{cases} \tag{3.9}$$

for all $k \in \{1, \dots, N+1\}$. Then the characterisation

$$\sigma_1 > \frac{1}{3a_j} \Leftrightarrow x_{j,1} = 1. \tag{3.10}$$

holds. For if $\sigma_1 \leqslant \frac{1}{3a_j}$, then $\frac{p_{j,1}}{a_j} = \frac{1}{2a_j} \geqslant \sigma_1$ and by (3.9), $z_{j,1} = x_{j,1}$ holds. But $z_{j,1} \leqslant \frac{1}{2}$ because of (3.7d), and so $x_{j,1} = 0$. Conversely, if $\sigma_1 > \frac{1}{3a_j}$ holds, then $z_{N+1,1} = 0$ by (3.9). Since (x, z) is optimal, $x_{N+1,1} = 1$ and (3.7d) must be tight. But then $z_{j,1} = \frac{1}{2}$, implying $x_{j,1} = 1$ by (3.7b).

All in all, this gives an algorithm for solving the following problem: Given an instance of *PCKP* (with positive weights and objective coefficients) and an item $j \in \{1, \dots, N\}$, decide whether there exists an optimal solution including item j, or there exists an optimal solution not including item j. (One of the alternatives must hold.)

First, construct the *D-MIP'*-instance (3.7) as above. Second, compute a split ratio σ_1. If $\sigma_1 > \frac{1}{3a_j}$, then *D-MIP'* has an optimal solution with $x_{j,1} = 1$ according to (3.10), and Step 1 gives an optimal solution of *PCKP* including j. Otherwise, if $\sigma_1 \leqslant \frac{1}{3a_j}$, then *D-MIP'* has an optimal solution with $x_{j,1} = 0$, and Step 1 gives an optimal solution of *PCKP* not including j.

This test for containment can be turned into an algorithm for solving *PCKP* in a straightforward way: Start with an arbitrary item and test for its containment in an optimal solution. If it is contained in an optimal solution, then mark them as included in the knapsack with all its predecessors. This yields a new, smaller problem with those items removed and the upper bound decreased by the sum of the weights of the removed items. Otherwise, if the item is not included for some optimal solution, then only remove it with all its successors and mark the removed items as not included in the knapsack. Continue by choosing another item until none are left.

This procedure finishes with an optimal solution after at most as many steps as items. Hence, if split ratios can be computed efficiently, then *PCKP* can be solved efficiently, i.e. in polynomial time. Since the precedence-constrained knapsack problem is NP-hard (see e.g. Kellerer et al. [32]), even when restricted to positive integer weights and objective coefficients, this proves the computation of split ratios for *D-MIP'* to be NP-hard as well.

To give an analogous proof for the case of *D-MIP*, the node splitting technique from the proof of Proposition 2.1 can be used. The proof is essentially identical except for the additional technical details from node splitting. □

3.3 Integrality of the precedence polytope

In this section, we will inspect the "precedence polyhedron" given by all the constraints of *D-MIP* except for the resource constraints, i.e. the polyhedron

$$P = \left\{ (x,y,z) \;\middle|\; \begin{array}{l} x \in [0,1]^{K\times T}, y \in [0,1]^{K\times T}, z \in [0,1]^{N\times T} \\ \text{with (2.5a), (2.5b), (2.5c) and (2.5d)} \end{array} \right\}.$$

We will prove that P is integral, i.e. each of the components of an extreme point of P is either 0 or 1. This holds essentially because of the close relation to unconstrained project scheduling. For the latter, Sankaran et al. [47] show this integrality property for a formulation with ordinary precedence constraints.

Proposition 3.10 *P is integral.*

Proof. Step 1: First of all, we prove the integrality for the polyhedron P_0 given by the inequalities

$$\left.\begin{array}{ll} x_{k,t-1} - x_{k,t} & \leqslant 0 \text{ for all } k \in \{1,\dots,K\},\, t \in \{2,\dots,T\}, \\ \displaystyle\sum_{s=1}^{t} y_{k,s} - x_{k,t} & \leqslant 0 \text{ for all } k \in \{1,\dots,K\},\, t \in \{1,\dots,T\}, \\ \displaystyle x_{k,t} - \sum_{s=1}^{t} y_{l,s} & \leqslant 0 \text{ for all } k \in \{1,\dots,K\},\, l \in \bar{\mathcal{P}}(k),\, t \in \{1,\dots,T\}, \\ 0 \leqslant \quad x_{k,t} & \leqslant 1 \text{ for all } k \in \{1,\dots,K\},\, t \in \{1,\dots,T\}, \\ 0 \leqslant \quad y_{k,t} & \leqslant 1 \text{ for all } k \in \{1,\dots,K\},\, t \in \{1,\dots,T\}. \end{array}\right\} \tag{3.11}$$

For all $k \in \{1,\dots,K\}$, perform the substitution given by $y_{k,1} = u_{k,1}$ and $y_{k,t} = u_{k,t} - u_{k,t-1}$ for $t \in \{2,\dots,T\}$. This transforms (3.11) equivalently into

$$\left.\begin{array}{ll} x_{k,t-1} - x_{k,t} & \leqslant 0 \text{ for all } k \in \{1,\dots,K\},\, t \in \{2,\dots,T\}, \\ u_{k,t-1} - u_{k,t} & \leqslant 0 \text{ for all } k \in \{1,\dots,K\},\, t \in \{2,\dots,T\}, \\ u_{k,t} - x_{k,t} & \leqslant 0 \text{ for all } k \in \{1,\dots,K\},\, t \in \{1,\dots,T\}, \\ x_{k,t} - u_{l,t} & \leqslant 0 \text{ for all } k \in \{1,\dots,K\},\, l \in \bar{\mathcal{P}}(k),\, t \in \{1,\dots,T\}, \\ 0 \leqslant \quad x_{k,t} & \leqslant 1 \text{ for all } k \in \{1,\dots,K\},\, t \in \{1,\dots,T\}, \\ 0 \leqslant \quad u_{k,t} & \leqslant 1 \text{ for all } k \in \{1,\dots,K\},\, t \in \{1,\dots,T\}. \end{array}\right\} \tag{3.12}$$

This transformation is linear, bijective and maps integer points to integer points. Hence, it induces a bijection between the extreme points of (3.11) and (3.12). Therefore, it suffices to show the integrality of the polyhedron given by (3.12).

This is due to total unimodularity: A matrix with integer entries is called *totally unimodular* if the determinant of each square submatrix is 0, 1 or -1. If an $m\times n$-matrix A is totally unimodular and $d \in \mathbb{R}^m$ is integral, then all

extreme points of the polyhedron $\{x \in \mathbb{R}^n \mid Ax \leqslant d\}$ are integral. Now the total unimodularity of the matrix given by (3.12) follows from a simple criterion, see for instance the book of Nemhauser and Wolsey [42, pp. 540]: Each row contains – besides zeros – either exactly one entry 1 or -1, or two entries 1 and -1. The right hand side is the zero vector and so the extreme points are integral.

Now, consider also the z-variables with the constraints

$$0 \leqslant z_{i,t} \leqslant y_{k,t} \quad \text{for all } k \in \{1,\dots,K\},\ i \in \mathcal{K}_k,\ t \in \{1,\dots,T\}. \tag{3.13}$$

Inequalities (3.11) and (3.13) together determine the polyhedron P as defined above. Let $(\hat{x},\hat{y},\hat{z})$ be an extreme point of P.

Step 2: We show that $(\hat{x},\hat{y})$ is an extreme point of P_0. Suppose

$$(\hat{x},\hat{y}) = \sum_{j=1}^{m} \alpha_j (x^j, y^j)$$

where $\sum_{j=1}^{m} \alpha_j (x^j, y^j)$ is a convex combination of some points $(x^j, y^j) \in P_0$. For each $k \in \{1,\dots,K\}$, $i \in \mathcal{K}_k$, $t \in \{1,\dots,T\}$ and $j \in \{1,\dots,m\}$, define

$$z^j_{i,t} = \begin{cases} 0 & \text{if } \hat{y}_{k,t} = 0, \\ \dfrac{\hat{z}_{i,t}}{\hat{y}_{k,t}}\, y^j_{k,t} & \text{otherwise.} \end{cases}$$

Since $0 \leqslant \dfrac{\hat{z}_{i,t}}{\hat{y}_{k,t}} \leqslant 1$, all (x^j, y^j, z^j), $j \in \{1,\dots,m\}$, satisfy (3.13) and are in P. By definition,

$$(\hat{x},\hat{y},\hat{z}) = \sum_{j=1}^{m} \alpha_j (x^j, y^j, z^j).$$

As $(\hat{x},\hat{y},\hat{z})$ is assumed to be an extreme point of P, it follows that $\alpha_j = 0$ holds for all $j \in \{1,\dots,m\}$ with $(x^j, y^j, z^j) \neq (\hat{x},\hat{y},\hat{z})$. Hence, for all $j \in \{1,\dots,m\}$

$$(x^j, y^j) \neq (\hat{x},\hat{y}) \Rightarrow \alpha_j = 0.$$

So the only possibilities to write $(\hat{x},\hat{y})$ as convex combination of other points in P_0 are trivial convex combinations, i.e. $(\hat{x},\hat{y})$ is an extreme point of P_0.

Step 3: Finally, we show that any extreme point $(\hat{x},\hat{y},\hat{z})$ of P is integral. Step 1 and 2 together yield that all $\hat{x}$- and $\hat{y}$-components are binary. For $k \in \{1,\dots,K\}$ and $t \in \{1,\dots,T\}$ with $\hat{y}_{k,t} = 0$, $\hat{z}_{i,t} = 0$ for all $i \in \mathcal{K}_k$ by (3.13). Suppose that $\hat{y}_{k^*,t^*} = 1$ and $0 < \hat{z}_{i^*,t^*} < 1$ holds for some $k^* \in \mathcal{K}$, $i^* \in \mathcal{K}_{k^*}$ and $t^* \in \{1,\dots,T\}$. Then we can form a non-trivial convex combination

$$(\hat{x},\hat{y},\hat{z}) = \hat{z}_{i^*,t^*}(\hat{x},\hat{y},z^1) + (1-\hat{z}_{i^*,t^*})(\hat{x},\hat{y},z^2),$$

where $z^1_{i^*,t^*} = 1$, $z^2_{i^*,t^*} = 0$ and $z^1_{i,t} = z^2_{i,t} = \hat{z}_{i,t}$ for all $(i,t) \in \mathcal{N} \times \{1,\dots,T\}$ different from (i^*,t^*). This is a contradiction to $(\hat{x},\hat{y},\hat{z})$ being an extreme point of P, thus all $\hat{z}$-variables are integral as well. □

The same result holds for the precedence polyhedron of *D-MIP'*, as can be shown by an analogous proof.

3.4 A lagrangean relaxation approach

3.4.1 Lagrangean relaxation in the literature

Fisher [17] gives a review of applications of the lagrangean approach in integer programming during the first decade since its initial application by Held and Karp [24]. As he points out, "[one] of the most computationally useful ideas of the 1970s is the observation that many hard problems can be viewed as easy problems complicated by a relatively small set of side constraints." The OPMPSP-formulations from Chapter 2 are prototypical examples for this: In Section 3.3 we showed that the precedence polytope, i.e. the feasible region remaining when the resource constraints are relaxed, are integral. Hence, optimisation over this polytope can be done in polynomial time – in this sense an "easy" problem. Adding the comparatively small number of $2T$ resource constraints, however, results in NP-hard optimisation problems, see Proposition 2.1.

This motivates the application of lagrangean relaxation to the resource constraints, hoping to obtain the same dual bounds faster than by the standard LP-relaxation. For the resource-contrained project scheduling problem (see Section 2.4.2), similar approaches have been developed and tested by Möhring et al. [41] and Kimms [33].

Particularly interesting in the OPMPSP-context is an application of lagrangean relaxation by Caccetta et al. [10] to an integer programming formulation of the OPMPSP. Although they use a slightly different model, the ideas and results are closely related. The resource constraints are dualised and the resulting problem is reduced in size. A subgradient algorithm is used to find optimal Lagrange multipliers. Schedules obtained from solving the lagrangean relaxation are precedence feasible, but violate the resource constraints. Thus a heuristic is proposed in order to reconstruct a fully feasible schedule. They also show how Lagrange multipliers may be understood as cutoff grades. Section 3.4.4 will give a similar interpretation for our OPMPSP-formulation *D-MIP*.

3.4.2 Lagrangean relaxation of the resource constraints

To formulate the lagrangean relaxation, we introduce non-negative dual Lagrange multipliers $\mu_1, \ldots, \mu_T$ for the mining constraints (2.5e) and $\pi_1, \ldots, \pi_T$ for the processing constraints (2.5f) in *D-MIP*. We drop the resource constraints from the list of constraints in *D-MIP* and add penal terms to the

objective function corresponding to their violation. For fixed Lagrange multipliers $\mu, \pi \in \mathbb{R}^T_{\geqslant 0}$, this gives the objective function

$$f_{D\text{-}LR(\mu,\pi)}(x,y,z) = \sum_{k=1}^{K}\sum_{t=1}^{T} -\bar{c}_{k,t}y_{k,t} + \sum_{i=1}^{N}\sum_{t=1}^{T} p_{i,t}z_{i,t} \tag{3.14}$$
$$+ \sum_{t=1}^{T}\mu_t\left(U_t^{\mathrm{m}} - \sum_{k=1}^{K}\bar{a}_k y_{k,t}\right) + \sum_{t=1}^{T}\pi_t\left(U_t^{\mathrm{p}} - \sum_{i=1}^{N} a_i z_{i,t}\right).$$

The resulting lagrangean relaxation $D\text{-}LR(\mu,\pi)$ reads

maximise

$$f_{D\text{-}LR(\mu,\pi)}(x,y,z) \tag{3.15}$$

subject to

$$x_{k,t-1} - x_{k,t} \leqslant 0 \qquad \text{for all } k \in \{1,\dots,K\},\ t \in \{2,\dots,T\}, \tag{3.15a}$$

$$\sum_{s=1}^{t} y_{k,s} - x_{k,t} \leqslant 0 \qquad \text{for all } k \in \{1,\dots,K\},\ t \in \{1,\dots,T\}, \tag{3.15b}$$

$$x_{k,t} - \sum_{s=1}^{t} y_{\ell,s} \leqslant 0 \qquad \text{for all } k \in \{1,\dots,K\},\ \ell \in \bar{\mathcal{P}}(k),\ t \in \{1,\dots,T\}, \tag{3.15c}$$

$$z_{i,t} - y_{k,t} \leqslant 0 \qquad \text{for all } k \in \{1,\dots,K\},\ i \in \mathcal{K}_k,\ t \in \{1,\dots,T\}, \tag{3.15d}$$

$$x_{k,t} \in \{0,1\} \qquad \text{for all } k \in \{1,\dots,K\},\ t \in \{1,\dots,T\}, \tag{3.15e}$$

$$0 \leqslant y_{k,t} \leqslant 1 \qquad \text{for all } k \in \{1,\dots,K\},\ t \in \{1,\dots,T\}, \tag{3.15f}$$

$$0 \leqslant z_{i,t} \leqslant 1 \qquad \text{for all } i \in \mathcal{N},\ t \in \{1,\dots,T\}. \tag{3.15g}$$

If we denote by $f^*_{D\text{-}LR(\mu,\pi)}$ the optimal objective value of $D\text{-}LR(\mu,\pi)$ for fixed Lagrange multipliers $\mu, \pi \in \mathbb{R}^T_{\geqslant 0}$, then the so-called dual function is given by

$$\phi : \mathbb{R}^T_{\geqslant 0} \times \mathbb{R}^T_{\geqslant 0} \longrightarrow \mathbb{R},\ (\mu,\pi) \longmapsto f^*_{D\text{-}LR(\mu,\pi)}.$$

The value of $\phi(\mu,\pi)$ gives an upper bound on $f^*_{D\text{-}MIP}$, the optimal objective value of *D-MIP*, for any choice of Lagrange multipliers $\mu, \pi \in \mathbb{R}^T_{\geqslant 0}$. The problem of computing the best, i.e. lowest, possible bound of that form,

$$\phi^* = \min_{\mu,\pi \in \mathbb{R}^T_{\geqslant 0}} \phi(\mu,\pi), \tag{D-LD}$$

is commonly referred to as lagrangean dual. It is a basic result from Integer Programming that ϕ^* is at least as low a bound as the bound from the LP-relaxation. However, because the so-called *Integrality Property* holds, i.e. because the feasible region of the lagrangean relaxation problem (3.15) is integral, we obtain the same bound as by LP-relaxation:

Proposition 3.11 $\phi^* = f^*_{D\text{-}LP}$.

Proof. The feasible region of the lagrangean relaxation given by constraints (3.15a) to (3.15g) is exactly the precedence polytope P, for which Proposition 3.10 guarantees integrality. From a result of Geoffrion [20] it follows that in this case the LD-bound equals the LP-bound. □

This result shows that from a computational perspective the integrality of the precedence polytope P is not only beneficial. If P was not integral, we could hope for the lagrangean relaxation approach to provide us with tighter bounds than the LP-relaxation – with P integral it is clear that lagrangean relaxation can do no better than the LP-relaxation, at least in terms of the dual bound obtained. The use of lagrangean relaxation, however, is not limited to the computation of dual bounds. Although solutions of the lagrangean relaxation are in general not completely feasible for the original problem, they can often be used as a basis for heuristics, an aspect that will be treated further in Chapter 6.

As Geoffrion [20] also points out, generally "it is more promising to use Lagrangean relaxations for which the Integrality Property does not hold." He adds, though, that "this negative conclusion rests on the implicit assumption that [the LP-relaxation] is of manageable size". Since realistic OPMPSP-instances are large-scale, this assumption may not hold. Geoffrion also mentions another case where the lagrangean approach may be valuable despite yielding at best the LP-bound – if "a near-optimal solution [of the lagrangean dual] can be found by specialised means more rapidly than [the LP-relaxation] can be solved by linear programming methods".

The following shall demonstrate exactly this: Programme (3.15) can be transformed to an unconstrained project scheduling problem and solved efficiently via minimum cut computations in a suitably constructed network, following an approach of Möhring et al. [41].

3.4.3 Solving the lagrangean relaxation by minimum cut computations

For this section fix Lagrange multipliers $\mu, \pi \in \mathbb{R}^T_{\geqslant 0}$ and consider $D\text{-}LR(\mu, \pi)$. The z-variables are merely bounded below by 0 and bounded above by corresponding y-variables according to constraints (3.15d). Hence, their value in an optimal solution may be deduced from their coefficient in the objective function.

For $k \in \{1, \ldots, K\}$, $i \in \mathcal{K}_k$, $t \in \{1, \ldots, T\}$,

$$\kappa_{i,t} = p_{i,t} - \pi_t a_i$$

is the objective coefficient of $z_{i,t}$ and

$$z^*_{i,t} = \begin{cases} 0 & \text{if } \kappa_{i,t} < 0, \\ y^*_{k,t} & \text{if } \kappa_{i,t} > 0, \end{cases} \tag{3.16}$$

will hold in any optimal solution (x^*, y^*, z^*) of *D-LR*(μ, π). For each aggregate $\mathcal{K}_k$, $k \in \{1, \dots, K\}$, and $t \in \{1, \dots, T\}$, define the "sub-aggregate"

$$\mathcal{K}^+_{k,t} = \{i \in \mathcal{K}_k \mid \kappa_{i,t} > 0\}.$$

With this notation and (3.16), the objective function (3.14) transforms to

$$\begin{aligned} &f_{xy\text{-}D\text{-}LR(\mu,\pi)}(x, y) \\ &\quad = \sum_{k=1}^{K} \sum_{t=1}^{T} -\bar{c}_{k,t} y_{k,t} + \sum_{k=1}^{K} \sum_{t=1}^{T} \sum_{i \in \mathcal{K}^+_{k,t}} p_{i,t} y_{k,t} \\ &\quad + \sum_{t=1}^{T} \mu_t \left(U^{\mathrm{m}}_t - \sum_{k=1}^{K} \bar{a}_k y_{k,t} \right) + \sum_{t=1}^{T} \pi_t \left(U^{\mathrm{p}}_t - \sum_{k=1}^{K} \sum_{i \in \mathcal{K}^+_{k,t}} a_i y_{k,t} \right) \\ &\quad = \sum_{k=1}^{K} \sum_{t=1}^{T} w_{k,t} y_{k,t} + \sum_{t=1}^{T} \left(\mu_t U^{\mathrm{m}}_t + \pi_t U^{\mathrm{p}}_t \right), \end{aligned}$$

where

$$w_{k,t} = -\bar{c}_{k,t} - \mu_t \bar{a}_k + \sum_{i \in \mathcal{K}^+_{k,t}} \underbrace{p_{i,t} - \pi_t a_i}_{=\kappa_{i,t}}.$$

Thus we obtain the programme *xy-D-LR*(μ, π):

maximise

$$f_{xy\text{-}D\text{-}LR(\mu,\pi)}(x, y) = \sum_{k=1}^{K} \sum_{t=1}^{T} w_{k,t} y_{k,t} + \sum_{t=1}^{T} \left(\mu_t U^{\mathrm{m}}_t + \pi_t U^{\mathrm{p}}_t \right) \tag{3.17}$$

subject to

$$x_{k,t-1} - x_{k,t} \leqslant 0 \qquad \text{for all } k \in \{1, \dots, K\},\ t \in \{2, \dots, T\}, \tag{3.17a}$$

$$\sum_{s=1}^{t} y_{k,s} - x_{k,t} \leqslant 0 \qquad \text{for all } k \in \{1, \dots, K\},\ t \in \{1, \dots, T\}, \tag{3.17b}$$

$$x_{k,t} - \sum_{s=1}^{t} y_{\ell,s} \leqslant 0 \qquad \text{for all } k \in \{1, \dots, K\},\ \ell \in \bar{\mathcal{P}}(k),\ t \in \{1, \dots, T\}, \tag{3.17c}$$

$$x_{k,t} \in \{0, 1\} \text{ for all } k \in \{1, \dots, K\},\ t \in \{1, \dots, T\}, \tag{3.17d}$$

$$0 \leqslant y_{k,t} \leqslant 1 \qquad \text{for all } k \in \{1, \dots, K\},\ t \in \{1, \dots, T\}. \tag{3.17e}$$

Step 1 in the proof of Proposition 3.10 showed that the feasible region is an integral polyhedron. Hence, we may assume the y-variables to be binary.

Furthermore, it can be seen that in any feasible solution, decreasing the value of $x_{k,t}$ to $\sum_{s=1}^{t} y_{k,s}$ for all $k \in \{1,\dots,K\}$ and $t \in \{1,\dots,T\}$ will preserve feasibility. The objective function $f_{xy\text{-}D\text{-}LR(\mu,\pi)}$ does not depend on the x-variables, hence we may assume without loss in the objective value, that constraint (3.17b) holds with equality and substitute the x-variables accordingly. This yields the programme $y\text{-}D\text{-}LR(\mu,\pi)$:

maximise

$$f_{y\text{-}D\text{-}LR(\mu,\pi)}(y) = \sum_{k=1}^{K}\sum_{t=1}^{T} w_{k,t}y_{k,t} + \sum_{t=1}^{T}\left(\mu_t U_t^{\mathrm{m}} + \pi_t U_t^{\mathrm{p}}\right) \tag{3.18}$$

subject to

$$\sum_{t=1}^{T} y_{k,t} \leqslant 1 \qquad \text{for all } k \in \{1,\dots,K\}, \tag{3.18a}$$

$$\sum_{s=1}^{t} y_{k,s} - \sum_{s=1}^{t} y_{\ell,s} \leqslant 0 \qquad \text{for all } k \in \{1,\dots,K\},\ \ell \in \bar{\mathcal{P}}(k),\ t \in \{1,\dots,T\}, \tag{3.18b}$$

$$y_{k,t} \in \{0,1\} \text{ for all } k \in \{1,\dots,K\},\ t \in \{1,\dots,T\}, \tag{3.18c}$$

which has almost the form of an unconstrained project scheduling problem. It only remains to add a slack time period $T+1$, during which all the aggregates not mined can be scheduled with zero cost. The resulting project scheduling formulation $PS\text{-}D\text{-}LR(\mu,\pi)$ reads

maximise

$$f_{y\text{-}D\text{-}LR(\mu,\pi)}(y) = \sum_{k=1}^{K}\sum_{t=1}^{T} w_{k,t}y_{k,t} + \sum_{t=1}^{T}\left(\mu_t U_t^{\mathrm{m}} + \pi_t U_t^{\mathrm{p}}\right) \tag{3.19}$$

subject to

$$\sum_{t=1}^{T+1} y_{k,t} = 1 \qquad \text{for all } k \in \{1,\dots,K\}, \tag{3.19a}$$

$$\sum_{s=1}^{t} y_{k,s} - \sum_{s=1}^{t} y_{\ell,s} \leqslant 0 \qquad \text{for all } k \in \{1,\dots,K\},\ \ell \in \bar{\mathcal{P}}(k),\ t \in \{1,\dots,T\}, \tag{3.19b}$$

$$y_{k,t} \in \{0,1\} \qquad \text{for all } k \in \{1,\dots,K\},\ t \in \{1,\dots,T{+}1\}. \tag{3.19c}$$

Möhring et al. [41] present an algorithm to solve unconstrained project scheduling efficiently by minimum cut computations in a suitably constructed network, which we will outline in the following.

For fixed Lagrange multipliers, the term $\sum_{t=1}^{T}\left(\mu_t U_t^{\mathrm{m}} + \pi_t U_t^{\mathrm{p}}\right)$ is an additive constant and can be disregarded when finding optimal solutions. Furthermore, we can shift the objective coefficients of the variables $y_{k,1},\dots,y_{k,T+1}$ for

some $k \in \{1, \dots, K\}$ by the same arbitrary constant C – due to constraints (3.19a) this is equivalent to adding a constant offset of C to the objective function:

$$\sum_{t=1}^{T}(w_{k,t}+C)y_{k,t} + Cy_{k,T+1} = \sum_{t=1}^{T} w_{k,t}y_{k,t} + \sum_{t=1}^{T+1} Cy_{k,t} = \sum_{t=1}^{T} w_{k,t}y_{k,t} + C.$$

We use this to transform *PS-D-LR*(μ, π) to a minimisation problem with non-negative objective coefficients. For each $k \in \{1, \dots, K\}$ let

$$w_k^{\max} = \max\left\{\max_{t=1,\dots,T} w_{k,t}, 0\right\}$$

and define

$$w_{k,t}^{+} = \begin{cases} w_k^{\max} - w_{k,t} & \text{for } t = 1, \dots, T, \\ w_k^{\max} & \text{for } t = T{+}1. \end{cases}$$

Then the objective function

$$\begin{aligned}
\sum_{k=1}^{K}\sum_{t=1}^{T+1} w_{k,t}^{+}y_{k,t} &= \sum_{k=1}^{K}\left(\sum_{t=1}^{T}(w_k^{\max} - w_{k,t})y_{k,t} + w_k^{\max}y_{k,T+1}\right) \\
&= \sum_{k=1}^{K}\left(-\sum_{t=1}^{T} w_{k,t}y_{k,t} + \sum_{t=1}^{T+1} w_k^{\max}y_{k,t}\right) \\
&= -\sum_{k=1}^{K}\sum_{t=1}^{T} w_{k,t}y_{k,t} + \sum_{k=1}^{K} w_k^{\max} \\
&= -f_{\textit{PS-D-LR}(\mu,\pi)}(y) + \underbrace{\sum_{t=1}^{T}\left(\mu_t U_t^{\mathrm{m}} + \pi_t U_t^{\mathrm{p}}\right) + \sum_{k=1}^{K} w_k^{\max}}_{\text{constant w.r.t. } y}
\end{aligned}$$

differs from $f_{\textit{PS-D-LR}(\mu,\pi)}$ only by its sign and an offset which is constant with respect to y. Thus, an optimal solution for *PS-D-LR*(μ, π) can be computed by solving the unconstrained project scheduling problem

minimise

$$\sum_{k=1}^{K}\sum_{t=1}^{T+1} w_{k,t}^{+}y_{k,t} \tag{3.20}$$

subject to (3.19a), (3.19b), (3.19c).

To this end, we construct the following capacitated digraph $D = (V, A)$ according to Möhring et al. [41]: The node set contains a source a, a sink b and one node for each $k \in \{1, \dots, K\}$ and $t \in \{0, \dots, T{+}1\}$, i.e.

$$V = \{a, b\} \cup \{v_{k,t} \mid k = 1, \dots, K,\ t = 0, \dots, T{+}1\}.$$

$A \subseteq V \times V$ consists of three types of arcs:

- auxiliary arcs $(a, v_{k,0})$ leaving the source and $(v_{k,T+1}, b)$ entering the sink for all $k \in \{1, \ldots, K\}$;
- assignment arcs $e_{k,t} = (v_{k,t-1}, v_{k,t})$ for all $k \in \{1, \ldots, K\}$ and $t \in \{1, \ldots, T+1\}$ – one directed chain for each aggregate; and
- precedence arcs $(v_{\ell,t}, v_{k,t})$ for all $k \in \{1, \ldots, K\}$, $\ell \in \bar{\mathcal{P}}(k)$ and $t \in \{1, \ldots, T+1\}$.

Each assignment arc $e_{k,t} = (v_{k,t-1}, v_{k,t})$ for $k \in \{1, \ldots, K\}$, $t \in \{1, \ldots, T+1\}$ takes capacity $c(v_{k,t-1}, v_{k,t}) = w_{k,t}$, which is non-negative by definition. The auxiliary arcs and the precedence arcs are assigned infinite capacity.

By an a-b-cut of D we understand any ordered pair $(X, \bar{X})$ of disjoint node sets $X, \bar{X} \subseteq V$ with $X \cup \bar{X} = V$ and $a \in X$, $b \in \bar{X}$. Its capacity is defined by

$$c(X, \bar{X}) = \sum_{u \in X, \bar{u} \in \bar{X}} c(u, \bar{u}),$$

possibly infinite. A minimum a-b-cut is an a-b-cut of minimal capacity.

We call an a-b-cut of D a K-cut if it contains exactly one assignment arc for each aggregate. For now, the capacities of assignment arcs are all finite, and an a-b-cut of finite capacity always exists. In Section 3.5, though, we will incorporate valid inequalities into the relaxations which will result in possibly infinite capacities even on the assignment arcs.

Lemma 3.12 *Let $(X, \bar{X})$ be a minimum a-b-cut of D. Then there exists a K-cut $(X^*, \bar{X}^*)$ of D with the same capacity. Given $(X, \bar{X})$, the K-cut $(X^*, \bar{X}^*)$ can be computed in time $\mathcal{O}(KT)$.*

Proof. W.l.o.g. we may assume that $c(X, \bar{X})$ is finite. $(X, \bar{X})$ must contain at least one assignment arc for each aggregate $\mathcal{K}_k$, $k \in \{1, \ldots, K\}$, otherwise there would be a path $(a, v_{k,0}, \ldots, v_{k,T+1}, b)$ connecting a and b. It can be checked that choosing for each aggregate only the assignment arc in the cut with lowest time index already gives an a-b-cut, for details see Möhring et al. [41]. This gives a K-cut with the same capacity, since $c(X, \bar{X})$ was already minimal.

To compute this K-cut, given $(X, \bar{X})$, one must inspect at most $T+1$ nodes for each aggregate. □

The relation between feasible schedules for (3.20) and finite K-cuts is intuitive. Given a finite K-cut, schedule each aggregate $\mathcal{K}_k$, $k \in \{1, \ldots, K\}$, in the time period $t \in \{1, \ldots, T\}$ for which the corresponding assignment arc $e_{k,t}$ is in the cut – or schedule $\mathcal{K}_k$ not at all if $e_{k,T+1}$ was included. Feasibility of the

resulting schedule is guaranteed since precedence arcs can never be included in a finite cut.

Conversely, any feasible schedule for (3.20) immediately gives a finite K-cut: For each $k \in \{1, \dots, K\}$, if aggregate $\mathcal{K}_k$ is scheduled in time period t, then delete the assignment arc $e_{k,t}$ from D, respectively $e_{k,T+1}$ if $\mathcal{K}_k$ was not scheduled. The project scheduling constraints guarantee that this splits D into two components which indeed form a finite a-b-cut. The objective function value of a schedule equals the capacity of the corresponding cut by construction. The following proposition now gives the precise relation between a-b-cuts of D and the original lagrangean relaxation $D\text{-}LR(\mu, \pi)$:

Proposition 3.13 *Let Lagrange multipliers $\mu, \pi \in \mathbb{R}^T_{\geqslant 0}$ be fixed. $D\text{-}LR(\mu, \pi)$ is feasible if and only if the digraph D contains an a-b-cut of finite value. Let (x, y, z) be an optimal solution of $D\text{-}LR(\mu, \pi)$, then $(X, V - X)$ with*

$$X = \{a\} \cup \{v_{k,0} \,|\, k = 1, \dots, K\} \cup \{v_{k,t} \,|\, x_{k,t} = 0,\, k = 1, \dots, K,\, t = 1, \dots, T\}$$

is a minimum K-cut of D. Conversely, let $(X, \bar{X})$ be a finite minimum K-cut of D, and define for each $k \in \{1, \dots, K\}$, $i \in \mathcal{K}_k$, $t \in \{1, \dots, T\}$,

$$x_{k,t} = \begin{cases} 1 & \text{if } v_{k,t} \in \bar{X}, \\ 0 & \text{otherwise,} \end{cases}$$

$$y_{k,t} = \begin{cases} 1 & \text{if } v_{k,t-1} \in X, v_{k,t} \in \bar{X}, \\ 0 & \text{otherwise,} \end{cases}$$

$$z_{i,t} = \begin{cases} y_{k,t} & \text{if } i \in \mathcal{K}^+_{k,t}, \\ 0 & \text{otherwise.} \end{cases}$$

Then (x, y, z) is an optimal solution of $D\text{-}LR(\mu, \pi)$.

Proof. In the above discussion we have explained in detail the relation between the solution sets of $D\text{-}LR(\mu, \pi)$ and (3.20). The mapping

$$F : \{0,1\}^{K \times (T+1)} \longrightarrow \mathbb{R}^{K \times T} \times \mathbb{R}^{K \times T} \times \mathbb{R}^{N \times T},\ y \longmapsto (\hat{x}, \hat{y}, \hat{z})$$

defined by

$$\hat{x}_{k,t} = \sum_{s=1}^{t} y_{k,s},$$

$$\hat{y}_{k,t} = y_{k,t},$$

$$\hat{z}_{i,t} = \begin{cases} y_{k,t} & \text{if } i \in \mathcal{K}^+_{k,t}, \\ 0 & \text{otherwise,} \end{cases}$$

for all $k \in \{1, \dots, K\}$, $i \in \mathcal{K}_k$ and $t \in \{1, \dots, T\}$, maps every solution y of (3.20) injectively to a solution $F(y)$ of $D\text{-}LR(\mu, \pi)$. The objective function values are related by

$$\sum_{k=1}^{K} \sum_{t=1}^{T+1} w_{k,t}^{+} y_{k,t} = -f_{D\text{-}LR(\mu,\pi)}(F(y)) + \underbrace{\sum_{t=1}^{T} \left(\mu_t U_t^{\mathrm{m}} + \pi_t U_t^{\mathrm{p}}\right) + \sum_{k=1}^{K} w_k^{\max}}_{\text{constant w.r.t. } y},$$

so minimisers of (3.20) correspond to maximisers of $D\text{-}LR(\mu, \pi)$. F is not surjective on the set of feasible solutions of $D\text{-}LR(\mu, \pi)$, but the optimal objective value of $D\text{-}LR(\mu, \pi)$ is contained in the image of $f_{D\text{-}LR(\mu,\pi)} \circ F$. Therefore, optimal solutions of $D\text{-}LR(\mu, \pi)$ can be computed by first solving (3.20) and subsequently applying the transformation F. Feasible solutions of (3.20) are in turn in a natural one-to-one-correspondence with finite K-cuts of D as explained in the last preceding paragraph, for details see the proof by Möhring et al. [41].

Now, because its feasible region is bounded, $D\text{-}LR(\mu, \pi)$ is feasible if and only if it has an optimal solution. Those are exactly of the form $F(y)$ for a solution y of (3.20). Since solutions of (3.20) correspond one-to-one to finite K-cuts of D, the first statement about feasibility is proven. Following this chain of relations in detail yields exactly the correspondences between optimal solutions as stated in the proposition. □

This shows that $D\text{-}LR(\mu, \pi)$ can be solved efficiently. The algorithm described has the following complexity:

Corollary 3.14 *For fixed Lagrange multipliers $\mu, \pi \in \mathbb{R}^T_{\geqslant 0}$, $D\text{-}LR(\mu, \pi)$ can be solved in time $\mathcal{O}(NT + mKT^2 \log(K^2T/m))$, if $m = \sum_{k=1}^{K} \left|\bar{\mathcal{P}}(k)\right|$ is in $\Omega(K)$.*

Proof. The digraph consists of $K(T+2)+2$, i.e. $\mathcal{O}(KT)$, nodes and $K(T+3)+m(T-1)$ arcs, which is $\mathcal{O}(mT)$ if we assume that m is $\Omega(K)$, i.e. that the precedence graph is not unrealistically sparse.[1] The analysis of the push-relabel-algorithm by Goldberg and Tarjan [23] gives the complexity of computing a minimum a-b-cut of D as $\mathcal{O}(mKT^2 \log(K^2T/m))$. Converting this into a K-cut is possible in $\mathcal{O}(KT)$ time, and deducing the solution for the original problem $D\text{-}LR(\mu, \pi)$ has complexity $\mathcal{O}(NT)$, because the values of the z-variables need to be determined. □

[1] By definition, $m \in \Omega K$ if $K \in \mathcal{O}(m)$.

3.4.4 Lagrange multipliers and cutoff grades

Consider again the original lagrangean relaxation $D\text{-}LR(\mu, \pi)$ for some fixed multipliers $\mu, \pi \in \mathbb{R}^T_{\geqslant 0}$. Above we already pointed out that in any optimal solution (x^*, y^*, z^*) the z-variables must satisfy condition (3.16), i.e.

$$z^*_{i,t} = \begin{cases} 0 & \text{if } \kappa_{i,t} < 0, \\ y^*_{k,t} & \text{if } \kappa_{i,t} > 0, \end{cases}$$

with $\kappa_{i,t} = p_{i,t} - \pi_t a_i$ for all $i \in \mathcal{N}$ and $t \in \{1, \ldots, T\}$. This means that in some time period $t \in \{1, \ldots, T\}$, a block $i \in \mathcal{N}$ is selected for processing if the ratio of processing profit to rock weight $\frac{p_{i,t}}{a_i}$ is greater than the Lagrange multiplier π_t, and discarded as waste if it is less.

The processing profit $p_{i,t}$ is typically of the form

$$\frac{a_i^{\text{ore}} C^{\text{ore}} - a_i C^{\text{p}}}{(1+r)^t},$$

where r is some discount rate, a_i^{ore} denotes the amount of ore in block i, C^{ore} the sale price per unit ore and C^{p} the processing cost per unit rock. Then

$$\begin{aligned} \kappa_{i,t} > 0 &\iff \left(a_i^{\text{ore}} C^{\text{ore}} - a_i C^{\text{p}}\right)/(1+r)^t > \pi_t a_i \\ &\iff \frac{a_i^{\text{ore}}}{a_i} > \frac{(1+r)^t \pi_t + C^{\text{p}}}{C^{\text{ore}}}, \end{aligned}$$

i.e. block i is selected for processing in time period t if its oregrade exceeds the value of $\frac{(1+r)^t \pi_t + C^{\text{p}}}{C^{\text{ore}}}$ – exactly a cutoff grade decision depending on the Lagrange multipliers for the processing constraints.

If we let $\pi_t = 0$, i.e. if we do not penalise violation of the processing constraint and allow for arbitrarily much rock to be processed, we get the lowest reasonable cutoff grade of $\frac{C^{\text{p}}}{C^{\text{ore}}}$ and select all blocks whose processing profit just outweighs the cost. As π_t grows, the cutoff grade also grows, and more and more material with oregrade above this basic cutoff grade is discarded so as to avoid further increase of processing constraint violation.

An analogous interpretation of Lagrange multipliers is given by Caccetta et al. [10] for a slightly different OPMPSP-formulation. Further insight is gained when looking at the role of optimal Lagrange multipliers, i.e. of Lagrange multipliers $\mu, \pi \in \mathbb{R}^T_{\geqslant 0}$ which solve the lagrangean dual and for which $D\text{-}LR(\mu, \pi)$ yields an upper bound as low as the LP-bound: $\phi(\mu, \pi) = f^*_{D\text{-}LR(\mu,\,\pi)} = \phi^*$. Because the precedence polyhedron remaining after relaxation of the resource constraints is integral, optimal Lagrange multipliers are exactly optimal dual multipliers associated with the relaxed constraints in the LP-relaxation (see e.g. Geoffrion [20]). With this in mind, the following result appears natural:

Proposition 3.15 *Let $\pi^* \in \mathbb{R}^T_{\geqslant 0}$ be a vector of optimal dual multipliers associated with the processing constraints*

$$\sum_{i=1}^{N} a_i z_{i,t} \leqslant U_t^{\mathrm{p}} \quad \text{for all } t \in \{1, \ldots, T\}$$

in D-LP. Then $\pi^* = (\pi_1^*, \ldots, \pi_T^*)$ is a sequence of split ratios for D-LP as defined in Section 3.2. Even more,

$$z_{i,t}^* = \begin{cases} 0 & \text{if } \frac{p_{i,t}}{a_i} < \pi_t^*, \\ y_{k,t}^* & \text{if } \frac{p_{i,t}}{a_i} > \pi_t^*, \end{cases}$$

holds for all $k \in \{1, \ldots, K\}$, $i \in \mathcal{K}_k$, $t \in \{1, \ldots, T\}$, not only for some, but for any optimal solution (x^*, y^*, z^*) of D-LP.

Proof. Fix any optimal primal solution (x^*, y^*, z^*) of *D-LP*. Let ω denote the vector of (non-negative) dual variables for constraints (2.5d) in *D-LP*. Fix values for $k \in \mathcal{K}$, $i \in \mathcal{K}_k$ and $t \in \{1, \ldots, T\}$. The primal variable $z_{i,t}$ corresponds to the constraint

$$\omega_{i,t} + a_i \pi_t \geqslant p_{i,t}$$

in the dual programme. By definition, the dual variable $\omega_{i,t}$ corresponds to the constraint $z_{i,t} \leqslant y_{k,t}$. Complementary slackness yields

$$\begin{aligned} \omega_{i,t}^* \left(y_{k,t}^* - z_{i,t}^*\right) &= 0, \\ z_{i,t}^* \left(\omega_{i,t}^* + a_i \pi_t^* - p_{i,t}\right) &= 0, \end{aligned}$$

where $\omega_{i,t}^*$ denotes the value of the dual variable $\omega_{i,t}$ in the same optimal dual solution from which π^* was taken. Now, if $\frac{p_{i,t}}{a_i} < \pi_t^*$ then

$$\omega_{i,t}^* + a_i \pi_t^* - p_{i,t} > \omega_{i,t}^* \geqslant 0$$

and so $z_{i,t}^* = 0$. If $\frac{p_{i,t}}{a_i} > \pi_t^*$ then

$$\omega_{i,t}^* \geqslant p_{i,t} - a_i \pi_t^* > 0$$

and $z_{i,t}^* = y_{k,t}^*$. □

3.5 Valid inequalities

This section treats the addition of valid inequalities to the LP- and lagrangean relaxation in order to tighten the dual bounds. We recall the following basic definition from integer programming:

Definition 3.16 *An inequality $b^\top x \leqslant b_0$ with $b \in \mathbb{R}^n$, $b_0 \in \mathbb{R}$, is called* valid inequality *for a set $X \subseteq \mathbb{R}^n$ if $b^\top x \leqslant b_0$ holds for all $x \in X$.*

Here, X is typically the feasible region of an integer or mixed-integer linear programme. By definition, adding a valid inequality $b^\top x \leqslant b_0$ to the list of constraints does not change the feasible region itself, i.e. $\{x \in X \mid b^\top x \leqslant b_0\} = X$. However, in general it can restrict the domain over which the LP-relaxation is solved. For "good" valid inequalities, this results in tighter dual bounds.

First, we will show how general valid inequalities for *D-MIP* are integrated into the LP- and lagrangean relaxation. Second, we will look at problem specific valid inequalities for *D-MIP*, in particular valid fixings of single variables.

3.5.1 Integrating valid inequalities in the LP-relaxation

A general valid inequality for *D-MIP* has the form

$$b^{1\top} x + b^{2\top} y + b^{3\top} z \leqslant b_0, \tag{3.21}$$

where $b^1, b^2 \in \mathbb{R}^{K\times T}$, $b^3 \in \mathbb{R}^{N\times T}$ and $b_0 \in \mathbb{R}$. (Because the zero solution is always feasible, we know that $b_0 \geqslant 0$.) This can simply be appended to the list of constraints of the standard LP-relaxation *D-LP*. For the reduced LP-relaxations *x-D-LP* and *y-D-LP* as well as for integration into the lagrangean approach, the following observation is crucial:

Lemma 3.17 *Let* (3.21) *be a valid inequality for D-MIP with* $b^1 \geqslant 0$*, then*

$$\sum_{k=1}^{K}\sum_{t=1}^{T}\left(b^2_{k,t} + \sum_{s=t}^{T} b^1_{k,s}\right) y_{k,t} + \sum_{i=1}^{N}\sum_{t=1}^{T} b^3_{i,t} z_{i,t} \leqslant b_0 \tag{3.22}$$

is also valid for D-MIP. The optimal objective values of D-LP with (3.21)*, of y-D-LP with* (3.22) *and of x-D-LP with*

$$\sum_{k=1}^{K}\left[\sum_{t=1}^{T-1}\left(b^1_{k,t} + b^2_{k,t} - b^2_{k,t+1}\right) x_{k,t} + \left(b^1_{k,T} + b^2_{k,T}\right) x_{k,T}\right] + \sum_{i=1}^{N}\sum_{t=1}^{T} b^3_{i,t} z_{i,t} \leqslant b_0 \tag{3.23}$$

are equal.

Proof. By definition, (3.21) holds for any feasible solution (x, y, z) of *D-MIP*. For $b^1 \geqslant 0$, constraint (2.5b) yields

$$\sum_{k=1}^{K}\sum_{t=1}^{T}\left(\sum_{s=t}^{T} b^1_{k,s}\right) y_{k,t} = \sum_{k=1}^{K}\sum_{t=1}^{T} b^1_{k,t}\left(\sum_{s=1}^{t} y_{k,s}\right) \leqslant \sum_{k=1}^{K}\sum_{t=1}^{T} b^1_{k,t} x_{k,t}.$$

With this, (3.22) follows directly from (3.21).

The solution sets of *y-D-LP* and *x-D-LP* are in one-to-one correspondence via the substitution $x_{k,t} = \sum_{s=1}^{t} y_{k,s}$. This is exactly how (3.23) is derived from (3.22) and vice versa, thus also the solution sets of *y-D-LP* with (3.22) and of *x-D-LP* with (3.23) are in one-to-one correspondence and the optimal objective values are equal.

The rest follows if we assume that (2.5b) holds with equality in *D-LP*. As explained in Section 3.1, this does not affect the objective value. □

First, note that in general (3.23) is not valid for *D-MIP*. Consider for example the mining constraints (2.5e), which are trivially valid inequalities. For these, (3.23) reads

$$\sum_{k=1}^{K} \bar{a}_k x_{k,1} \leqslant U_1^{\mathrm{m}} \text{ resp. } \sum_{k=1}^{K} \bar{a}_k (x_{k,t} - x_{k,t-1}) \leqslant U_t^{\mathrm{m}} \text{ for all } t \in \{2, \dots, T\},$$

(For the variant *D-MIP'*, where each aggregate must be completely mined during one time period, (3.23) is valid, though.)

Second, the condition $b^1 \geqslant 0$ holds for a large class of valid inequalities, for example for combinatorial cutting planes such as the knapsack cover inequalities mentioned in Section 3.5.3, where coefficients are 0 or 1.

Here, we treated only the case of one single valid inequality, but everything holds equally for arbitrarily many valid inequalities.

3.5.2 Integrating valid inequalities in the lagrangean relaxation

Consider again a valid inequality for *D-MIP* of the form (3.21). One possibility is to append (3.21) to the list of constraints of the lagrangean relaxation. This would destroy the integrality of the feasible region and the solution via computing a minimum K-cut could not be applied anymore.

Hence, we rather dualise the valid inequality in the same way as the resource constraints. However, the transformation of the lagrangean relaxation to a project scheduling problem relied on zero coefficients of the x-variables in the objective function – this is not satisfied if $b^1 \neq 0$ in (3.21). Under the assumption that $b^1 \geqslant 0$, Proposition 3.17 showed that (3.22) is equally valid and yields the same dual bound. Compared to the original one, it has the favourable property that it does not depend on the x-variables. Now, if we introduce a non-negative Lagrange multiplier α and penalise the violation of (3.22) in the objective function, then the extended lagrangean relaxation reads

maximise

$$\begin{aligned}
&\sum_{k=1}^{K} \sum_{t=1}^{T} -\bar{c}_{k,t} y_{k,t} + \sum_{i=1}^{N} \sum_{t=1}^{T} p_{i,t} z_{i,t} \\
&+ \sum_{t=1}^{T} \mu_t \left(U_t^{\mathrm{m}} - \sum_{k=1}^{K} \bar{a}_k y_{k,t} \right) + \sum_{t=1}^{T} \pi_t \left(U_t^{\mathrm{p}} - \sum_{i=1}^{N} a_i z_{i,t} \right) \\
&+ \alpha \left[b_0 - \sum_{k=1}^{K} \sum_{t=1}^{T} \left(b_{k,t}^2 + \sum_{s=t}^{T} b_{k,s}^1 \right) y_{k,t} - \sum_{i=1}^{N} \sum_{t=1}^{T} b_{i,t}^3 z_{i,t} \right]
\end{aligned} \tag{3.24}$$

subject to (3.15a), (3.15b), (3.15c), (3.15d), (3.15e), (3.15f), (3.15g).

The underlying polyhedron is the same as for the original lagrangean relaxation and thus integral. Hence, Proposition 3.11 continues to hold: The lowest possible bound obtained from the extended lagrangean relaxation (3.24) equals the optimal objective value of *D-LP* with (3.21) added.

For fixed Lagrange multipliers $\mu, \pi \in \mathbb{R}^T_{\geqslant 0}$, $\alpha \in \mathbb{R}_{\geqslant 0}$, the extended lagrangean relaxation can be solved by computing a minimum K-cut as described in Section 3.4 – only the new objective coefficients must be taken into account. The objective coefficient of variable $z_{i,t}$, $i \in \mathcal{N}$, $t \in \{1, \ldots, T\}$, is now

$$\kappa_{i,t} = p_{i,t} - \pi_t a_i - \alpha b^3_{i,t},$$

and in general $\mathcal{K}^+$, w and w^+ change.

Note that adding valid inequalities which involve some of the z-variables can destroy the structure given by split ratios as presented in Section 3.2. Also the relation between Lagrange multipliers and cutoff grades given in Section 3.4.4 is distorted. For valid inequalities (3.21) with $b^3 = 0$, though, the outlined structure on the z-variables is preserved.

Proposition 3.15 gave the result that any optimal dual multiplier vector π for the processing constraints gives split ratios for *D-LP*. If we add valid inequalities to *D-LP* which do not involve the z-variables, then this result continues to hold. Since *D-LP* strengthened by the additional valid inequalities gives a better approximation of *D-MIP*, the split ratios for this strengthened LP-relaxation might be expected to be closer to the split ratios for *D-MIP*. In particular the problem specific valid inequalities presented next only depend on the x-variables.

3.5.3 OPMPSP-specific valid inequalities

In Section 2.4.1 we pointed out the structure of the precedence-constrained knapsack problem found in the OPMPSP-formulations. Inequalities which are valid for the precedence-constrained knapsack problem can be used to deduce valid inequalities for *D-MIP* and *D-MIP'*. Prototypical examples are knapsack cover-like inequalities:

Consider the knapsack problem *KP* from Section 2.4.1. We call s subset of items $S \subseteq \mathcal{N}$ a *cover* if not all of the items in S can be included in the knapsack due to $\sum_{i \in S} w_i > U$. Then the at most $|S| - 1$ items from S can be chosen, i.e. the so-called knapsack cover inequality $\sum_{i \in S} x_i \leqslant |S| - 1$ is valid for *KP*.

Fricke [18] gives an extensive exposition of well-known valid inequalities for the precedence-constrained knapsack problem and presents new results. From these, he derives valid inequalities for the two OPMPSP-formulations (2.3) and (2.4), which are also valid for *D-MIP'* and *D-MIP*, respectively. He tests their effect on solving the models by standard LP-based branch-and-cut – for

many inequalities the results were ambiguous, except for the simplest class, which we will explain in the following and use in all further experiments.

The basic idea is simple: As an example, suppose that for some aggregate $\mathcal{K}_k$, $k \in \{1, \dots, K\}$, the weights $\bar{a}_\ell$ of all aggregates $\mathcal{K}_\ell$ in the complete predecessor cone of $\mathcal{K}_k$ sum up to an amount greater than or equal to the mining capacity of the first time period U_1^{m}. Then it is intuitively clear that none of aggregate $\mathcal{K}_k$ can be mined in the first time period and $x_{k,1} \leqslant 0$ is valid for *D-MIP*. A valid inequality of this form is often called variable fixing. Denote by

$$\bar{\bar{\mathcal{P}}}(k) = \left\{ \ell \;\middle|\; \begin{array}{l} \exists\, (k = k_0, \dots, k_m = \ell) \in \{1, \dots, K\}^{\ell+1} \colon \\ k_i \in \bar{\mathcal{P}}(k_{i-1})\ \forall i = 1, \dots, m \end{array} \right\}$$

the set of indices of aggregates which are contained in the complete predecessor cone of some aggregate $\mathcal{K}_k$, $k \in \{1, \dots, K\}$. Then we can prove the following variable fixings to be valid for *D-MIP*:

Proposition 3.18 *If for $k \in \{1, \dots, K\}$ and $t \in \{1, \dots, T\}$,*

$$\sum_{\ell \in \bar{\bar{\mathcal{P}}}(k)} \bar{a}_\ell \geqslant \sum_{s=1}^{t} U_s^{\mathrm{m}},$$

then the variable fixing

$$y_{k,s} = 0$$

is valid for D-MIP for all $s \in \{1, \dots, t\}$. If even $\sum_{\ell \in \bar{\bar{\mathcal{P}}}(k)} \bar{a}_\ell > \sum_{s=1}^{t} U_s^{\mathrm{m}}$ holds, then also

$$x_{k,t} = 0$$

is valid for D-MIP.

Proof. We prove the contraposition: Suppose $x_{k,t} = 1$ in a feasible solution of *D-MIP*, then

$$\begin{aligned} \sum_{\ell \in \bar{\bar{\mathcal{P}}}(k)} \bar{a}_\ell = \sum_{\ell \in \bar{\bar{\mathcal{P}}}(k)} \bar{a}_\ell x_{k,t} &\overset{(2.5c)}{\leqslant} \sum_{\ell \in \bar{\bar{\mathcal{P}}}(k)} \bar{a}_\ell \left(\sum_{s=1}^{t} y_{\ell,s} \right) + \bar{a}_k \sum_{s=1}^{t} y_{k,s} \\ &\overset{(\star)}{\leqslant} \sum_{\ell=1}^{K} \bar{a}_\ell \left(\sum_{s=1}^{t} y_{\ell,s} \right) = \sum_{s=1}^{t} \left(\sum_{\ell=1}^{K} \bar{a}_\ell y_{\ell,s} \right) \overset{(2.5e)}{\leqslant} \sum_{s=1}^{t} U_s^{\mathrm{m}}, \end{aligned}$$

must hold, the negation of the inequality required in the proposition. In the case when even $\sum_{s=1}^{t} y_{k,s} > 0$, then inequality $(\star)$ is strict since we assume $\bar{a}_k$ to be strictly positive. □

The reason why the weak inequality condition does not imply valid fixings on x-variables is that a value of 1 for some $x_{k,t}$, $k \in \{1, \dots, K\}$, $t \in \{1, \dots, T\}$,

only allows the mining of aggregate $\mathcal{K}_k$ to be started in time period t – it does not force mining to be started, i.e. $\sum_{s=1}^{t} y_{k,s}$ can still be zero. Although fixing $x_{k,t}$ to zero in the case where $\sum_{\ell \in \bar{\bar{\mathcal{P}}}(k)} \bar{a}_\ell = \sum_{s=1}^{t} U_s^{\mathrm{m}}$ is not theoretically valid, it does not cut off any feasible mine schedules, which are in fact given by the values of y- and z-variables. In the reduced LP-relaxations and in the lagrangean approach, both cases coincide automatically.

For the LP-relaxation, the variable fixings from Proposition 3.18 can simply be added to the list of constraints in order to strengthen the relaxation. For the lagrangean relaxation, they could theoretically be treated as valid inequalities as described in the last preceding section – introduce a new Lagrange multiplier and dualise the "inequality". This increases the space of dual multipliers and makes solving the lagrangean dual more difficult.

Dualising the variable fixing inequality is unnecessary, though, since the variable fixing does not destroy the integrality of the precedence polytope and can be taken into account implicitly in the minimum cut approach: Suppose $x_{k,t}$ is fixed to zero for some $k \in \{1, \ldots, K\}$, $t \in \{1, \ldots, T\}$, then we only have to modify the minimum cut digraph D slightly: either by shortening the directed chain $(a, v_{k,0}, \ldots, v_{k,T+1}, b)$ to $(a, v_{k,t}, \ldots, v_{k,T+1}, b)$ and deleting the precedence arcs leaving nodes $v_{k,1}, \ldots, v_{k,t}$; or by setting the capacities on the assignment arcs $e_{k,1}, \ldots, e_{k,t}$ to infinity. Both methods guarantee that aggregate $\mathcal{K}_k$ is mined no earlier than during time period $t+1$.

3.6 Conclusion

This chapter provided a starting point for the next chapters by analysing basic structural properties of the OPMPSP-formulation *D-MIP*. Section 3.1 treated the standard LP-relaxation together with a simple reduction, Section 3.4 was concerned with relaxing the resource constraints in a lagrangean fashion. Both methods for computing dual bounds will be evaluated in detail in Chapter 4.

Section 3.2 highlighted important knapsack structure and its implications for optimal cutoff grades. These results will be the basis for the column aggregation techniques discussed in Chapter 5.

In Section 3.5, we showed how valid inequalities can be integrated in the reduced LP-relaxation and the lagrangean relaxation under a simple condition. We concluded by briefly outlining valid inequalities which arise from the precedence-constrained knapsack problem structure.

Chapter 4

Dual bound computation

Our central aim is naturally to obtain high quality OPMPSP-solutions, a challenging task for an NP-hard problem. To this end, heuristic methods will be presented and tested in Chapter 6. In some sense complementary, this chapter is concerned with how to evaluate the quality of primal solutions by computing dual bounds. Being able to obtain good bounds on the optimal objective value within reasonable running times is a key ingredient for many optimisation algorithms, such as for a classical branch-and-bound approach. The computation of dual bounds is not only of theoretical importance, though. It is crucial also in practise in order to evaluate the quality of solutions and provide engineers with confidence in the mine plans developed.

In Chapter 3 we considered two relaxations of *D-MIP*, the standard LP-relaxation with simplifications and a lagrangean relaxation of the resource constraints. This chapter compares the computational performance of both approaches. Section 4.1 gives a brief survey of several methods for solving the lagrangean dual. The computational results from Section 4.2 show that solving the lagrangean dual via a bundle algorithm clearly outperforms solving the LP-relaxation directly.

4.1 The lagrangean dual

4.1.1 The lagrangean dual – a convex nondifferentiable optimisation problem

In Section 3.4 we defined the dual function

$$\phi : \mathbb{R}^T_{\geqslant 0} \times \mathbb{R}^T_{\geqslant 0} \longrightarrow \mathbb{R},\ (\mu, \pi) \longmapsto f^*_{D\text{-}LR(\mu,\pi)},$$

which for a fixed vector of multipliers gives the optimal objective value of the lagrangean relaxation, providing an upper bound on the optimal objective

value of *D-MIP*. The so-called *lagrangean dual* is the problem of computing ϕ^*, the tightest bound achievable by any $\phi(\mu, \pi)$, i.e.

$$\phi^* = \min_{\mu,\pi \in \mathbb{R}^T_{\geqslant 0}} \phi(\mu, \pi).$$

The dual function ϕ is well-known to be piecewise-linear and convex,[1] thus solving the lagrangean dual poses a convex nondifferentiable optimisation problem. In the next sections, we will outline the most prominent methods from the literature for solving the lagrangean dual.

As a general setting, consider the primal problem

$$\max \{c^\top x \mid Ax \leqslant b,\, x \in Q\}, \tag{P}$$

where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $Q \subseteq \mathbb{R}^m$. In the case of P being a mixed-integer linear programme, Q takes the form $\{x \in \mathbb{Z}^{n_1} \times \mathbb{R}^{n_2} \mid Bx \leqslant d\}$.[2] For fixed Lagrange multiplier vector $\lambda \in \mathbb{R}^m_{\geqslant 0}$, the lagrangean relaxation with respect to constraints $Ax \leqslant b$ reads

$$\phi(\lambda) = \max_{x \in Q} \{c^\top x + \lambda^\top (b - Ax)\} \tag{LR}$$

and the lagrangean dual becomes

$$\phi^* = \min_{\lambda \geqslant 0} \phi(\lambda). \tag{LD}$$

A fundamental object in nondifferential, convex optimisation is the *subdifferential* of a function $f : \mathbb{R}^m \to \mathbb{R}$ at a point $u \in \mathbb{R}^m$, defined as

$$\partial f(u) = \{y \in \mathbb{R}^n \mid f(v) \geqslant f(u) + y^\top (v - u) \text{ for all } v \in \mathbb{R}^m\}.$$

Its elements are called *subgradients* of f at u and by definition provide a linear underestimation of f. At points, where f is differentiable, the subdifferential is formed by the gradient alone, but otherwise subgradients are not unique. By definition of the subdifferential, a point $u \in \mathbb{R}^m$ is a global minimiser of f if and only if $0 \in \partial f(u)$. In the lagrangean setting, subgradients are immediately at hand when the lagrangean relaxation is solved:

Theorem 4.1 *Let a Lagrange multiplier vector $\lambda \in \mathbb{R}^m_{\geqslant 0}$ be fixed and $x^0 \in Q$ be an optimal solution of the lagrangean relaxation, i.e. $\phi(\lambda) = c^\top x^0 + \lambda^\top (b - Ax^0)$. Then $b - Ax^0 \in \partial\phi(\lambda)$.*

[1] Let Q be the finitely many extreme points of the feasible region of *D-LR*(μ, π), then $f_{\text{D-LR}(\mu,\pi)}(x, y, z)$ is an affine function of (μ, π) for each fixed $(x, y, z) \in Q$. Thus, $\phi(\mu, \pi) = \max_{(x,y,z) \in Q} f_{\text{D-LR}(\mu,\pi)}(x, y, z)$ is piecewise-linear and convex as the maximum over finitely many affine functions.

[2] Lagrangean relaxation is a versatile and general tool and by far not restricted to (mixed-integer) linear programmes. Further prominent applications are found in Quadratic Programming, Semidefinite Programming and others.

Proof. For any $\lambda' \in \mathbb{R}^m_{\geqslant 0}$,

$$\begin{aligned}\phi(\lambda') &= \max_{x \in Q} \left\{ c^\top x + \lambda'^\top (b - Ax) \right\} \\ &\geqslant c^\top x^0 + \lambda'^\top (b - Ax^0) \\ &\geqslant c^\top x^0 + \lambda^\top (b - Ax^0) + (\lambda' - \lambda)^\top (b - Ax^0) \\ &= \phi(\lambda) + (b - Ax^0)^\top (\lambda' - \lambda),\end{aligned}$$

thus $b - Ax^0 \in \partial\phi(\lambda)$ by definition. □

Next, we give a brief survey of methods for solving the lagrangean dual, largely following Lemaréchal [36]. These methods are – except when we come to column generation – "oracle-based" in the sense that no further knowledge about the function ϕ is assumed than the existence of a "black box" type of algorithm or oracle, which for each $\lambda \in \mathbb{R}^m_{\geqslant 0}$, returns the function value $\phi(\lambda)$ together with a subgradient $g_\lambda \in \partial\phi(\lambda)$. The latter is typically obtained via Theorem 4.1, though this is not explicitly used by the general convex optimisation algorithms. Only the column generation approach, which we will present in connection with the cutting plane method, will make use of this specific structure.

4.1.2 The subgradient method (Uzawa [49])

The subgradient method initially devised by Uzawa [49] rests on the following observation. Suppose we start with a candidate $\lambda \in \mathbb{R}^m_{\geqslant 0}$, then by definition it holds for any $\lambda^* \in \mathbb{R}^m_{\geqslant 0}$ with lower function value, e.g. any minimiser of ϕ, that

$$\phi(\lambda) > \phi(\lambda^*) \geqslant \phi(\lambda) + g_\lambda^\top (\lambda^* - \lambda).$$

Hence, $g_\lambda^\top (\lambda^* - \lambda) < 0$, which can be used to show that for $t > 0$ sufficiently small, the point $\lambda - t g_\lambda$ is strictly closer to λ^* than λ.[3] This motivates an iterative algorithm, where starting from some multiplier vector λ^1, we choose a stepsize $t_K > 0$ at each iteration $K = 1, 2, \ldots$, and set

$$\lambda^{K+1} = \lambda^K - t_K g_{\lambda^K},$$

hoping to move closer to a global minimiser. Convergence of the sequence $\left(\phi(\lambda^k)\right)_{k=1,2,\ldots}$ to ϕ^* is guaranteed for suitable choice of stepsizes, but it is in general rather slow and depends heavily on the right choice of stepsizes, often a difficult issue.

[3] $\|\lambda - t g_\lambda - \lambda^*\|^2 = (\lambda - t g_\lambda - \lambda^*)^\top (\lambda - t g_\lambda - \lambda^*) = \|\lambda - \lambda^*\|^2 - 2t g_\lambda^\top (\lambda - \lambda^*) + t^2 g_\lambda^\top g_\lambda$ $< \|\lambda - \lambda^*\|^2$ for $0 < t < 2\frac{g_\lambda^\top (\lambda - \lambda^*)}{g_\lambda^\top g_\lambda}$. ($g_\lambda \neq 0$ because λ is not optimal.)

4.1.3 The cutting plane method of Cheney-Goldstein [13] and Kelley [31]

The subgradient method uses only the current subgradient at each iteration. In contrast, the cutting plane method makes use of the fact that after K iterations, the K subgradients provide us with a piecewise-linear underestimation ϕ^K of ϕ, called the current *cutting plane model* of ϕ:

$$\phi^K(\lambda) = \max_{k=1,\dots,K} \left\{ \phi\big(\lambda^k\big) + g_{\lambda^k}^\top \big(\lambda - \lambda^k\big) \right\},$$

Taking ϕ^K as an approximation of ϕ, we minimise ϕ^K instead of ϕ itself to obtain the next iterate by solving the linear programme

$$\begin{array}{ll} \text{minimise} & r \\ \text{subject to} & r \geqslant \phi\big(\lambda^k\big) + g_{\lambda^k}^\top \big(\lambda - \lambda^k\big) \text{ for all } k \in \{1,\dots,K\}, \\ & (\lambda, r) \in \mathbb{R}^m_{\geqslant 0} \times \mathbb{R}. \end{array} \tag{4.1}$$

Let $\big(r^{K+1}, \lambda^{K+1}\big)$ be an optimal solution of (4.1), then we set λ^{K+1} as the next iterate and call the oracle again. At any time, the optimum ϕ^* is bracketed by

$$r^{K+1} \leqslant \phi^* = \min_{\lambda \geqslant 0} \phi(\lambda) \leqslant \phi\big(\lambda^K\big),$$

which can be used as a stopping criterion. Since the sequence $\big(r^k\big)_{k=2,3,\dots}$ is nondecreasing and bounded, it converges to a limit. Under the important condition that the sequence $\big(\lambda^k\big)_{k=1,2,\dots}$ is bounded, we can prove that $\lim_{k\to\infty} r_k = \phi^*$.

4.1.4 Column generation

The subgradient and the cutting plane method are both algorithms designed for general convex optimisation problems and do not make use of the specific structure of the original problem. Nevertheless, the cutting plane method applied to the lagrangean dual of a mixed-integer programme allows for a clarifying interpretation as the dual version of *column generation*. Suppose we want to solve the linear programme

$$\max \left\{ c^\top x \mid Ax \leqslant b,\, x \in \operatorname{conv}(Q) \right\}, \tag{4.2}$$

a relaxation of the original mixed-integer programming problem P. If Q is such that $\operatorname{conv}(Q)$ is a bounded polyhedron with extreme points $\varkappa^1, \dots, \varkappa^\infty$, then all points $x \in \operatorname{conv}(Q)$ take the form of a convex combination $\sum_{k=1}^{\infty} \alpha_k \varkappa^k$ and

(4.2) can be reformulated as

$$\begin{array}{lll} \text{maximise} & c^\top \left(\sum_{k=1}^{\infty} \alpha_k \mathscr{x}^k \right) & \\ \text{subject to} & A \left(\sum_{k=1}^{\infty} \alpha_k \mathscr{x}^k \right) \leqslant b, & \\ & \sum_{k=1}^{\infty} \alpha_k = 1, & \\ & \alpha_k \geqslant 0 \quad \text{for all } k \in \{1, \ldots, \infty\}, & \end{array} \tag{MP}$$

the so-called *Dantzig-Wolfe decomposition* of (4.2) (see [15]), also referred to as the *master problem*. *MP* contains a large number of variables, but in a column generation approach most of them are handled implicitly and only a small number of columns $\{x^1, \ldots, x^K\} \subseteq \{\mathscr{x}^1, \ldots, \mathscr{x}^K\}$ is present explicitly in the so-called *restricted master problem*

$$\begin{array}{lll} \text{maximise} & c^\top \left(\sum_{k=1}^{K} \alpha_k x^k \right) & \\ \text{subject to} & A \left(\sum_{k=1}^{K} \alpha_k x^k \right) \leqslant b, & \\ & \sum_{k=1}^{K} \alpha_k = 1, & \\ & \alpha_k \geqslant 0 \quad \text{for all } k \in \{1, \ldots, K\}. & \end{array} \tag{RMP}$$

Solving *RMP* gives a primal solution as well as a vector $\lambda \in \mathbb{R}^m_{\geqslant 0}$ of optimal dual multipliers associated with the inequality constraints $A\big(\sum_{k=1}^{K} \alpha_k x^k\big) \leqslant b$.

If for all columns $k \in \{1, \ldots, \infty\}$ of *MP* the reduced cost $c^\top \mathscr{x}^k - \lambda^\top A \mathscr{x}^k$ is non-positive, then the optimal *RMP*-solution is also optimal for *MP*. Otherwise, solving the so-called *pricing problem*

$$\max_{k=1,\ldots,\infty} \left\{ c^\top \mathscr{x}^k - \lambda^\top A \mathscr{x}^k \right\} \tag{4.3}$$

results in one ore more columns with positive reduced cost, which we can add to the restricted master problem and reoptimise it.

To see the connection with the cutting plane method, form the (bi-)dual of the linear programme (4.1),

$$\begin{array}{lll} \text{maximise} & \sum_{k=1}^{K} \left[\phi(\lambda^k) - g_{\lambda^k}^\top \lambda^k \right] \alpha_k & \\ \text{subject to} & \sum_{k=1}^{K} -g_{\lambda^k} \alpha_k \leqslant 0, & \\ & \sum_{k=1}^{K} \alpha_k = 1, & \\ & \alpha_k \geqslant 0 \quad \text{for all } k \in \{1, \ldots, K\}. & \end{array} \tag{4.4}$$

We know that for all $k \in \{1, \ldots, K\}$,

$$\phi(\lambda^k) = \max_{x \in Q} \left\{ c^\top x + {\lambda^k}^\top (b - Ax) \right\} = c^\top x^k + {\lambda^k}^\top (b - Ax^k) \tag{4.5}$$

holds for some optimal $x^k \in Q$. If subgradients are computed according to Theorem 4.1, then $g_{\lambda^k} = b - Ax^k$. With this, (4.4) takes exactly the form of

the restricted master problem *RMP*. The evaluation of ϕ at an iterate λ^k as in (4.5) is recognised to be the pricing problem (4.3) from column generation, only shifted by an additive constant ${\lambda^k}^\top b$.

This connection also shows that column generation is another approach which can be used to solve the lagrangean dual. Because column generation is essentially the primal counterpart of the cutting plane method, it exhibits the same type of "instability" as the cutting plane method which will be explained in the next section. Here we only want to mention that, as for the cutting plane method, stabilised variants of column generation exist. For further details on column generation, see for instance the book of Wolsey [50, Chap. 11].

4.1.5 ACCPM – analytic centre cutting plane methods (Goffin et. al [21])

The standard cutting plane method as presented above is inherently instable: Programme (4.1) is always feasible, but by no means guaranteed to be bounded below and the optimal solution can be given by $r = -\infty$ and λ at infinity. Even worse, this behaviour is independent on how close the current iterate might be to the global minimum of ϕ. This may be seen as somewhat related to the technical condition of $(\lambda^k)_{k=0,1,\dots}$ being bounded, which is needed for a convergence proof of the cutting plane method.

In practise, this condition means the necessity to "stabilise" the algorithm, i.e. to limit the variation of the dual multipliers between the iterations. This is addressed by the ACCPM, a method first proposed by Goffin et al. [21], as follows: At iterate K, we define the polyhedron

$$\begin{aligned} P_K &= \left\{ (\lambda, r) \in \mathbb{R}^m_{\geqslant 0} \times \mathbb{R} \;\middle|\; \phi^K(\lambda) \leqslant r \leqslant \min\{\phi(\lambda^k) \mid k = 1, \dots, K\} \right\} \\ &= \left\{ (\lambda, r) \in \mathbb{R}^m_{\geqslant 0} \times \mathbb{R} \;\middle|\; \begin{array}{l} r \geqslant \phi(\lambda^k) + g_{\lambda^k}^\top (\lambda - \lambda^k) \text{ for all } k \in \{1, \dots, K\}, \\ r \leqslant \min\{\phi(\lambda^k) \mid k = 1, \dots, K\} \end{array} \right\}, \end{aligned}$$

which is assumed to be bounded, possibly by artificial means. The standard cutting plane method chooses a point (λ^{K+1}, r^{K+1}) on the boundary of P_K with minimal r^{K+1}. The idea of the ACCPM is to use a point in the interior of P_K instead, more precisely the *analytic centre* of P_K.

The analytic centre of a bounded polyhedron is defined as the unique point in the polyhedron maximising the product (or equivalently the sum of logs) of slacks with respect to the defining constraints. Then instead of (4.1) we maximise the barrier function

$$B(\lambda, r) = \log \left[\min_{k=1,\dots,K} \phi(\lambda^k) - r \right] + \sum_{k=1}^{K} \log \left[r - \phi(\lambda^k) - g_{\lambda^k}^\top (\lambda - \lambda^k) \right]$$

subject to $(\lambda, r) \in P_K$. From the unique optimum (λ^{K+1}, r^{K+1}), we obtain the next iterate λ^{K+1}.

The ACCPM can be viewed as a stabilisation of the standard cutting plane method in the following sense: We still compute a feasible solution of (4.1), but we are not as optimistic as to take the cutting plane model ϕ^K for a good approximation of the original function ϕ – hence, we do not solve (4.1) to optimality.

There are further variants like the Proximal-ACCPM, which incorporate the idea of bundle methods from the next section: Additionally, a quadratic stabilising term $\frac{1}{2t}\|\lambda - \tilde{\lambda}\|^2$, $\tilde{\lambda}$ a "proximal reference point", is subtracted from the barrier function, resulting in improved convergence. Further details can be found in the survey paper of Goffin and Vial [22].

4.1.6 Bundle methods (Lemaréchal [35])

The starting point for bundle methods is again the cutting plane method of Cheney-Goldstein and Kelley. In order to stabilise it, we now choose a stability centre $\tilde{\lambda} \in \mathbb{R}^m_{\geqslant 0}$ which we want the next iterate to be close to. Suppose, we are in the Kth iteration and one of the last iterates was chosen as the current stability centre. We add a euclidean penalty term $\frac{1}{2t}\|\lambda - \tilde{\lambda}\|^2 = \frac{1}{2t}(\lambda - \tilde{\lambda})^\top(\lambda - \tilde{\lambda})$ to the cutting plane model ϕ^K, where $t > 0$ is a "spring strength", possibly varying across the iterations.

Thus, instead of the linear programme (4.1) we solve the linear constrained quadratic programme

$$\begin{array}{ll} \text{minimise} & r + \frac{1}{2t}\|\lambda - \tilde{\lambda}\|^2 \\ \text{subject to} & r \geqslant \phi(\lambda^k) + g_{\lambda^k}^\top(\lambda - \lambda^k) \text{ for all } k \in \{1, \ldots, K\}, \\ & (\lambda, r) \in \mathbb{R}^m_{\geqslant 0} \times \mathbb{R}, \end{array} \tag{4.6}$$

always giving a finite minimiser (λ^{K+1}, r^{K+1}). To complete the iteration, we evaluate ϕ at λ^{K+1} and take care of the stability centre: We test if the actual decrease of the ϕ-values is large enough compared to the decrease in the cutting plane model, i.e. if

$$\phi(\tilde{\lambda}) - \phi(\lambda^{K+1}) \geqslant \kappa[\phi(\tilde{\lambda}) - \phi^K(\lambda^{K+1})] = \kappa[\phi^K(\tilde{\lambda}) - \phi^K(\lambda^{K+1})]$$

holds for some fixed tolerance $\kappa \in (0, 1)$. In this case, we treat the current cutting plane model as sufficiently good approximation of ϕ and perform a *descent step* by setting λ^{K+1} to be the new stability centre, $\tilde{\lambda} = \lambda^{K+1}$. Otherwise, the stability centre remains unchanged and only the cutting plane model is enriched, which is called a *null step*.

Stopping criteria and convergence proofs are at hand: The sequence given by the objective values $\phi(\tilde{\lambda})$ of the stability centres converges to the global minimum ϕ^*, and if a global minimiser for ϕ exists, then also the sequence of stability centres converges to such a minimising point. Initially, a bundle algorithm of this form was described by Lemaréchal [35]. For a thorough treatment of the details we refer to [26, Chap. XV].

4.2 Computational comparison of LP-relaxation and lagrangean dual

4.2.1 Computational experiments with the LP-relaxation

To evaluate the performance of solving the LP-relaxations, the author conducted experiments using the state-of-the-art software package CPLEX 11.0 [29], which provides implementations of primal and dual simplex algorithms as well as an interior-point barrier solver for linear programmes. The main goal was to determine the fastest way for solving the LP-relaxation in practise using any of the solvers and formulations from Section 3.1, in order to compare this with the performance of the lagrangean approach.

The experiments yielded two main results: First, the barrier solver (without crossover to a basic solution) clearly outperformed both simplex variants. As an example, using the formulation *x-D-LP*, the barrier solver computed the optimal solution to the three "marvin"-instances within 11, 27 and 260 seconds, respectively, whereas the primal simplex algorithm took 34, 97 and 506 seconds. The dual simplex was even slower by a factor of 2 to 10. Even more importantly, the largest instance wa-125-96821 could not be solved at all within a time limit of 1 hour by the primal or dual simplex algorithm. In contrast, the barrier solver was able to compute an optimal solution within 325 seconds. Even when performing a subsequent crossover to a basic solution, the barrier solver was by far the fastest algorithm. (For the largest instance wa-125-96821, a crossover could not be performed due to memory limits.)

Second, among the reduced LP-relaxations, the formulation *x-D-LP* performed better than *y-D-LP* on all but the largest instance. For wa-125-96821, using the barrier solver, *y-D-LP* was solved within 314 seconds compared to 324 seconds for *x-D-LP*. On all other instances, *y-D-LP* was slower by a factor between 1.1 and 2.4. Moreover, the experiments indicated that *y-D-LP* is numerically more difficult to solve using the barrier solver: On 4 out of the 25 "ca"-scenarios, the barrier solver terminated only with a near- optimal solution of *y-D-LP* due to numerical difficulties, necessitating a crossover to a basic solution. In contrast, *x-D-LP* could be solved to optimality on all instances.

All in all, the experiments established solving the reduced formulation *x-D-LP* with the barrier solver as the fastest method on the experimental test set. For this, the results can be seen in Table 4.1 and are used for comparison with the lagrangean approach discussed next.

4.2.2 Computational experiments with the lagrangean dual

Which of the methods outlined in Section 4.1 is "best" for solving the lagrangean dual cannot be decided generally, but will depend on the problem

under consideration. Briant et al. [7] compare the classical column generation approach (respectively the cutting plane method) without stabilisation on the one hand, and the bundle algorithm on the other hand in experiments on prominent combinatorial optimisation problems. Their results do not show one approach clearly outperforming the other. Nevertheless, they indicate that stabilisation is called for, especially as the dimension of the dual space grows. It also becomes clear that the bundle algorithm more often works as an "out-of-the-box" method, while column generation sometimes requires many considerations to make it work well.

To evaluate the lagrangean approach presented in Section 3.4, the author decided to conduct experiments with a bundle algorithm, also due to a good implementation by Helmberg [25] readily available. It provides a scaling heuristic and the possibility for "active bounds fixing", i.e. for fixing multipliers temporarily to the stability centre value if their (nonnegativity) bounds are strongly active. Applying both features helped to improved the performance of the bundle algorithm in the experiments.

Solving the lagrangean subproblem The lagrangean relaxation subproblem was solved using the minimum cut approach from Section 3.4.3. For computing the minimum a-b-cuts, a relatively new algorithm for the maximum flow problem in networks was used, the pseudoflow algorithm of Hochbaum [27]. It is based on "normalised trees" and pseudoflows, i.e. "flows" which violate the flow balance constraints arbitrarily, but satisfy the capacity constraints. Although its worst-time complexity is – with $\mathcal{O}(mn \log n)$ for a graph on n nodes and m arcs – theoretically worse than the push-relabel-algorithm of Goldberg and Tarjan, it performed better in a recent computational study of Chandran and Hochbaum [11]. The implementation of Chandran and Hochbaum [12] could be used in the experiments and showed good performance. It is only necessary to run the first phase of the pseudoflow algorithm, which computes a minimum cut, while the second phase for recovering a maximum flow can be skipped.

Warmstarting One reason for choosing the pseudoflow algorithm was that it can be initialised with any feasible pseudoflow and hence allows for an internal warmstart after the capacities have changed. This is exactly the case here: During the run of the bundle algorithm, the lagrangean subproblem and thus the minimum cut problem must be solved numerous times on the same network with capacities changed according to the dual multipliers – there is reason to hope that warmstarting from a pseudoflow optimal for the previous assignment of capacities might improve the running time. Unfortunately, the implementation of Chandran and Hochbaum [12] did not provide this feature.

A general purpose warmstart method for any maximum flow algorithm is described in [11] using the residual graph. Experiments with this approach proved worse, however, than solving each lagrangean subproblem from scratch. This might be expected, since the number of arcs is doubled when switching to the residual graph and further excess and deficit arcs need to be added. The time for computing each minimum cut turned out to increase by a factor of at least 2.

In most instances, however, solving the minimum cut problems did not consume most of the time, anyway. Computing the values of the z-variables via (3.16) has complexity $\mathcal{O}(NT)$ and is time-consuming especially for the "ca"-instances and for instance wa-125-96821. For instances marvin-115-8513, marvin-296-8513 and marvin-1038-8513, the minimum cut computations consumed approximately 20%, 47% respectively 79% of the time needed to solve the lagrangean subproblems. The more aggregates, the more nodes and arcs are in the network D, hence computing the minimum a-b-cut takes longer. For the "ca"-instances, minimum cut computations consumed less than 10%, and for wa-125-96821, only about 3% of the time for solving the subproblem was spent with computing the minimum cut. All in all, solving the subproblems consumed the vast majority of the bundle algorithm's running time.

A last remark on solving the subproblems: The infinite capacity must be replaced by an upper bound on the minimum cut capacity, which was computed as $\sum_{k=1}^{K} c(e_{k,T+1})$, compare Section 3.4.3.

Impact of the maximum bundle size As presented in Section 4.1, the cutting plane model is continuously enriched. In practise, this is usually suboptimal though, since solving (4.6) becomes more and more difficult as K grows and the information from subgradients in early iterations might even be redundant for the cutting plane model in the neighbourhood of the current iterate. Thus, the "size of the bundle", i.e. the number of subgradients kept is limited and old subgradients are removed if necessary. Figures 4.1, 4.2 and 4.3 show how the running time depends on the maximum bundle size ranging from 10 to 100. As can be seen, especially too small a maximum bundle size can lead to overly long running times. Choosing the maximum bundle size larger and larger, a slight increase is recognisable for the instances marvin-115-8513, marvin-296-8513 and the "ca"-instances.

For comparison with the LP-relaxation, taking the minimal running time is certainly not representative – the dependence on the maximum bundle sizes is rather volatile, so one can not guarantee to find the "optimal" value of the maximum bundle size. Rather it seems fair to select a range of maximum bundle sizes where the algorithm works well and take the worst value found in this range. Hence, Table 4.1 reports a min-max running time: From the running times for 10 consecutive maximum bundle sizes take the highest one

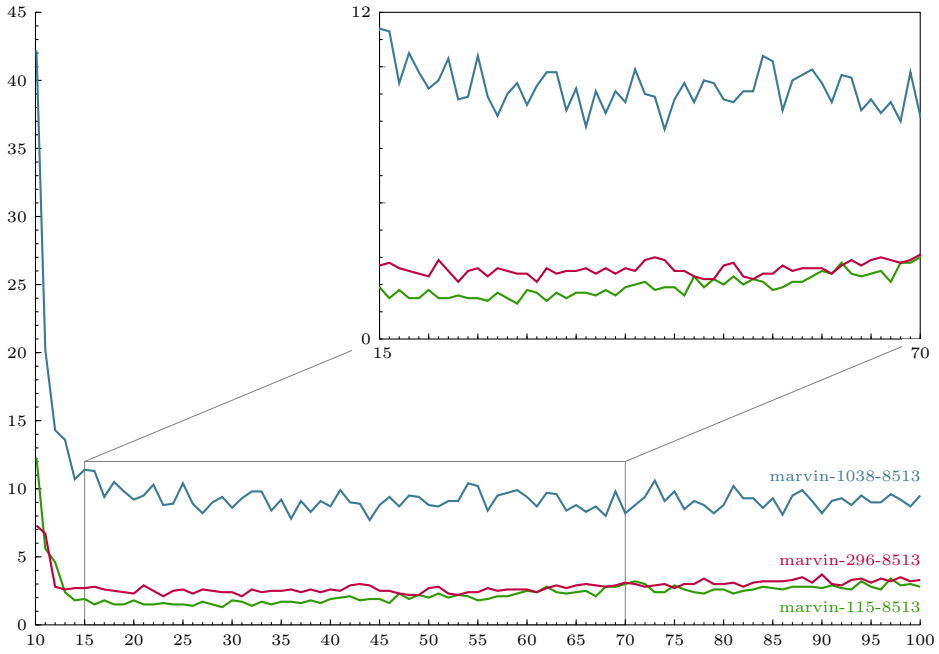

Figure 4.1: Running time (in seconds) of the bundle algorithm vs. maximum bundle size for the three “marvin”-instances

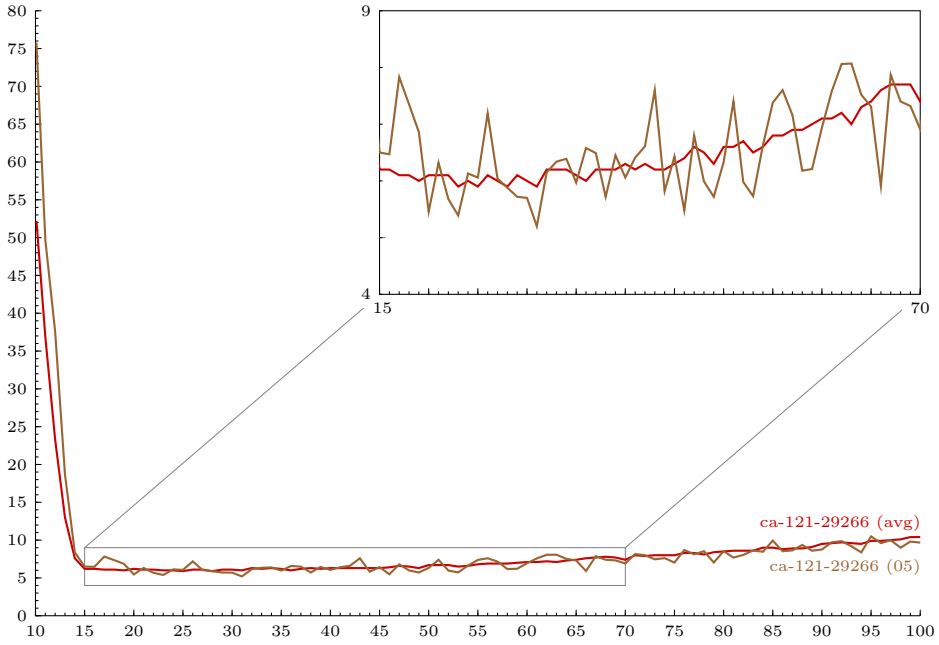

Figure 4.2: Running time (in seconds) of the bundle algorithm vs. maximum bundle size for instance ca-121-29266, scenario 5, and for all 25 “ca”-scenarios as average running time

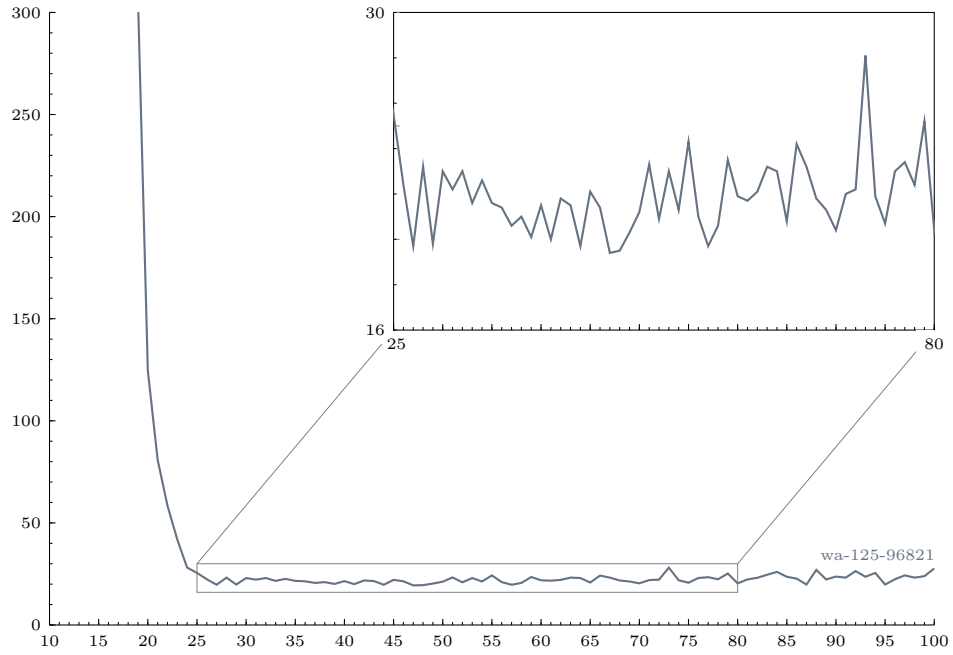

Figure 4.3: Running time (in seconds) of the bundle algorithm vs. maximum bundle size for instance wa-125-96821

The running times increase drastically for small maximum bundle sizes, up to 1213 seconds for a maximum bundle size of 10.

and from all these "local worst-case running times" choose the smallest one. The objective values are reported the same way.

The ConicBundle software [25] of Helmberg also provides a variant of the bundle algorithm where the subgradients in the bundle are aggregated, so effectively no bundle is used at all. This variant performed poorly, however. No better relative precision on the dual bound than 10^{-3} could be achieved and running times for this were increased by factors of 10 and more compared to the values from Table 4.1.

Multiplier deviation Furthermore, the value of optimal Lagrange multipliers are of interest, since the next chapter will devise an aggregation method based on them. Since the lagrangean dual is only solved to near-optimality, the final multipliers given by the bundle algorithm are not necessarily optimal. We report their deviation from the optimal multipliers given by the LP-relaxation: In the last column, we report the relative euclidean distance $\|(\mu^{D\text{-}LD}, \pi^{D\text{-}LD}) - (\mu^{D\text{-}LP}, \pi^{D\text{-}LP})\|/\|(\mu^{D\text{-}LP}, \pi^{D\text{-}LP})\|$, while the column "max. rel. mul. dev." gives the maximum relative deviation of a single multiplier, $\max_{t=1,\ldots,T} \left\{|\mu_t^{D\text{-}LD} - \mu_t^{D\text{-}LP}|/|\mu_t^{D\text{-}LP}|, |\pi_t^{D\text{-}LD} - \pi_t^{D\text{-}LP}|/|\pi_t^{D\text{-}LP}|\right\}$. (To account for zero denominators, we added 10^{-6} in the denominators.) For these two columns, not the min-max values as above are reported, but the largest, i.e. worst values over all maximum bundle sizes.

Problem instance	LP-relaxation		Lagrangean dual					
	time [s]	obj. val. [10^8]	time [s]	obj. val. [10^8]	rel. time	rel. obj. val.	max. rel. mul. dev.	rel. euc. mul. dev.
marvin-115-8513	11.2	7.230097	1.8	7.230097	0.158787	1.000000	0.008098	0.000263
marvin-296-8513	27.2	7.276624	2.6	7.276631	0.094993	1.000001	0.150439	0.003054
marvin-1038-8513	259.5	7.306851	9.5	7.306854	0.036415	1.000001	0.032616	0.000609
ca-121-29266 (avg)	34.9	59.898146	6.1	59.898190	0.175164	1.000001	0.028502	0.001276
ca-121-29266 (05)	64.7	60.899745	6.6	60.899800	0.101763	1.000001	0.078292	0.005544
wa-125-96821	324.4	0.508814	21.8	0.508815	0.067139	1.000001	0.005166	0.000223

Table 4.1: Computational results for solving LP-relaxation and lagrangean dual
For the LP-relaxation, formulation *x-D-LP* from Section 3.1 was solved using the CPLEX barrier solver [29]. Crossover to a basic solution was not performed.
The minimum cut computations for the lagrangean relaxation subproblems were solved using the Pseudoflow Solver [12] (highest label variant) of Chandran and Hochbaum. The lagrangean dual itself was solved using the ConicBundle software [25] of Helmberg with a relative precision of 10^{-9} on the objective value. However, the bundle algorithm always stopped before reaching this precision due to numerical limits of the interplay between subproblem and quadratic programming solver. The effective precision reached (relative to the objective value from the LP-relaxation) can be seen in column "rel. obj. val." to be very small: at most 10^{-6}.
For the "ca"-instances, we report the arithmetic mean over all 25 scenarios. The values for scenario 5, the "hardest" scenario with respect to the running time for solving the LP-relaxation, are stated separately.

4.2.3 Conclusion

Using the bundle method, the lagrangean dual yielded practically the same bound as the LP-relaxation, i.e. within a relative precision of 10^{-6}. With respect to computing time, the lagrangean approach clearly outperformed solving the LP-relaxation directly, even using the fastest method available, i.e. solving *x-D-LP* using the barrier solver. For the standard LP-relaxation *D-LP* and the simplex variants, the result would be even clearer. In particular, for the largest problem instances wa-125-96821 (with the largest number of continuous variables) and marvin-1038-8513 (with the largest number of binary variables), where computing the LP-relaxation took more than 4 respectively 5 minutes, the lagrangean dual could be solved within 10 and 22 seconds.

The maximum relative deviation of the single multipliers, however, seems unsatisfactory large at first sight, in particular for instance marvin-296-8513. First, these are worst case values over all bundle sizes, the average is significantly lower. Second, the high deviations always stem from the last time periods, when the multipliers are almost negligible. If we would disregard these time periods, all the values would drop by a factor between 5 and 10, e.g. to a value of 0.012 for marvin-296-8513. Third, it is not even guaranteed that the optimal multipliers have to be unique, so deviation from one optimal multiplier vector does not necessarily indicate suboptimality.

As a last remark, the dual multipliers for the mining constraints were zero in all LP- and LD-solutions, i.e. the mining constraints are never tight.

The processing constraints seem to be the main restrictions on the mining operations.

Chapter 5

Aggregation of processing decisions

The integration of cutoff grade optimisation in the new OPMPSP-formulation *D-MIP* introduced in Chapter 2 first appears to come at the cost of the comparatively large additional number of continuous variables. However, the results in Section 3.2 showed that many of them take the same value in an optimal solution. This chapter exploits this insight by determining variables with similar values in an optimal solution in a heuristic manner based on the LP-relaxation of *D-MIP*, and subsequently aggregating such variables to reduce the size of the programme.

Section 5.1 briefly outlines the concept of aggregation in large-scale optimisation, especially the method of column aggregation for linear programmes. Based on this, Section 5.2 introduces the general idea of *binnings*, thus employing column aggregation to the OPMPSP-formulation *D-MIP*. Section 5.3 proposes various binnings based on primal and dual optimal solutions of the LP-relaxation and compares them computationally. Finally, Section 5.4 explains how a solution to the original *D-MIP* can be reconstructed easily from a solution to the aggregated problem and evaluates this disaggregation method giving experimental results.

5.1 Aggregation in large-scale optimisation

Modelling real world systems inherently involves abstraction and approximation. Generally speaking, more accurate modelling of a problem will usually improve the quality of the final solution, but also increase the difficulty of finding it. The question to what level of detail a problem can be modelled and solved, therefore, becomes a crucial one. It is dealt with in two different, in some sense complementary ways: *aggregate modelling* and *model aggregation*. The first is concerned with the aggregate level of detail employed in a model,

while the latter focuses on solving a given, large optimisation problem by replacing it with a smaller more tractable model. As an example, the idea to model the precedence constraints in the OPMPSP on aggregate instead of on block level may be understood as aggregate modelling technique. In contrast, this section will deal with a model aggregation technique to reduce the size of the OPMPSP-formulation *D-MIP*.

Rogers et al. [46] develop a general framework for model aggregation and disaggregation methodology, which they describe as a set of methods for solving optimisation problems by

- combining data,
- solving an auxiliary model (or models) which is reduced in size and/or complexity relative to the original model and
- analysing the results of the auxiliary model in terms of the original model.

In Section 5.1.1, we outline how this general idea is applied in linear programming by aggregating variables. Section 5.2 introduces the concept of binnings, a column aggregation approach for the OPMPSP-formulation *D-MIP*. In Section 5.3 we propose several specific types of binning and compare them computationally.

5.1.1 Column aggregation in linear programming

We largely follow the exposition in Litvinchev and Tsurkov [38, Chap. 1]. Consider a problem

$$\begin{aligned} z = \max \quad & c^\top x \\ \text{subject to} \quad & Ax \leqslant b, \\ & x \geqslant 0, \end{aligned} \tag{5.1}$$

where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. (5.1) is referred to as the *original problem* and assumed to have a finite optimal solution, which cannot be computed directly due to a large number of variables n. Let $\mathcal{S}$ be a partition of the set of column indices $\{1, \dots, n\}$ into subsets S_k, $k \in \{1, \dots, K\}$, i.e. $S_k \cap S_l = \emptyset$ for any $k \neq l$ and $\bigcup_{k=1}^{K} S_k = \{1, \dots, n\}$. For each $k \in \{1, \dots, K\}$, choose a (nonnegative non-zero) weighting vector $g^k \in \mathbb{R}^n_{\geqslant 0}$, $g^k \neq 0$, with $g^k_i = 0$ for all $i \notin S_k$. Set

$$\hat{A}^k = Ag^k,\ \hat{c}^k = c^\top g^k,$$

i.e. $\hat{A}^k$ is an m-vector ($\hat{c}^k$ a real value) obtained as non-zero linear combination of the columns of A (of the entries of c) with indices in S_k.

Define $\hat{A} = \left(\hat{A}^1, \ldots, \hat{A}^K\right) \in \mathbb{R}^{m \times K}$ and $\hat{c} = \left(\hat{c}^1, \ldots, \hat{c}^K\right)^\top \in \mathbb{R}^K$. Then the programme

$$\begin{aligned} \hat{z} = \max \quad & \hat{c}^\top X \\ \text{subject to} \quad & \hat{A}X \leqslant b, \\ & X \geqslant 0, \end{aligned} \tag{5.2}$$

is called the (column or variable) *aggregated problem* of (5.1). $X \in \mathbb{R}^K$ is a vector of aggregated decision variables. The matrix $g = \left(g^1, \ldots, g^K\right) \in \mathbb{R}^{n \times K}_{\geqslant 0}$ is referred to as the *weighting matrix* of the aggregation.[1] For a given original problem, (5.2) is fully determined by the pair $(\mathcal{S}, g)$.

Note that (5.2) should be significantly reduced in size compared to the original problem such that an optimal solution can be computed. Also, aggregation can result in an infeasible problem even if the original problem has a feasible solution. We exclude this case:

Assumption 5.1 $(\mathcal{S}, g)$ *is chosen such that* (5.2) *is feasible.*

With the weighting matrix g, we can write $\hat{c}$ and $\hat{A}$ as $g^\top c$ and Ag, respectively, and so (5.2) is the same as

$$\begin{aligned} \hat{z} = \max \quad & c^\top g\, X \\ \text{subject to} \quad & Ag\, X \leqslant b, \\ & X \geqslant 0. \end{aligned} \tag{5.3}$$

Since all entries of g are nonnegative and each column contains at least one non-zero entry, $X \geqslant 0$ if and only if $gX \geqslant 0$. This shows that (5.3) is exactly the problem obtained by the linear substitution $x = gX$ in (5.1), and for any feasible solution X of (5.2), gX is feasible for (5.1). It follows that

$$\hat{z} = \max\left\{c^\top gX \mid AgX \leqslant b,\, X \in \mathbb{R}^K_{\geqslant 0}\right\} \leqslant \max\left\{c^\top x \mid Ax \leqslant b,\, x \in \mathbb{R}^n_{\geqslant 0}\right\} = z,$$

proving the intuition that aggregating variables yields a lower objective value. Also, we may interpret (5.2) as the problem obtained from (5.1) by additionally constraining the feasible region to the set $\{x = gX \mid X \in \mathbb{R}^K_{\geqslant 0}\}$.

Remark 5.2 Zipkin [52] uses the normalising condition

$$g \in G = \left\{ g \in \mathbb{R}^{n \times K}_{\geqslant 0} \;\middle|\; \sum_{i=1}^{n} g_i^k = \sum_{i \in S_k} g_i^k = 1 \text{ for all } k \in \{1, \ldots, K\} \right\}$$

for the weighting matrix, which is also adopted by Litvinchev and Tsurkov [38].

[1] One might imagine, without loss of generality, the case where the variables in (5.1) are ordered such that aggregation occurs in increasing order, i.e. S_1 contains $1, \ldots, |S_1|$, $S_2 = \{|S_1| + 1, \ldots, |S_1| + |S_2|\}$, etc., where g has a block-diagonal-like structure.

For $x = gX$ this yields the property that

$$X_k = \sum_{i \in S_k} g_i^k X_k = \sum_{i \in S_k} x_i,$$

i.e. an aggregated variable is equal to the sum of all variables of the aggregate. Note that this normalisation poses no restrictions on the possibilities of aggregations, since the set $\left\{x = gX \mid X \in \mathbb{R}^K_{\geqslant 0}\right\}$ is invariant when we multiply columns of g by positive scalars. This property is also unnecessary for the theory (e.g. proofs for error bounds do not involve the normalisation) and in some cases it may be convenient to consider non-normalised weighting matrices where $\sum_{i \in S_k} g_i^k \neq 1$ for some aggregates $k \in \{1, \ldots, K\}$.

5.1.2 Standard disaggregation methods

Solving the aggregated problem is only an auxiliary step: Once a solution $\hat{X}$ of (5.2) is at hand, it has to be interpreted in terms of the original problem (5.1). A simple way to deduce a feasible solution for the original problem is to apply *fixed-weight disaggregation*, i.e.

$$\hat{x}_i = g_i^k \hat{X}_k \text{ for all } i \in S_k, k \in \{1, \ldots, K\}.$$

We noted that variable aggregation is essentially the linear substitution $x = gX$. In this sense, the fixed-weight disaggregation transforms $\hat{X}$ back to $\hat{x} = g\hat{X}$. From the reformulation of (5.2) in (5.3), it is clear that the fixed-weight disaggregated solution $\hat{x}$ is feasible for (5.1) and $c\hat{x} = \hat{z}$.

In general, we can obtain a better objective solution value, however, when performing so-called *optimal disaggregation* of $\hat{X}$: For all $k \in \{1, \ldots, K\}$, solve the subproblems

$$\begin{aligned} \max \quad & c^\top x^k \\ \text{subject to} \quad & Ax^k \leqslant \hat{A}^k \hat{X}_k, \\ & x_i^k = 0 \quad \text{for all } i \notin S_k, \\ & x^k \geqslant 0, \end{aligned} \tag{5.4}$$

yielding optimal solutions $\tilde{x}^k$. Combining these via $\tilde{x} = \sum_{k=1}^K \tilde{x}^k$ gives a feasible solution to (5.1), since $\tilde{x} \geqslant 0$ and

$$A\tilde{x} = \sum_{k=1}^K A\tilde{x}^k \leqslant \sum_{k=1}^K \hat{A}^k \hat{X}_k = \hat{A}\hat{X} \leqslant b.$$

Moreover, the vector $g^k \hat{X}_k$ is feasible for (5.4). So first an optimal solution always exists for each of the subproblems, and second

$$c^\top \tilde{x} = \sum_{k=1}^K c^\top \tilde{x}^k \geqslant \sum_{k=1}^K c^\top g^k \hat{X}_k = c^\top \left(\sum_{k=1}^K g^k \hat{X}_k \right) = c^\top g \hat{X} = c^\top \hat{x},$$

i.e. $\tilde{x}$ is at least as good a solution to (5.1) as $\hat{x}$.

The quality of a feasible solution of (5.1) obtained this way naturally depends on the problem at hand and the specific aggregation used. Given a linear programme (5.1) and a weighting matrix, provable error bounds on the loss of accuracy caused by aggregation can be computed, and thus the optimal objective value of (5.1) can be bounded. There are two general types of error bounds: While *a priori error bounds* can be computed merely on the basis of (5.1) and the weighting matrix, for *a posteriori error bounds*, an optimal solution to the aggregated problem (5.2) must be at hand. For specific error bounds and further details we refer to Litvinchev and Tsurkov [38].

5.2 Binnings – a column aggregation scheme for the open pit mining production scheduling problem

The OPMPSP-formulation *D-MIP* models the mining operations more accurately than other formulations by distinguishing between mining and processing. This comes at the price of a larger, more difficult to solve programme: While the mixed-integer programming formulation (2.4), for instance, consists of $2KT$ variables, *D-MIP* has NT additional continuous variables, and thus $2KT+NT$ variables altogether. The problem instance wa-125-96821 described in Section 1.3 and used in the experiments shows how the number of blocks N can easily be one or two orders of magnitude greater than the number of aggregates K. Hence, *D-MIP* can be significantly more difficult to solve than the programmes without integrated cutoff grade optimisation.

To alleviate the computational burden springing from the additional z-variables, Section 3.2 motivates a column aggregation approach. With the existence of split ratios, Proposition 3.4 proves that in any optimal solution of *D-MIP*, for each aggregate $\mathcal{K}_k$, $k \in \{1, \dots, K\}$, and time period $t \in \{1, \dots, T\}$, the variables $z_{i,t}$, $i \in \mathcal{K}_k$, take at most three distinct values: 0, $y_{k,t}$ and possibly a value in between for blocks with $\frac{p_{i,t}}{a_i}$ equal to the split ratio σ_t. With the values of split ratios at hand, we could use this to substitute the z-variables for blocks of almost all oregrades.

With Proposition 3.9, however, we showed that split ratios are NP-hard to compute. Thus determining their exact value is not efficiently possible unless P is NP, so a less strict approach seems to be called for. The following concept proves to be helpful:

Definition 5.3 *Consider an instance of D-MIP with aggregates $\mathcal{K}_1, \dots, \mathcal{K}_K$. A set of blocks $B \subseteq \mathcal{K}_k$ is called a* bin *for aggregate $\mathcal{K}_k$, $k \in \{1, \dots, K\}$. A* binning *is a collection $\mathcal{B} = \big(\mathcal{B}_{k,t} \,|\, k = 1, \dots, K,\, t = 1, \dots, T\big)$ where each $\mathcal{B}_{k,t}$ is a set of pairwise disjoint bins partitioning $\mathcal{K}_k$, i.e.*

$$\mathcal{K}_k = \dot{\bigcup_{B \in \mathcal{B}_{k,t}}} B$$

for all $k \in \{1,\dots,K\}$ *and* $t \in \{1,\dots,T\}$. *Write* $\mathcal{B}_{k,t} = \{B_1,\dots,B_{n_{k,t}}\}$ *with* $n_{k,t} = |\mathcal{B}_{k,t}|$.

First some more notation: Each block $i \in \mathcal{N}$ is contained in exactly one aggregate: denote its index by $k(i)$. Furthermore, given a binning, then for fixed time period $t \in \{1,\dots,T\}$, each block $i \in \mathcal{N}$ is contained in exactly one bin – denote this bin number by $b(i,t)$. Now any binning $\mathcal{B}$ partitions the set of z-variables in *D-MIP*, more precisely their index set:

$$\{(i,t) \mid i \in \mathcal{N}, t = 1,\dots,T\} = \dot{\bigcup_{\substack{k=1,\dots,K,\\ t=1,\dots,T,\\ b=1,\dots,n_{k,t}}}} \{(i,t) \mid i \in B_b\}\,.$$

Suppose the binning can be chosen such that any of the parts above comprises the indices of z-variables with similar value in an optimal solution. In this case, it is reasonable to aggregate the z-variables of *D-MIP* according to this partition. For each $k \in \{1,\dots,K\}$ and $t \in \{1,\dots,T\}$, introduce new aggregate variables

$$Z_{k,t,1},\dots,Z_{k,t,n_{k,t}}$$

and aggregate according to

$$z_{i,t} = Z_{k(i),t,b(i,t)} \tag{5.5}$$

for all blocks $i \in \mathcal{N}$ and time period $t \in \{1,\dots,T\}$. The corresponding weighting matrix has all entries from $\{0,1\}$ and does not satisfy Zipkin's normalising condition from Remark 5.2. Substituting (5.5) in *D-MIP* gives the aggregate programme *B-MIP*($\mathcal{B}$),

maximise

$$\sum_{k=1}^{K}\sum_{t=1}^{T}\left[-\bar{c}_{k,t}\,y_{k,t} \;+\; \sum_{b=1}^{n_{k,t}}\big(\Sigma_{i\in B_b}\,p_{i,t}\big)Z_{k,t,b}\right] \tag{5.6}$$

subject to

$$x_{k,t-1} - x_{k,t} \leqslant 0 \qquad \text{for all } k \in \{1,\dots,K\},\ t \in \{2,\dots,T\}, \tag{5.6a}$$

$$\sum_{s=1}^{t} y_{k,s} - x_{k,t} \leqslant 0 \qquad \text{for all } k \in \{1,\dots,K\},\ t \in \{1,\dots,T\}, \tag{5.6b}$$

$$x_{k,t} - \sum_{s=1}^{t} y_{\ell,s} \leqslant 0 \qquad \text{for all } k \in \{1,\dots,K\},\ \ell \in \bar{\mathcal{P}}(k),\ t \in \{1,\dots,T\}, \tag{5.6c}$$

$$Z_{k,t,b} - y_{k,t} \leqslant 0 \qquad \text{for all } k \in \{1,\dots,K\},\ t \in \{1,\dots,T\},\ b \in \{1,\dots,n_{k,t}\}, \tag{5.6d}$$

$$\sum_{k=1}^{K} \bar{a}_k y_{k,t} \leqslant U_t^{\mathrm{m}} \quad \text{for all } t \in \{1,\dots,T\}, \tag{5.6e}$$

$$\sum_{\substack{k=1,\dots,K \\ b=1,\dots,n_{k,t}}} \left(\Sigma_{i\in B_b} a_i\right) Z_{k,t,b} \leqslant U_t^{\mathrm{p}} \quad \text{for all } t \in \{1,\dots,T\}, \tag{5.6f}$$

$$x_{k,t} \in \{0,1\} \quad \text{for all } k \in \{1,\dots,K\},\ t \in \{1,\dots,T\}, \tag{5.6g}$$

$$0 \leqslant y_{k,t} \leqslant 1 \quad \text{for all } k \in \{1,\dots,K\},\ t \in \{1,\dots,T\}, \tag{5.6h}$$

$$0 \leqslant Z_{k,t,b} \leqslant 1 \quad \text{for all } k \in \{1,\dots,K\},\ t \in \{1,\dots,T\},\ b \in \{1,\dots,n_{k,t}\}. \tag{5.6i}$$

Two extreme choices of binnings are the "block binning" $\mathcal{B}^{\text{block}}$, where bins are singletons consisting of one block each, and the "aggregate binning" $\mathcal{B}^{\text{agg}}$, where each aggregate forms only one bin. $\mathcal{B}^{\text{block}}$ is the most refined binning – *B-MIP*$(\mathcal{B}^{\text{block}})$ is identical to the original *D-MIP* – and gives the best optimal objective value achievable by any *B-MIP*$(\mathcal{B})$. Binning $\mathcal{B}^{\text{agg}}$ is the coarsest binning with the least number of Z-variables and yields the lowest optimal objective value. The optimal objective value for any binning $\mathcal{B}$ will lie in between:

$$f^*_{\textit{B-MIP}(\mathcal{B}^{\text{agg}})} \leqslant f^*_{\textit{B-MIP}(\mathcal{B})} \leqslant f^*_{\textit{B-MIP}(\mathcal{B}^{\text{block}})} = f^*_{\textit{D-MIP}}.$$

Our goal is to determine binnings with as few bins as possible in order to reduce the number of variables, while the objective value remains close to $f^*_{\textit{D-MIP}}$, the optimal objective value of *D-MIP*.

Remark 5.4 The structure of the aggregated problem is essentially the same as for the original *D-MIP*. Looking at one fixed time period, *B-MIP*$(\mathcal{B})$ can be imagined as *D-MIP* with the same aggregate, but lower block resolution. This "block resolution", however, is allowed to vary over time, while it is constant in *D-MIP*. Although perhaps counterintuitive at first, this variability is crucial. In general, cutoff grades change over time and thus for different time periods different blocks will have comparable processing priority. Therefore, applying the same aggregation for each time period is in general suboptimal.

It is straightforward to check that due to the similar structure, all of the results for *D-MIP* from Chapter 3 hold for *B-MIP*$(\mathcal{B})$ correspondingly.

5.3 LP-based binnings

Determing the exact "processing priority" of each block, i.e. computing split ratios for *D-MIP*, is NP-hard, see Proposition 3.9. The idea of this chapter is to use the LP-relaxation as an approximation. Although this is only a heuristic approach, Section 3.2 supports it to some extent: The values of z-variables

in optimal LP-solutions also follow split ratios, although their values will in general differ from the ones for *D-MIP*.

5.3.1 Binnings based on primal LP-solutions

One idea is to create binnings by inspecting the values of z-variables in an optimal (primal) solution (x^*, y^*, z^*) of *D-LP*. Different z-values indicate different processing decisions and the corresponding blocks should be in different bins. Precisely, construct a *variable primal LP-binning* $\mathcal{B}^{\text{prim-var}}$ as follows:

For each $k \in \{1, \dots, K\}$ and $t \in \{1, \dots, T\}$, let $s_1, \dots, s_{n_{k,t}}$ be the distinct values in $\{z^*_{i,t} \,|\, i \in \mathcal{K}_k\}$, $n_{k,t} = \big|\{z^*_{i,t} \,|\, i \in \mathcal{K}_k\}\big|$. Then, define $n_{k,t}$ bins

$$B_b = \{i \in \mathcal{N} \,|\, z^*_{i,t} = s_b\},$$

one for each z-value s_b, $b \in \{1, \dots, n_{k,t}\}$, yielding $\mathcal{B}^{\text{prim-var}}_{k,t} = \{B_1, \dots, B_{n_{k,t}}\}$. All in all, this gives the binning

$$\mathcal{B}^{\text{prim-var}} = \big(\mathcal{B}^{\text{prim-var}}_{k,t} \,\big|\, k = 1, \dots, K,\, t = 1, \dots, T\,\big).$$

This type of binning exhibits the following shortcoming: For each aggregate $\mathcal{K}_k$, $k \in \{1, \dots, K\}$, the value $y^*_{k,t}$ will be zero for the majority of the time periods $t \in \{1, \dots, T\}$. Then the z-values do not provide any information about the processing priority of the blocks and $\mathcal{B}^{\text{prim-var}}_{k,t}$ consists of only one bin, $\mathcal{K}_k$ itself.

One possibility to overcome this is to consider only the time period during which most of an aggregate is mined, motivated by viewing this time period as the most indicative one. Precisely, for each $k \in \{1, \dots, K\}$, let $t^*(k)$ be the (earliest) time period with $y^*_{k,t^*(k)} = \max_{t=1,\dots,T} y^*_{k,t}$. In a similar manner as above, let $n_k = \big|\{z^*_{i,t^*(k)} \,|\, i \in \mathcal{K}_k\}\big|$ and let $s_1, \dots, s_{n_k}$ be the distinct values in the set $\{z^*_{i,t^*(k)} \,|\, i \in \mathcal{K}_k\}$. Define n_k bins

$$B_b = \{i \in \mathcal{N} \,|\, z^*_{i,t^*(k)} = s_b\},$$

for all $b \in \{1, \dots, n_k\}$, and with this $\mathcal{B}^{\text{prim-cons}}_{k,1} = \dots = \mathcal{B}^{\text{prim-var}}_{k,T} = \{B_1, \dots, B_{n_k}\}$. All in all, this gives a *constant primal LP-binning*

$$\mathcal{B}^{\text{prim-cons}} = \big(\mathcal{B}^{\text{prim-cons}}_{k,t} \,\big|\, k = 1, \dots, K,\, t = 1, \dots, T\,\big).$$

Compared to $\mathcal{B}^{\text{prim-var}}$, this binning gives a much more refined picture of the block structure of an aggregate for all those periods with zero processing in the optimal LP-solution. However, it is restricted to the same aggregation level over all time periods, which is in general suboptimal when cutoff grades vary, see Remark 5.4.

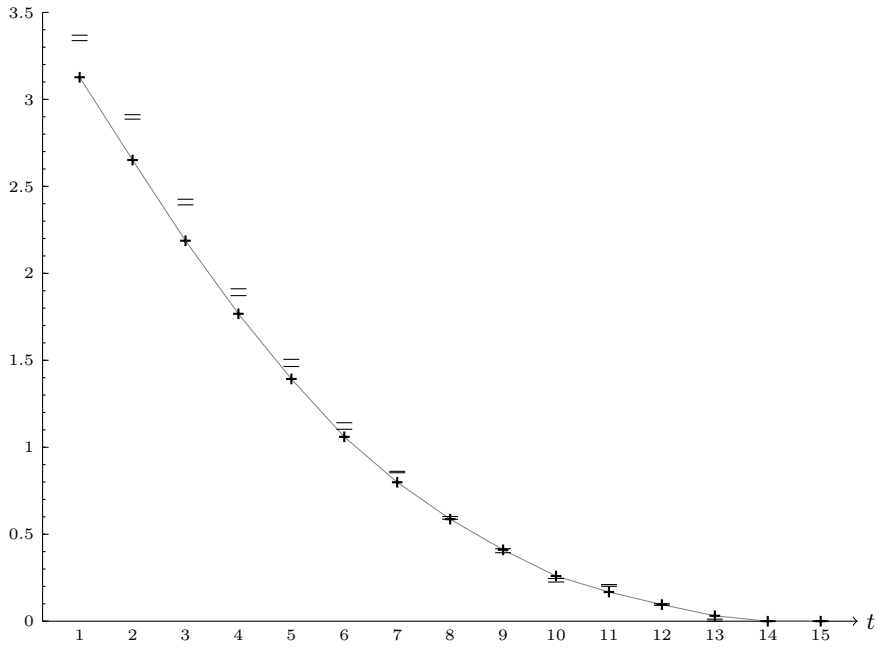

Figure 5.1: Comparison of split ratios for *D-MIP* and *D-LP* for instance marvin-115-8513

The bars indicate the intervals $[\underline{\sigma}_t, \overline{\sigma}_t]$ computed from an optimal solution of *D-MIP* according to Proposition 3.7. The solid line follows a sequence of split ratios π_t for the corresponding LP-relaxation *D-LP* computed as optimal dual multipliers of the processing constraints. The open pit mine is fully exploited during the first 13 years (in the LP-solution), thus the y-variables are zero for the last two years and $\overline{\sigma}_{14} = \overline{\sigma}_{15} = +\infty$. The maximum relative deviation between π_t and the closest ratio in $[\underline{\sigma}_t, \overline{\sigma}_t]$ is 8.3%.

5.3.2 Binnings based on dual LP-solutions

A straightforward idea is to use split ratios for *D-LP* as approximate split ratios for *D-MIP*. There is no proven connection between both, but Figure 5.1, for instance, shows comparable behaviour. We must not expect the two to coincide, though: As Figure 5.1 shows, split-ratios for *D-LP* can exceed or fall below the range of split ratios for *D-MIP* given by Proposition 3.7. Therefore, treating split ratios from *D-LP* as split ratios for *D-MIP* seems to be too optimistic. Instead we try to over- respectively underestimate them in the following way.

Let $\pi \in \mathbb{R}^T_{\geqslant 0}$ be a sequence of split ratios for *D-LP*, for instance given by optimal dual multipliers of the processing constraints, and choose $\epsilon \geqslant 0$. For each $k \in \{1, \ldots, K\}$, $t \in \{1, \ldots, T\}$, form two bins

$$B^- = \left\{ i \in \mathcal{N} \;\middle|\; \frac{p_{i,t}}{a_i} < (1-\epsilon)\pi_t \right\}$$

and

$$B^+ = \left\{ i \in \mathcal{N} \;\middle|\; \frac{p_{i,t}}{a_i} > (1+\epsilon)\pi_t \right\}.$$

For the remaining blocks, let $r_1, \ldots, r_l$ be the distinct values in the set

$\left\{ \frac{p_{i,t}}{a_i} \;\middle|\; \frac{p_{i,t}}{a_i} \in [(1-\epsilon)\pi_t, (1+\epsilon)\pi_t] \right\}$ and form bins

$$B_b = \left\{ i \in \mathcal{N} \;\middle|\; \frac{p_{i,t}}{a_i} = r_b \right\}$$

for $b \in \{1, \dots, l\}$. All in all, this gives the partition $\mathcal{B}^\epsilon_{k,t} = \{B^-, B^+, B_1, \dots, B_l\}$. Together, these form the *dual LP-binning*

$$\mathcal{B}^\epsilon = \big(\mathcal{B}^\epsilon_{k,t} \,\big|\, k = 1, \dots, K,\, t = 1, \dots, T\,\big).$$

If π provided split ratios for *D-MIP*, then even for $\epsilon = 0$, *B-MIP*$\big(\mathcal{B}^\epsilon\big)$ would yield the same optimal solution as *D-MIP*. In general, we expect larger values of ϵ to be necessary, though, in order to have the optimal objective value of *B-MIP*$\big(\mathcal{B}^\epsilon\big)$ still close to the one of *D-MIP*.

Since π is a sequence of split ratios for *D-LP*, the binning $\mathcal{B}^\epsilon$ is a refinement of the variable primal LP-binning $\mathcal{B}^{\text{prim-var}}$ for any $\epsilon \geqslant 0$, due to (3.4). With respect to the objective value, the dual LP-binning will therefore be superior to the variable primal LP-binning.

5.3.3 Computational comparison

To compare the different binnings presented, each *B-MIP*($\mathcal{B}$) was solved using the CPLEX MIP-solver [29], also for the extreme cases of $\mathcal{B}^{\text{block}}$ and $\mathcal{B}^{\text{agg}}$. The results can be found in Table 5.1. Progress is reported by stating the running time and relative primal-dual-gap when the gap first dropped below 1% and below 0.5%. Final values are given after reaching proven optimality or hitting the time limit of 1 hour. For the instances which could not be solved to optimality within 1 hour, a second run was performed using two parallel threads and a time limit of 5 hours. The resulting "best objective value" is stated in the previous but last column. This way all instances could be solved to very small gaps, most to proven optimality, which is marked with a star. The last column reports how this "best objective value" for the respective binning relates to the one obtained for *B-MIP*$\big(\mathcal{B}^{\text{block}}\big)$.

First, the comparatively small number of bins is notable for all binnings except for $\mathcal{B}^{\text{block}}$, of course. In most cases, there are on average less than two bins per aggregate. Even for instance wa-125-96821 with the largest number of 96821 blocks, the most refined dual LP-binning $\mathcal{B}^{0.1}$ only yields on average 1904.1 bins per period, thus decreasing the number of z-variables by a factor of almost 50, even more for the coarser binnings.

Second, this results in a substantial decrease of the running times as can be seen in Table 5.1. Only for instance marvin-1038-8513 the effect is minor, even with a slight increase for $\mathcal{B}^{\text{prim-var}}$. This might be expected, since the number of blocks is already comparatively small for $\mathcal{B}^{\text{block}}$. The number of instances which could be solved to optimality within the time limit of 1 hour

Problem instance	Binning $\mathcal{B}$	Bins per period	1% gap		0.5% gap		Optimality/time limit				Best obj. val. [10^8]	Best obj. val. rel. to $\mathcal{B}^{\text{block}}$
			time [s]	gap [%]	time [s]	gap [%]	time [s]	b&b nodes	obj. val [10^8]	gap [%]		
marvin	$\mathcal{B}^{\text{block}}$	8513.0	240.0	0.865	337.8	0.444	3600.0	4652	7.177235	0.014	7.177235*	1.000000
-115	$\mathcal{B}^{0.1}$	266.0	15.7	0.806	34.1	0.393	427.5	5174	7.177185	0	7.177185*	0.999993
-8513	$\mathcal{B}^{0.05}$	214.3	21.9	1.000	32.8	0.442	964.0	17918	7.176498	0	7.176498*	0.999897
	$\mathcal{B}^{0.01}$	175.6	24.8	0.543	27.6	0.477	2972.7	124119	7.173551	0	7.173551*	0.999487
	$\mathcal{B}^{\text{prim-cons}}$	163.0	18.6	0.642	23.0	0.455	94.9	3035	7.170070	0	7.170070*	0.999002
	$\mathcal{B}^{\text{prim-var}}$	123.3	34.4	0.727	38.0	0.399	46.8	517	6.990731	0	6.990731*	0.974015
	$\mathcal{B}^{\text{agg}}$	115.0	42.3	0.970	55.6	0.486	117.3	1264	5.392583	0	5.392583*	0.751345
marvin	$\mathcal{B}^{\text{block}}$	8513.0	495.3	0.922	679.0	0.432	3600.0	2501	7.220962	0.093	7.221446	1.000000
-296	$\mathcal{B}^{0.1}$	489.1	130.9	0.767	251.4	0.433	3600.0	9026	7.221193	0.055	7.221193	0.999965
-8513	$\mathcal{B}^{0.05}$	439.3	152.3	0.529	169.2	0.462	3600.0	9536	7.220084	0.056	7.220664*	0.999892
	$\mathcal{B}^{0.01}$	399.9	158.6	0.712	196.4	0.484	3600.0	14119	7.218661	0.042	7.218820*	0.999636
	$\mathcal{B}^{\text{prim-cons}}$	392.0	156.4	0.682	184.9	0.368	3600.0	22090	7.213103	0.022	7.213103*	0.998845
	$\mathcal{B}^{\text{prim-var}}$	313.2	133.3	0.760	188.8	0.472	872.8	4538	7.168841	0	7.168841*	0.992715
	$\mathcal{B}^{\text{agg}}$	296.0	284.4	0.703	323.8	0.374	3600.0	18386	5.767265	0.029	5.767305*	0.798630
marvin	$\mathcal{B}^{\text{block}}$	8513.0	2869.3	0.355	2869.3	0.355	3600.0	181	7.260396	0.267	7.262462	1.000000
-1038	$\mathcal{B}^{0.1}$	1294.3	2481.9	0.559	2825.4	0.373	3600.0	271	7.259532	0.270	7.262117	0.999952
-8513	$\mathcal{B}^{0.05}$	1249.9	2715.6	0.188	2715.6	0.188	3600.0	171	7.258831	0.157	7.261993	0.999935
	$\mathcal{B}^{0.01}$	1213.8	2394.1	0.112	2394.1	0.112	3600.0	331	7.261893	0.049	7.262143*	0.999956
	$\mathcal{B}^{\text{prim-cons}}$	1234.0	2247.9	0.282	2247.9	0.282	3600.0	378	7.253915	0.170	7.256165	0.999133
	$\mathcal{B}^{\text{prim-var}}$	1072.3	3067.2	0.461	3067.2	0.461	3600.0	101	7.255896	0.320	7.256970*	0.999244
	$\mathcal{B}^{\text{agg}}$	1038.0	–	–	–	–	3600.0	31	5.863011	9.185	6.328216	0.871360
ca-121	$\mathcal{B}^{\text{block}}$	29266.0	974.6	0.391	990.0	0.269	2272.9	501	59.730918	0.000	59.730918*	1.000000
-29266	$\mathcal{B}^{0.1}$	403.8	57.7	0.480	61.4	0.225	145.0	607	59.730819	0	59.730819*	0.999998
(avg)	$\mathcal{B}^{0.05}$	293.3	52.7	0.350	53.9	0.222	121.1	620	59.730726	0	59.730726*	0.999997
	$\mathcal{B}^{0.01}$	202.7	42.6	0.396	42.9	0.252	93.4	674	59.729226	0	59.729226*	0.999972
	$\mathcal{B}^{\text{prim-cons}}$	178.1	24.6	0.382	26.1	0.170	40.3	314	59.562580	0	59.562580*	0.997182
	$\mathcal{B}^{\text{prim-var}}$	138.1	34.4	0.700	36.7	0.308	41.2	148	59.305581	0	59.305581*	0.992879
	$\mathcal{B}^{\text{agg}}$	121.0	29.1	0.399	30.2	0.275	34.5	159	36.518095	0	36.518095*	0.611377
wa-125	$\mathcal{B}^{\text{block}}$	96821.0	–	–	–	–	–	–	–	–	–	–
-96821	$\mathcal{B}^{0.1}$	1904.1	–	–	–	–	3600.0	189	0.494093	1.477	0.494147	–
	$\mathcal{B}^{0.05}$	1078.8	3397.0	0.993	–	–	3600.0	520	0.494152	0.963	0.494159	–
	$\mathcal{B}^{0.01}$	398.6	495.6	0.752	648.0	0.499	3600.0	9049	0.494107	0.037	0.494107*	–
	$\mathcal{B}^{\text{prim-cons}}$	234.0	159.4	0.935	269.7	0.500	1407.5	12615	0.494068	0	0.494068*	–
	$\mathcal{B}^{\text{prim-var}}$	148.8	98.6	0.940	126.2	0.493	256.3	2147	0.494076	0	0.494076*	–
	$\mathcal{B}^{\text{agg}}$	125.0	–	–	–	–	3600.0	21846	0.281546	1.009	0.281553	–

Table 5.1: Computational results for solving *B-MIP*($\mathcal{B}$) for different binnings $\mathcal{B}$

The CPLEX MIP-solver [29] was used to solve *B-MIP*($\mathcal{B}$) for the binnings introduced in Section 5.3. The LP-relaxation at the root node was solved using the barrier solver with subsequent crossover to a simplex basis. MIP-emphasis was set to 0 ("balance optimality and integer feasibility") and traditional branch-and-cut search without parallelisation was used. In most cases, the problems could not be solved within the time limit of 1 hour. For these, a second run with 2 parallel threads and a time limit of 5 hours was performed. The resulting "best objective value" is reported in the previous but last column and marked with a star, if proven to be optimal. The last column reports how this "best objective value" for the respective binning relates to the one obtained for *B-MIP*($\mathcal{B}^{\text{block}}$). Note also that since the number of bins varies over time, column "Bins per period" gives the average value equal to the number of z-variables divided by number of time periods.

For the "ca"-instances, the values reported are arithmetic mean values over all 25 scenarios. Except for two scenarios with binning $\mathcal{B}^{\text{block}}$, all could be solved to optimality within the time limit of 1 hour. For wa-125-96821, *B-MIP*($\mathcal{B}^{\text{block}}$) could not be solved at all due to memory limits.

also increased. For the largest instance wa-125-96821 especially the binning approach was effective: Whereas *B-MIP*$\left(\mathcal{B}^{\text{block}}\right)$ could not be solved at all due to memory limits, all other binnings yielded gaps of less than 1.5%, even proven optimality for the primal LP-binnings.

Third, and most importantly, the last column shows that these improvements in solvability do not come at the cost of solution quality. The dual LP-binnings decreased the objective value by less than 0.06% in all cases, a value absolutely negligible for practical purposes. The primal LP-binnings prove to be inferior to the dual LP-binnings in terms of objective value, although the running times might be smaller. All in all, these results confirm the usefulness of aggregation approach by binnings. The disaggregation experiments from next section will strengthen them even more.

5.4 Disaggregation of binnings

Once an optimal or near-optimal solution (x^*, y^*, Z^*) is computed for *B-MIP*$(\mathcal{B})$ with some binning $\mathcal{B}$, the next and final step is to disaggregate this to a solution for the original problem *D-MIP*. Standard fixed-weight disaggregation simply keeps the x- and y-values and assigns the Z^*-values for some bin to the corresponding z-variables for the blocks contained in that bin. More precisely, let

$$z^*_{i,t} = Z^*_{k,t,b}$$

for all $k \in \{1, \ldots, K\}$, $i \in \mathcal{K}_k$ and $t \in \{1, \ldots, T\}$, then (x^*, y^*, z^*) is feasible for *D-MIP* and has the same objective value as (x^*, y^*, Z^*).

Performing optimal disaggregation as described in Section 5.1.2 is another possibility, but in terms of objective value, the following approach will be superior. The main difficulty in solving *D-MIP* is determining optimal values for the integer variables x. Fixing those to the values given by x^* from the solution to *B-MIP*$(\mathcal{B})$ leaves only a linear programme to be solved. This way, a feasible solution to *D-MIP* can be determined in reasonable time with best objective value – assuming that the integer variables, which were not subject to aggregation, are not affected by the disaggregation procedure.

To evaluate the effect of this disaggregation, the author conducted experiments similar to those for Table 5.1: *B-MIP*$(\mathcal{B})$ was solved with the time limit of 1 hour, the obtained solution was subsequently disaggregated by solving a linear programme as described above, results are reported in Table 5.2. Using the CPLEX barrier-solver [29] with crossover to a basic solution, disaggregation took less than 38 seconds in all cases, for the smaller marvin-instances at most 5 seconds. The increase of the objective value compared to fixed-weight disaggregation can be seen to be very small, especially for the dual LP-binnings. This emphasises once more the high quality of this aggregation approach.

Problem instance	*B-MIP*($\mathcal{B}$)						Disaggregation to *D-MIP*			Disaggr. obj. val.
	binning $\mathcal{B}$	bins per period	time [s]	b&b nodes	obj. val [10^8]	gap [%]	time [s]	obj. val [10^8]	rel. obj. val.	rel. to $\mathcal{B}^{\text{block}}$
marvin-115-8513	$\mathcal{B}^{\text{block}}$	8513.0	3600.0	4652	7.177235	0.014	–	7.177235	1.000000	1.000000
	$\mathcal{B}^{0.1}$	266.0	427.5	5174	7.177185	0	1.8	7.177235	1.000007	1.000000
	$\mathcal{B}^{0.05}$	214.3	964.0	17918	7.176498	0	1.8	7.177044	1.000076	0.999973
	$\mathcal{B}^{0.01}$	175.6	2972.7	124119	7.173551	0	1.8	7.173963	1.000058	0.999544
	$\mathcal{B}^{\text{prim-cons}}$	163.0	94.9	3035	7.170070	0	2.1	7.175644	1.000777	0.999778
	$\mathcal{B}^{\text{prim-var}}$	123.3	46.8	517	6.990731	0	4.8	7.128473	1.019703	0.993206
	$\mathcal{B}^{\text{agg}}$	115.0	117.3	1264	5.392583	0	3.8	6.086696	1.128716	0.848056
marvin-296-8513	$\mathcal{B}^{\text{block}}$	8513.0	3600.0	2501	7.220962	0.093	–	7.220962	1.000000	1.000000
	$\mathcal{B}^{0.1}$	489.1	3600.0	9026	7.221193	0.055	1.0	7.221232	1.000005	1.000037
	$\mathcal{B}^{0.05}$	439.3	3600.0	9536	7.220084	0.056	1.0	7.220209	1.000017	0.999896
	$\mathcal{B}^{0.01}$	399.9	3600.0	14119	7.218661	0.042	1.0	7.219413	1.000104	0.999785
	$\mathcal{B}^{\text{prim-cons}}$	392.0	3600.0	22090	7.213103	0.022	1.0	7.218101	1.000693	0.999604
	$\mathcal{B}^{\text{prim-var}}$	313.2	872.8	4538	7.168841	0	1.1	7.196448	1.003851	0.996605
	$\mathcal{B}^{\text{agg}}$	296.0	3600.0	18386	5.767265	0.029	0.7	6.381381	1.106483	0.883730
marvin-1038-8513	$\mathcal{B}^{\text{block}}$	8513.0	3600.0	181	7.260396	0.267	–	7.260396	1.000000	1.000000
	$\mathcal{B}^{0.1}$	1294.3	3600.0	271	7.259532	0.270	1.2	7.259980	1.000062	0.999943
	$\mathcal{B}^{0.05}$	1249.9	3600.0	171	7.258831	0.157	1.4	7.258997	1.000023	0.999807
	$\mathcal{B}^{0.01}$	1213.8	3600.0	331	7.261893	0.049	1.3	7.261937	1.000006	1.000212
	$\mathcal{B}^{\text{prim-cons}}$	1234.0	3600.0	378	7.253915	0.170	1.5	7.258118	1.000579	0.999686
	$\mathcal{B}^{\text{prim-var}}$	1072.3	3600.0	101	7.255896	0.320	1.4	7.257076	1.000163	0.999543
	$\mathcal{B}^{\text{agg}}$	1038.0	3600.0	31	5.863011	9.185	1.1	7.006841	1.195093	0.965077
ca-121-29266 (avg)	$\mathcal{B}^{\text{block}}$	29266.0	2272.9	501	59.730918	0.000	–	59.730918	1.000000	1.000000
	$\mathcal{B}^{0.1}$	403.8	145.0	607	59.730819	0	37.0	59.730918	1.000002	1.000000
	$\mathcal{B}^{0.05}$	293.3	121.1	620	59.730726	0	36.2	59.730917	1.000003	1.000000
	$\mathcal{B}^{0.01}$	202.7	93.4	674	59.729226	0	37.3	59.730303	1.000018	0.999990
	$\mathcal{B}^{\text{prim-cons}}$	178.1	40.3	314	59.562580	0	34.7	59.722332	1.002679	0.999856
	$\mathcal{B}^{\text{prim-var}}$	138.1	41.2	148	59.305581	0	32.5	59.662778	1.006120	0.998859
	$\mathcal{B}^{\text{agg}}$	121.0	34.5	159	36.518095	0	24.8	57.004820	1.561625	0.954360
wa-125-96821	$\mathcal{B}^{\text{block}}$	96821.0	–	–	–	–	–	–	–	–
	$\mathcal{B}^{0.1}$	1904.1	3600.0	189	0.494093	1.477	22.6	0.494129	1.000072	–
	$\mathcal{B}^{0.05}$	1078.8	3600.0	520	0.494152	0.963	19.8	0.494155	1.000007	–
	$\mathcal{B}^{0.01}$	398.6	3600.0	9049	0.494107	0.037	17.3	0.494162	1.000112	–
	$\mathcal{B}^{\text{prim-cons}}$	234.0	1407.5	12615	0.494068	0	19.5	0.494154	1.000174	–
	$\mathcal{B}^{\text{prim-var}}$	148.8	256.3	2147	0.494076	0	16.4	0.494162	1.000175	–
	$\mathcal{B}^{\text{agg}}$	125.0	3600.0	21846	0.281546	1.009	17.1	0.470499	1.671130	–

Table 5.2: Computational results for solving *B-MIP*($\mathcal{B}$) for different binnings $\mathcal{B}$ with subsequent disaggregation to *D-MIP*

The CPLEX MIP-solver [29] was used to solve *B-MIP*($\mathcal{B}$) with a time limit of 1 hour, for details see Table 5.1. Subsequently, the best integer solution obtained for *B-MIP*($\mathcal{B}$) was disaggregated to a solution of *D-MIP* as described in Section 5.4 – via fixing the x-variables in *D-MIP* to their values in the *B-MIP*($\mathcal{B}$)-solution and solving the resulting linear programme. This linear programme is solved using the CPLEX barrier-solver [29] with crossover to a basic solution, the time and objective value is reported. (For $\mathcal{B}^{\text{block}}$ no disaggregation is necessary, since *B-MIP*($\mathcal{B}^{\text{block}}$) is identical to *D-MIP*.) Column "rel. obj. val" states the disaggregated objective value relative to the objective value obtained for *B-MIP*($\mathcal{B}$) and thus shows how this disaggregation method compares to fixed-weight disaggregation. The last column reports how the disaggregated objective value for the respective binning relates to the objective value obtained for *B-MIP*($\mathcal{B}^{\text{block}}$). Where *B-MIP*($\mathcal{B}^{\text{block}}$) could not be solved to optimality within the time limit of 1 hour, the disaggregated objective value for coarser binnings can even exceed the objective value for *B-MIP*($\mathcal{B}^{\text{block}}$).

For the "ca"-instances, the values reported are the arithmetic mean values over all 25 scenarios. For wa-125-96821, *B-MIP*($\mathcal{B}^{\text{block}}$) could not be solved at all due to memory limits.

The last column in Table 5.2 shows how the disaggregated objective value compares to the objective value obtained by directly solving the original programme *D-MIP*, i.e. *B-MIP*$\left(\mathcal{B}^{\text{block}}\right)$. Even more than Table 5.1, this shows that the loss in solution quality is very minor and completely negligible for practical purposes. For the marvin-instances, *B-MIP*$\left(\mathcal{B}^{\text{block}}\right)$ could not be solved to optimality within the time limit of 1 hour and so in two cases, the disaggregated objective value could even exceed the objective value obtained by directly solving *B-MIP*$\left(\mathcal{B}^{\text{block}}\right)$. For the "ca"-instances, the dual LP-binnings $\mathcal{B}^{0.1}$ and $\mathcal{B}^{0.05}$ yielded the same objective value on average, and $\mathcal{B}^{0.01}$ a decrease as small as 0.001%.

5.5 Conclusion

This chapter was concerned with the novelty of the OPMPSP-formulation *D-MIP* used in this thesis: the integrated optimisation of cutoff grades. To achieve this, a possibly large number of additional continuous variables had to be used in *D-MIP*, which at first sight increases the computational difficulty of solving the programme. To overcome this disadvantage, this chapter developed the concept of binnings, a column aggregation approach: Following this, most of the new variables can be aggregated to reduce the size of *D-MIP*. Various binnings were proposed and compared computationally. The results proved this concept very successful, leading to significantly decreased running times when solving the aggregated programmes by the commercial MIP-solver CPLEX 11.0 [29]. For one instance in the test set, finding a solution was even impossible using the original formulation, whereas it could be solved to small optimality gaps when applying suitable binnings. Most importantly, for the best binnings the aggregation led to practically no loss in the objective value. All in all, this shows that the additional flexibility provided by the new OPMPSP-formulation does not necessarily come at the cost of increasing the computational difficulty immensely.

Chapter 6

Primal solutions

In this chapter we are finally concerned with probably the most relevant question from the application point of view: methods for computing primal solutions of high quality. Since the OPMPSP is NP-hard, it is natural to consider heuristic approaches. This chapter presents several integer-programming-based heuristics.

Section 6.1 introduces a generic class of greedy sub-MIP algorithms, which can serve as start heuristics. Two specific heuristics of this form are proposed, one proceeding by time periods and one based on a solution of the lagrangean relaxation from Section 3.4. Section 6.2 presents a large neighbourhood search heuristic, which can be used to improve given solutions further. All heuristics are evaluated computationally. Section 6.3 describes preliminary experiments, which have been conducted with a promising branch-and-bound approach based on lagrangean relaxation. Instead of solving the LP-relaxation directly, dual bounds were computed using the lagrangean approach described in Section 3.4 and successfully tested in Chapter 4. With further research concerning the integration of valid inequalities, this promises to be a successful approach to obtain solutions of provably high quality.

6.1 Start heuristics for the OPMPSP

As Kleinberg and Tardos [34] explain, "[it] is hard, if not impossible, to define precisely what is meant by a *greedy algorithm*." Vaguely they call an algorithm greedy "if it builds up a solution in small steps, choosing a decision at each step myopically to optimize some underlying criterion. One can often design many different greedy algorithms for the same problem, each one locally, incrementally optimizing some different measure on its way to a solution."

In the following, we first introduce a generic greedy heuristic for the OPMPSP. Subsequently, we present two specific algorithms of this type, and then describe an improved optimality measure for the greedy steps in the generic algorithm. For the exposition, we consider the special case of *D-MIP*,

but the concept can be applied analogously to the other formulations from Chapter 2. In particular, we will test the algorithms on the aggregated mixed-integer programme *B-MIP*$\big(\mathcal{B}^{0.01}\big)$ from Chapter 5.

6.1.1 A generic greedy sub-MIP heuristic

The general idea is simple: We try to determine "good" values for the integer variables of *D-MIP* in a greedy fashion. To this end, solve *D-MIP* on a suitable subset of its variables and fix some of the x-variables to the values obtained. Subsequently, new variables are added and the programme is resolved. This procedure is iterated until all variables have been added to the model.

Heuristics, which apply the optimisation of smaller mixed-integer programmes, are also termed *sub-MIP heuristics*. One of the advantages of sub-MIP heuristics is for instance that any improvement in general MIP-solving automatically leads to improved performance of the sub-MIP heuristic. Using sub-MIP heuristics for general MIP-solving was probably first considered by Fischetti and Lodi [16]. A detailed description and computational evaluation of several general sub-MIP heuristics is given by Berthold [5, 4].

To ensure that the procedure described above yields a feasible solution, the following notion is useful:

Definition 6.1 *Let an instance of D-MIP on K aggregates and T time periods be given, and let the set of predecessors of aggregate $\mathcal{K}_k$, $k \in \{1, \dots, K\}$ be denoted by $\bar{\mathcal{P}}(k)$. We call $\mathcal{J} \subseteq \{1, \dots, K\} \times \{1, \dots, T\}$ a* precedence-feasible index set *for D-MIP if for all $(k,t) \in \mathcal{J}$, also*

$$(\ell, s) \in \mathcal{J} \quad \text{for all } \ell \in \bar{\mathcal{P}}(k) \cup \{k\},\ s \in \{1, \dots, t\}. \tag{6.1}$$

By the very definition of a precedence-feasible index set $\mathcal{J}$, the following mixed-integer programme, which arises from *D-MIP* by restriction of its feasible region, is well-defined:

maximise

$$\sum_{(k,t)\in\mathcal{J}} -\bar{c}_{k,t}\, y_{k,t} \;+\; \sum_{(k,t)\in\mathcal{J}} p_{i,t}\, z_{i,t} \tag{6.2}$$

subject to

$$x_{k,t-1} - x_{k,t} \leqslant 0 \qquad \text{for all } (k,t) \in \mathcal{J},\ t \geqslant 2, \tag{6.2a}$$

$$\sum_{s=1}^{t} y_{k,s} - x_{k,t} \leqslant 0 \qquad \text{for all } (k,t) \in \mathcal{J}, \tag{6.2b}$$

$$x_{k,t} - \sum_{s=1}^{t} y_{\ell,s} \leqslant 0 \qquad \text{for all } (k,t) \in \mathcal{J},\ \ell \in \bar{\mathcal{P}}(k), \tag{6.2c}$$

$$z_{i,t} - y_{k,t} \leqslant 0 \qquad \text{for all } (k,t) \in \mathcal{J},\ i \in \mathcal{K}_k, \tag{6.2d}$$

$$\sum_{k:(k,t)\in\mathcal{J}} \bar{a}_k y_{k,t} \leqslant U_t^{\mathrm{m}} \qquad \text{for all } t \in \{1,\dots,T\}, \tag{6.2e}$$

$$\sum_{k:(k,t)\in\mathcal{J}} \sum_{i\in\mathcal{K}_k} a_i z_{i,t} \leqslant U_t^{\mathrm{p}} \qquad \text{for all } t \in \{1,\dots,T\}, \tag{6.2f}$$

$$x_{k,t} \in \{0,1\} \qquad \text{for all } (k,t) \in \mathcal{J}, \tag{6.2g}$$

$$0 \leqslant y_{k,t} \leqslant 1 \qquad \text{for all } (k,t) \in \mathcal{J}, \tag{6.2h}$$

$$0 \leqslant z_{i,t} \leqslant 1 \qquad \text{for all } (k,t) \in \mathcal{J},\ i \in \mathcal{K}_k. \tag{6.2i}$$

Now suppose we are given an increasing and exhausting sequence of precedence-feasible index sets,

$$\emptyset \neq \mathcal{J}_1 \subset \mathcal{J}_2 \subset \ldots \subset \mathcal{J}_q = \{1,\dots,K\} \times \{1,\dots,T\}.$$

The algorithm starts by solving the above programme for $\mathcal{J} = \mathcal{J}_1$, which we refer to as *G-MIP*$_1$. This yields a solution $\left(x^{(1)}, y^{(1)}, z^{(1)}\right)$. At iteration $r \geqslant 2$ of the algorithm, let $\left(x^{(r-1)}, y^{(r-1)}, z^{(r-1)}\right)$ be a solution of *G-MIP*$_{r-1}$. Furthermore, choose a subset of indices $\mathcal{J}'_r \subseteq \mathcal{J}_{r-1}$, for which the corresponding x-variables shall be fixed to their value in $x^{(r-1)}$. Then solve programme (6.2) for $\mathcal{J} = \mathcal{J}_r$ with the additional constraint

$$x_{k,t} = x_{k,t}^{(r-1)} \text{ for all } (k,t) \in \mathcal{J}'_r. \tag{6.3}$$

We will refer to this programme as *G-MIP*$_r$. This process is iterated until $r = q$ is reached. The solution of the last programme, *G-MIP*$_q$, provides a solution of *D-MIP*, since $\mathcal{J}_q = \{1,\dots,K\} \times \{1,\dots,T\}$.

The following simple lemma shows that this generic algorithm is well-defined in the sense that the programmes *G-MIP*$_1$, …, *G-MIP*$_q$ can be solved sequentially, each of them having a feasible solution:

Lemma 6.2

(i) G-MIP$_1$ *is feasible.*

(ii) For $2 \leqslant r \leqslant q$, *let* $\left(x^{(r-1)}, y^{(r-1)}, z^{(r-1)}\right)$ *be a feasible solution of G-MIP*$_{r-1}$ *and* $\mathcal{J}'_r \subseteq \mathcal{J}_{r-1}$. *Then programme G-MIP*$_r$ *is feasible.*

Proof. For (i), because we consider only nonnegative mining and processing capacities, the zero solution is feasible for *G-MIP*$_1$ in any case.

For (ii), we can extend the solution $\left(x^{(r-1)}, y^{(r-1)}, z^{(r-1)}\right)$ of *G-MIP*$_{r-1}$ by assigning zero mining and processing "activity" to the new variables. It is

straightforward to check that $(\tilde{x}, \tilde{y}, \tilde{z})$ defined by

$$\tilde{x}_{k,t} = \begin{cases} x_{k,t}^{(r-1)} & \text{if } (k,t) \in \mathcal{J}_{r-1}, \\ \max\left\{ x_{k,s}^{(r-1)} \;\middle|\; s \leqslant t : (k,s) \in \mathcal{J}_{r-1} \right\} & \text{if } \exists\, s \leqslant t : (k,s) \in \mathcal{J}_{r-1}, \\ 0 & \text{otherwise,} \end{cases}$$

$$\tilde{y}_{k,t} = \begin{cases} y_{k,t}^{(r-1)} & \text{if } (k,t) \in \mathcal{J}_{r-1}, \\ 0 & \text{otherwise,} \end{cases}$$

$$\tilde{z}_{i,t} = \begin{cases} z_{i,t}^{(r-1)} & \text{if } (k,t) \in \mathcal{J}_{r-1}, \\ 0 & \text{otherwise,} \end{cases}$$

for all $(k,t) \in \mathcal{J}_r$, $i \in \mathcal{K}_k$, is feasible for *G-MIP*$_r$. □

Since the procedure is heuristic anyway, the *G-MIP*$_r$ do not necessarily have to be solved to optimality. Solving only to a certain optimality gap or, when using a standard LP-based branch-and-cut solver, imposing a limit on the number of branch-and-bound nodes or the number of simplex iterations, might result in superior performance in practise. We summarise the generic algorithm in pseudocode:

Generic greedy algorithm

Input: OPMPSP-instance as described in Section 2.1
Sequence of precedence-feasible index sets
$\emptyset \neq \mathcal{J}_1 \subset \ldots \subset \mathcal{J}_q = \{1,\ldots,K\} \times \{1,\ldots,T\}$
Parameter: Gap limit $\epsilon \geqslant 0$
Output: Solution (x^*, y^*, z^*) of *D-MIP*

```
1 begin
2     solve G-MIP_1 to a proven optimality gap of ε
3     (x^(1), y^(1), z^(1)) ⟵ incumbent solution obtained for G-MIP_1
4     for r = 2 to q do
5         choose J'_r ⊆ J_{r-1}
6         solve G-MIP_r to a proven optimality gap of ε
7         (x^(r), y^(r), z^(r)) ⟵ incumbent solution obtained for G-MIP_r
8     (x*, y*, z*) ⟵ (x^(q), y^(q), z^(q))
9 end
```

The next sections will present two choices for the sequence of precedence-feasible index sets, yielding specific algorithms of this type.

6.1.2 A time-based greedy heuristic

The first specific algorithm is a generalisation of a greedy heuristic proposed by Fricke [18, pp. 196] for the OPMPSP-formulation (2.3). First, (2.3) is solved only for the time period $t = 1$. This schedule is fixed and a schedule for the next time period is determined on top of that. Iterating this up to time period T yields a feasible schedule. In computational experiments conducted by Fricke, this heuristic performed well only for a relatively shallow open pit mine, where most of the orebody can be accessed already in early time periods. For deeper open pit mines, the heuristic proved to be not sufficiently farsighted and yielded highly suboptimal objective values.

Fricke suggested to take into account all time periods as follows: At iteration t, the time periods $t+1, \dots, T$ are not disregarded as above, but aggregated suitably into one time period and considered in the optimisation. This approach yielded significantly better results with respect to the objective value, but consumed considerably more running time in turn.

We propose a slightly different possibility to overcome the disadvantages of the "greediness": Let L be the *look-ahead*, i.e. the number of time periods to consider during each iteration and let F be the number of time periods to be fixed after each iteration, $1 \leqslant F \leqslant L \leqslant T$. First, we optimise a schedule for time periods $1, \dots, L$, and fix the x-variables for time periods $1, \dots, F$. Subsequently, we add the next F time periods and iterate.

Precisely, define $\mathcal{T}_t = \{(k,t) \mid k = 1, \dots, K\}$ for all $t \in \{1, \dots, T\}$. This gives a sequence of index sets

$$\mathcal{J}_r = \mathcal{T}_1 \cup \dots \cup \mathcal{T}_{\min\{(r-1)F+L, T\}}$$

for $r = 1, \dots, q$, where $q = \min\{r \mid (r-1)F + L \geqslant T\}$. These index sets are trivially precedence-feasible according to Definition 6.1, increasing, and $\mathcal{J}_q = \{1, \dots, K\} \times \{1, \dots, T\}$. Hence, we obtain a well-defined algorithm of the generic type from Section 6.1.1, if we specify the choice in Line 5:

$$\mathcal{J}'_r = \mathcal{T}_1 \cup \dots \cup \mathcal{T}_{(r-1)F}$$

for $r = 2, \dots, q$.

For $L = F = 1$, the algorithm proceeds as the heuristic of Fricke, which was described initially. Computational results for $1 \leqslant F \leqslant L \leqslant 3$ are reported in Section 6.1.5.

6.1.3 A greedy heuristic based on lagrangean relaxation

While the above heuristic used index sets for all aggregates and a certain range of time periods, this one will select indices only for a subset of aggregates, but all time periods $1, \dots, T$. It is based on the observation that any solution to the lagrangean relaxation described in Section 3.4, already provides us with

a precedence-feasible schedule, "only" the resource constraints are possibly violated. A common approach is to transform such partially feasible solutions to fully feasible solutions in a heuristic manner.

Suppose we have computed an optimal solution $\left(x^{D-LR}, y^{D-LR}, z^{D-LR}\right)$ of *D-LR*(μ, π) for some Lagrange multipliers $\mu, \pi \in \mathbb{R}^T_{\geqslant 0}$, preferably optimal or near-optimal. Then for each $t \in \{1, \ldots, T\}$,

$$\mathcal{C}_t = \left\{k \in \{1, \ldots, K\} \,\middle|\, x^{D-LR}_{k,t} = 1\right\}$$

is the set of indices of aggregates which are started to be mined before or during time period t in the solution of the lagrangean relaxation. This gives a sequence of index sets

$$\mathcal{J}_r = \{(k, t) \,|\, k \in \mathcal{C}_r,\, t = 1, \ldots, T\},$$

for $r = 1, \ldots, T$, and $\mathcal{J}_q = \{1, \ldots, K\} \times \{1, \ldots, T\}$, $q = T+1$. The index sets $\mathcal{J}_1, \ldots, \mathcal{J}_T$ are precedence-feasible because $\left(x^{D-LR}, y^{D-LR}, z^{D-LR}\right)$ satisfies the constraints of *D-LR*(μ, π), $\mathcal{J}_{T+1}$ trivially. After each iteration, we fix all the newly added x-variables, i.e. for $r = 2, \ldots, q$, we choose

$$\mathcal{J}'_r = \mathcal{J}_{r-1}$$

in the generic algorithm.

Note that $\mathcal{C}_t$ may be empty for some $t \in \{1, \ldots, T\}$, depending on the solution of the lagrangean relaxation. Then in the corresponding iteration no new variables are added and it can be skipped without even building and solving the programme *G-MIP*$_t$.

6.1.4 An improved optimality measure for the generic greedy sub-MIP heuristic

The greedy heuristics presented above proceed from top of the open pit mine to the bottom, at each iteration finding a locally optimal schedule. In cases, where most of the ore is located at the bottom of the mine, this is problematic during the first iterations, where only blocks close to the surface can be feasibly mined. In particular, if negative net present value during early time periods must necessarily occur in order to access high-value material in later time periods, a greedy approach will fail: The zero solution will appear superior to the greedy algorithm with its limited horizon.

The time-based greedy heuristic from Section 6.1.2 tried to overcome this by looking ahead at time periods in the close future. However, if we want to remain within reasonable running times, we will unlikely be able to use a large look-ahead. One approach to reach a more global perspective is the following: At iteration r of the generic greedy algorithm from Section 6.1.1,

the values of $x_{k,t}$, $(k,t) \in \mathcal{J}_r$, were determined by disregarding the columns corresponding to indices not comprised by $\mathcal{J}_r$.

Instead of completely removing those columns from the model, one might merely relax the integrality conditions on them. This also leads to reduced computational effort, but keeps the global perspective. Instead of *G-MIP*$_r$, we solve the programme *PIR-MIP*$_r$,[1]

maximise

$$\sum_{k=1}^{K}\sum_{t=1}^{T} -\bar{c}_{k,t}\, y_{k,t} \;+\; \sum_{i=1}^{N}\sum_{t=1}^{T} p_{i,t}\, z_{i,t} \tag{6.4}$$

subject to

$$x_{k,t-1} - x_{k,t} \leqslant 0 \qquad \text{for all } k \in \{1,\dots,K\},\ t \in \{2,\dots,T\}, \tag{6.4a}$$

$$\sum_{s=1}^{t} y_{k,s} - x_{k,t} \leqslant 0 \qquad \text{for all } k \in \{1,\dots,K\},\ t \in \{1,\dots,T\}, \tag{6.4b}$$

$$x_{k,t} - \sum_{s=1}^{t} y_{\ell,s} \leqslant 0 \qquad \text{for all } k \in \{1,\dots,K\},\ \ell \in \bar{\mathcal{P}}(k),\ t \in \{1,\dots,T\}, \tag{6.4c}$$

$$z_{i,t} - y_{k,t} \leqslant 0 \qquad \text{for all } k \in \{1,\dots,K\},\ i \in \mathcal{K}_k,\ t \in \{1,\dots,T\}, \tag{6.4d}$$

$$\sum_{k=1}^{K} \bar{a}_k y_{k,t} \leqslant U_t^{\mathrm{m}} \qquad \text{for all } t \in \{1,\dots,T\}, \tag{6.4e}$$

$$\sum_{i=1}^{N} a_i z_{i,t} \leqslant U_t^{\mathrm{p}} \qquad \text{for all } t \in \{1,\dots,T\}, \tag{6.4f}$$

$$x_{k,t} = x_{k,t}^{(r-1)} \qquad \text{for all } (k,t) \in \mathcal{J}_r', \tag{6.4g}$$

$$x_{k,t} \in \{0,1\} \qquad \text{for all } (k,t) \in \mathcal{J}_r, \tag{6.4h}$$

$$0 \leqslant x_{k,t} \leqslant 1 \qquad \text{for all } k \in \{1,\dots,K\},\ t \in \{1,\dots,T\}, \tag{6.4i}$$

$$0 \leqslant y_{k,t} \leqslant 1 \qquad \text{for all } k \in \{1,\dots,K\},\ t \in \{1,\dots,T\}, \tag{6.4j}$$

$$0 \leqslant z_{i,t} \leqslant 1 \qquad \text{for all } i \in \mathcal{N},\ t \in \{1,\dots,T\}. \tag{6.4k}$$

This is exactly the original problem *D-MIP*, when (6.4g) is added and (2.5g), i.e. the integrality constraint on the x-variables, is replaced by (6.4h) and (6.4i). *PIR-MIP*$_1$ does not contain constraint (6.4g), of course.

A corresponding version of Lemma 6.2 holds equally for *PIR-MIP*$_r$ and is not restated here. Hence, if we replace *G-MIP*$_r$ by *PIR-MIP*$_r$ in the generic greedy heuristic from Section 6.1.1, we obtain a well-defined algorithm. The underlying criterion which the greedy algorithm optimises at each step is now extended to a more global perspective.

[1] *PIR-MIP* for "partially integer-relaxed MIP".

Following the same reasoning as in Section 3.1, programme *PIR-MIP*$_r$ can be simplified: We may assume that for all $(k,t) \notin \mathcal{J}_r$, constraint (6.4b) holds with equality, without loss in objective value. Then we can perform the substitution $x_{k,t} = \sum_{s=1}^{t} y_{k,s}$ for all $(k,t) \in \{1,\dots,K\} \times \{1,\dots,T\} - \mathcal{J}_r$ and remove the corresponding constraints in (6.4a).

The next section will compare the heuristics which were proposed so far computationally.

6.1.5 Computational comparison

In order to evaluate the two types of start heuristics following the generic greedy scheme from Section 6.1.1 – the time-based heuristic from Section 6.1.2 for $1 \leqslant F \leqslant L \leqslant 3$, and the lagrangean-based heuristic from Section 6.1.3 –, the author conducted experiments on the data sets described in Section 1.3. The algorithms were run for the aggregated programme *B-MIP*$\left(\mathcal{B}^{0.01}\right)$ from the previous chapter.

The sub-MIPs – *G-MIP*$_r$ respectively *PIR-MIP*$_r$ – were solved to a relative optimality gap of 0.01% using the CPLEX MIP-solver [29] with MIP-emphasis 1 (emphasis on integer feasibility), traditional branch-and-cut search and primal simplex for the root relaxation. The latter proved faster than using barrier or dual simplex algorithm. The results are reported in Table 6.1. All of the heuristics showed a very consistent performance, always yielding objective values within 10% of the global optimum. Only 7 runs with time-based heuristics yielded objective values lower than 95% of the optimum. For all instances, solutions within 0.5% of the optimum could be obtained within running times of at most 81 seconds.

The variant using *G-MIP*$_r$ as sub-MIPs consumed running times of less than 16 seconds for almost all problem instances. Only for the largest and most difficult instance marvin-1038-8513, the running time was as large as 133.2 seconds. Among the time-based heuristics, the versions with $L > F$ performed better in general, as might be expected. Only for the "ca"-instances, surprisingly the most shortsighted heuristic with $L = F = 1$ yielded the highest objective value on average.

With *G-MIP*$_r$ as sub-MIPs, the time-based heuristics were mostly faster than the lagrangean-based heuristic, but the latter outperformed them clearly with respect to the objective value. Even for the most difficult problem instance marvin-1038-8513, the lagrangean-based heuristic computed a schedule within 0.432% of the optimum and 1.052% of the LP-gap, taking 81 seconds. Compared to this, in the experiments conducted for Table 5.1, it took almost 40 minutes to obtain a solution with less than 1% primal-dual-gap using CPLEX 11.0 [29]. For all other instances, the optimality gap was consistently below 0.1% and consumed never more than 16 seconds of running time. For

Problem instance	Heuristic	L	F	*G-MIP$_r$* as sub-MIPs			*PIR-MIP$_r$* as sub-MIPs				
				time [s]	opt. gap [%]	LP-gap [%]	time [s]	opt. gap [%]	LP-gap [%]	rel. time	rel. obj. val.
marvin-115-8513	time-based	1	1	0.5	2.401	3.268	12.9	0.005	0.793	23.888889	1.024552
	time-based	2	1	1.0	0.317	1.108	18.9	0.005	0.793	19.285714	1.003125
	time-based	3	1	2.3	0.287	1.078	30.5	0.003	0.791	13.038462	1.002846
	time-based	2	2	0.6	2.310	3.171	11.2	0.008	0.796	19.362069	1.023566
	time-based	3	2	1.0	0.267	1.058	16.8	0.003	0.791	16.421569	1.002651
	time-based	3	3	0.8	0.312	1.103	14.9	0.003	0.791	17.761905	1.003099
	lagr.-based	–	–	0.6	0.075	0.864	10.9	0.030	0.819	17.901639	1.000449
marvin-296-8513	time-based	1	1	1.3	8.744	10.459	210.9	0.009	0.810	158.556391	1.095724
	time-based	2	1	3.6	7.871	9.413	244.9	0.023	0.824	67.460055	1.085187
	time-based	3	1	9.9	1.096	1.917	326.0	0.016	0.816	32.865927	1.010920
	time-based	2	2	1.6	9.685	11.610	169.5	0.032	0.833	103.975460	1.106877
	time-based	3	2	4.8	0.563	1.371	165.6	0.023	0.824	34.497917	1.005429
	time-based	3	3	3.9	3.421	4.371	130.5	0.016	0.816	33.803109	1.035260
	lagr.-based	–	–	15.7	0.060	0.861	144.9	0.032	0.833	9.258786	1.000278
marvin-1038-8513	time-based	1	1	7.5	9.667	11.383	–	–	–	–	–
	time-based	2	1	29.4	5.571	6.552	–	–	–	–	–
	time-based	3	1	133.2	2.668	3.374	–	–	–	–	–
	time-based	2	2	13.2	8.372	9.809	–	–	–	–	–
	time-based	3	2	67.7	3.835	4.629	–	–	–	–	–
	time-based	3	3	49.1	4.814	5.705	–	–	–	–	–
	lagr.-based	–	–	81.0	0.432	1.052	–	–	–	–	–
ca-121-29266 (avg)	time-based	1	1	0.5	1.266	1.572	16.6	0.502	0.790	36.009565	1.007717
	time-based	2	1	0.8	3.295	3.717	22.9	0.021	0.304	28.868115	1.033845
	time-based	3	1	1.5	3.740	4.209	30.7	0.012	0.295	21.087243	1.038791
	time-based	2	2	0.5	2.175	2.526	14.5	0.107	0.391	31.540400	1.021156
	time-based	3	2	0.8	4.172	4.692	20.2	0.014	0.297	25.460765	1.043609
	time-based	3	3	0.6	2.318	2.679	15.5	0.111	0.394	26.885813	1.022765
	lagr.-based	–	–	6.3	0.084	0.368	19.5	0.026	0.309	3.120772	1.000575
wa-125-96821	time-based	1	1	1.8	9.469	13.747	168.4	1.107	4.130	92.500000	1.092360
	time-based	2	1	3.4	1.990	5.067	220.9	0.005	2.982	65.741071	1.020248
	time-based	3	1	6.8	1.377	4.415	272.0	0.006	2.983	40.001471	1.013906
	time-based	2	2	1.7	2.083	5.167	117.3	0.093	3.073	67.436782	1.020316
	time-based	3	2	3.7	0.676	3.677	136.8	0.001	2.978	36.981081	1.006793
	time-based	3	3	2.5	4.631	7.977	90.2	0.761	3.766	36.366935	1.040576
	lagr.-based	–	–	8.7	0.000	2.977	137.0	0.000	2.977	15.761795	1.000001

Table 6.1: Computational results on the performance of start heuristics for *B-MIP*$(\mathcal{B}^{0.01})$
Two heuristics following the generic greedy scheme from Section 6.1.1 were tested: the time-based heuristic from Section 6.1.2 for $1 \leqslant F \leqslant L \leqslant 3$, and the lagrangean-based heuristic from Section 6.1.3.
For each heuristic, we report the overall running time including the time to build and alter the sub-MIPs. The latter is negligible compared to the time to solve the sub-MIPs, though. Columns "opt. gap" report the gap of the objective value obtained by the heuristic to the optimal objective value available from Table 5.1 and is calculated as $\frac{\text{opt. obj. val} - \text{heur. obj. val}}{\text{opt. obj. val}}$. Columns "LP-gap" state the gap to the dual bound given by the LP-relaxation, calculated as $\frac{\text{LP-bound} - \text{heur. obj. val}}{\text{heur. obj. val}}$. The last two columns compare the variant using *PIR-MIP$_r$* from Section 6.1.4 with the heuristics using *G-MIP$_r$* as initially introduced. For the instance marvin-1038-8513, the heuristics with *PIR-MIP$_r$* as sub-MIPs failed to finish within the time limit of 30 minutes. For the "ca"-instances, the values given are arithmetic mean values over all 25 scenarios.

wa-125-96821, the lagrangean-based heuristic even yielded a scheduled with optimal objective value.

The variant using *PIR-MIP*$_r$ as sub-MIPs proved even more effective. For the time-based heuristics especially the more global perspective led to significantly lower optimality gaps. In most cases, the objective value obtained was within 0.05% of the optimum. Again, the lagrangean heuristic showed the most consistent behaviour.

This superior performance comes at the cost of increased running times as large as 326 seconds. Even more, for the largest instance marvin-1038-8513, none of the heuristics using *PIR-MIP*$_r$ finished within the time limit of 30 minutes. However, there is great potential to speed up the solution of the sub-MIPs *PIR-MIP*$_r$.

To begin with, for simplicity of implementation the sub-MIPs were solved as (6.4) and not in the simplified version as explained. Using the latter should yield shorter running times. In general, a large fraction of the time for solving the sub-MIPs is spent on the solution of the LP-relaxations, in particular at the root node. The experiments in Chapter 4 demonstrated that the lagrangean dual can be solved much faster than the LP-relaxation. Solving the lagrangean dual for example via a bundle algorithm as in Chapter 4 (or other means, such as stabilised column-generation) can also provide a solution to the LP-relaxation. This may be expected to speed up the solution of the sub-MIPs *PIR-MIP*$_r$, yielding significantly reduced overall running times for the heuristics. Then the heuristics can be applied to larger problem instances, such as marvin-1038-8513, as well.

6.2 An improvement heuristic for the OPMPSP

In contrast to start heuristics as presented in the previous sections, improvement heuristics require one or more feasible solutions as input, on basis of which they search for solutions with superior objective value.

One type of improvement heuristics following the local search paradigm are so-called *large neighbourhood search heuristics*.[2] Given some reference point in the solution space (or a relaxation of the solution space), the idea is to define a neighbourhood about this point – typically "large" compared to classical local search methods – and search it for superior feasible solutions. This search is typically conducted by solving a sub-MIP obtained from the original mixed-integer programme by merely restricting the domains of the variables.

In the following, we present an OPMPSP-specific heuristic of this type. For the exposition, we consider again the case of *D-MIP*, but the concept can be

[2]One example of a large neighbourhood search heuristic used in general MIP-solving is the so-called relaxation-induced neighbourhood search by Danna et al. [14].

applied analogously to the other formulations from Chapter 2. In particular, we will test the heuristic on the solutions from Table 6.1 for *B-MIP*$\left(\mathcal{B}^{0.01}\right)$.

6.2.1 An OPMPSP-specific large neighbourhood search heuristic

Let a feasible solution (x^*, y^*, z^*) of *D-MIP* be given. To define a neighbourhood about this solution, we regard all solutions in which the mining of each aggregate starts at most δ time periods earlier or later than in (x^*, y^*, z^*), for fixed $\delta \in \{1, 2, \ldots\}$. For each $k \in \{1, \ldots, K\}$, let $t^*(k)$ denote the time period during which the mining of aggregate $\mathcal{K}_k$ is started, $T+1$ if it is not scheduled at all:

$$t^*(k) = \begin{cases} \min\{t \mid x_{k,t} = 1\} & \text{if } x_{k,T} = 1, \\ T+1 & \text{otherwise.} \end{cases}$$

Then we can define a neighbourhood about (x^*, y^*, z^*) as described above by

$$\delta\text{-}LN(x^*, y^*, z^*) = \left\{ \begin{array}{c} (x,y,z) \\ \text{feasible for} \\ \textit{D-MIP} \end{array} \middle| \begin{array}{l} \forall\, k \in \{1,\ldots,K\}, t \in \{1,\ldots,T\}: \\ x_{k,t} = \begin{cases} 0 & \text{if } t \leqslant t^*(k) - \delta - 1, \\ 1 & \text{if } t \geqslant t^*(k) + \delta. \end{cases} \end{array} \right\},$$

and solve the programme δ-*LNS-MIP*(x^*, y^*, z^*):

$$\begin{aligned} &\text{maximise} \\ &\sum_{k=1}^{K} \sum_{t=t^*(k)-1}^{T} \left(-\bar{c}_{k,t}\, y_{k,t} \;+ \sum_{i \in \mathcal{K}_k} p_{i,t}\, z_{i,t} \right) \qquad (6.5) \\ &\text{subject to } (x,y,z) \in \delta\text{-}LN(x^*, y^*, z^*). \end{aligned}$$

Because $(x^*, y^*, z^*) \in \delta\text{-}LN(x^*, y^*, z^*)$, this programme is feasible. It contains at most $(2\delta + 1)K$ (non-fixed) integer variables and is, for small values of δ, significantly reduced in size compared to the full *D-MIP*.

The next section will give computational results on the performance of this improvement heuristic for $\delta = 1$ and $\delta = 2$.

6.2.2 Computational evaluation

The author conducted experiments solving programmes *1-LNS-MIP*(x^*, y^*, z^*) and *2-LNS-MIP*(x^*, y^*, z^*). The reference solutions (x^*, y^*, z^*) were obtained by the start heuristics from Section 6.1 with *G-MIP*$_r$ as sub-MIPs, see Table 6.1. Again, the CPLEX MIP-solver [29] was used to solve (6.5) with MIP-emphasis 1 (emphasis on integer feasibility), traditional branch-and-cut search and primal simplex for the root relaxation. The programmes were solved near-optimally with a relative optimality gap of $\epsilon = 0.01\%$ and a time limit of 30 minutes, which was hit once for instance marvin-296-8513. The objective value of the reference solutions provides a lower bound on the optimal

Problem instance	Reference solution				*1-LNS-MIP*				*2-LNS-MIP*			
	heuristic	L	F	opt. gap [%]	time [s]	rel. obj. val.	opt. gap [%]	LP-gap [%]	time [s]	rel. obj. val.	opt. gap [%]	LP-gap [%]
marvin	time-based	1	1	2.401	6.8	1.024355	0.024	0.812	88.0	1.024584	0.001	0.790
-115	time-based	2	1	0.317	4.0	1.003107	0.007	0.795	45.9	1.003153	0.002	0.791
-8513	time-based	3	1	0.287	1.3	1.002598	0.027	0.816	51.4	1.002859	0.001	0.790
	time-based	2	2	2.310	8.9	1.023551	0.009	0.798	83.7	1.023646	0.000	0.788
	time-based	3	2	0.267	8.3	1.002461	0.022	0.810	109.3	1.002679	0.000	0.788
	time-based	3	3	0.312	3.3	1.002907	0.022	0.810	51.8	1.003108	0.002	0.790
	lagr.-based	–	–	0.075	8.7	1.000726	0.003	0.791	69.0	1.000739	0.001	0.790
marvin	time-based	1	1	8.744	1.1	1.087429	0.766	1.579	1669.7	1.094933	0.081	0.882
-296	time-based	2	1	7.871	0.9	1.076337	0.838	1.653	1179.5	1.084651	0.072	0.874
-8513	time-based	3	1	1.096	398.4	1.009744	0.132	0.934	1171.2	1.011078	0.000	0.801
	time-based	2	2	9.685	0.9	1.084565	2.047	2.907	685.9	1.106043	0.107	0.909
	time-based	3	2	0.563	88.2	1.004979	0.067	0.869	1454.2	1.005657	0.000	0.801
	time-based	3	3	3.421	26.2	1.033586	0.177	0.980	1800.0	1.034716	0.068	0.869
	lagr.-based	–	–	0.060	69.1	1.000361	0.024	0.825	815.1	1.000597	0.000	0.801
marvin	time-based	1	1	9.667	27.9	1.089816	1.554	2.204	1499.5	1.106811	0.019	0.634
-1038	time-based	2	1	5.571	36.6	1.056380	0.247	0.865	790.2	1.058776	0.021	0.637
-8513	time-based	3	1	2.668	178.9	1.026597	0.079	0.695	1397.2	1.027307	0.010	0.626
	time-based	2	2	8.372	8.1	1.081951	0.863	1.492	1032.7	1.091058	0.029	0.645
	time-based	3	2	3.835	85.8	1.038690	0.115	0.731	1218.9	1.039696	0.018	0.634
	time-based	3	3	4.814	62.4	1.049420	0.110	0.727	1021.5	1.050363	0.021	0.636
	lagr.-based	–	–	0.432	129.5	1.003939	0.039	0.655	804.9	1.004236	0.010	0.626
ca-121	time-based	1	1	1.266	0.6	1.012671	0.019	0.301	2.9	1.012769	0.009	0.292
-29266	time-based	2	1	3.295	0.4	1.029783	0.424	0.714	3.6	1.034184	0.006	0.289
(avg)	time-based	3	1	3.740	0.4	1.031939	0.682	0.978	2.9	1.039110	0.004	0.287
	time-based	2	2	2.175	0.5	1.021108	0.119	0.404	2.8	1.022293	0.007	0.290
	time-based	3	2	4.172	0.4	1.032759	1.054	1.362	2.4	1.043897	0.006	0.289
	time-based	3	3	2.318	0.4	1.021961	0.185	0.470	3.3	1.023810	0.008	0.291
	lagr.-based	–	–	0.084	0.4	1.000787	0.007	0.289	2.3	1.000813	0.004	0.287
wa-125	time-based	1	1	9.469	14.6	1.071364	3.008	6.170	25.7	1.072236	2.929	6.084
-96821	time-based	2	1	1.990	2.5	1.018723	0.155	3.136	79.6	1.020303	0.000	2.977
	time-based	3	1	1.377	11.4	1.013962	0.000	2.977	181.3	1.013965	0.000	2.977
	time-based	2	2	2.083	22.8	1.021271	0.000	2.977	108.7	1.021271	0.000	2.977
	time-based	3	2	0.676	15.4	1.006804	0.000	2.977	72.5	1.006804	0.000	2.977
	time-based	3	3	4.631	8.7	1.047638	0.087	3.067	40.7	1.048547	0.001	2.977
	lagr.-based	–	–	0.000	30.7	1.000001	0.000	2.977	82.2	1.000002	0.000	2.977

Table 6.2: Computational results on the performance of an OPMPSP-specific large neighbourhood search heuristic for *B-MIP*$(\mathcal{B}^{0.01})$

Programmes *1-LNS-MIP*(x^*, y^*, z^*) and *2-LNS-MIP*(x^*, y^*, z^*) were solved for reference solutions (x^*, y^*, z^*) obtained by the start heuristics from Section 6.1 with *G-MIP*$_r$ as sub-MIPs, see Table 6.1. Columns "rel. obj. val." report the final objective value of δ-*LNS-MIP*(x^*, y^*, z^*) relative to the objective value of the reference solution. Columns "opt. gap" state the gap of the final objective value obtained by the heuristic to the optimal objective value available from Table 5.1 and is calculated as $\frac{\text{opt. obj. val} - \text{heur. obj. val}}{\text{opt. obj. val}}$. Columns "LP-gap" report the gap to the dual bound given by the LP-relaxation, calculated as $\frac{\text{LP-bound} - \text{heur. obj. val}}{\text{heur. obj. val}}$. For the "ca"-instances, the values given are arithmetic mean values over all 25 scenarios.

objective value and was applied as a lower cutoff value during the branch-and-cut algorithm. The results are reported in Table 6.2.

The heuristic *1-LNS-MIP* already showed very good performance. Running times are reasonable compared to the values in Table 5.1, often even within a few seconds, except for some runs with the larger problem instances marvin-296-8513 and marvin-1038-8513. The highest objective values were obtained when starting from best solutions, i.e. from the solution obtained by

the lagrangean-based heuristic. By this, for all instances a final objective value within 0.05% of the optimum could be reached. Moreover, for the reference solutions obtained by the time-based heuristics, which had mostly comparatively large optimality gaps, the optimality gap could be reduced significantly.

Naturally, solving *2-LNS-MIP* yielded even better results with respect to the objective value. Starting from the lagrangean-based heuristic solution, the objective value obtained was within 0.01% of the optimum in all cases. However, especially for the "marvin"-instances running times seem too large when compared with the values in Table 5.1.

Note though that the sub-MIPs must not necessarily be solved to a relative optimality gap of 0.01%, as done here, where a lot of time is consumed to lower the dual bound. Solving only to a larger gap might yield almost as high an objective value, while consuming less time. When integrating these large neighbourhood search heuristics within a branch-and-cut framework, the running time allowed can easily be controlled, for example by limiting the number of branch-and-bound nodes or the number of simplex iterations used for solving the sub-MIP.

The programmes *δ-LNS-MIP* have exactly the same structure as the original formulation. The next section will explain how the standard LP-based branch-and-cut approach might allow for improvements through integration of the lagrangean approach which was developed in Section 3.4 and successfully tested in Chapter 4. A simple possibility is to solve the LP-relaxation at the root node via solution of the lagrangean dual.

6.3 A lagrangean-based branch-and-cut approach – preliminary experiments

In the previous sections we presented heuristics which produced high quality solutions within less than 0.05% of the optimal objective value on all test instances. The optimum needed for this evaluation could be computed using the branch-and-cut MIP-solver CPLEX 11.0 [29]. This was possible because we used data sets where the number of aggregates (and hence integer variables) was rather small compared to realistically-sized block models.[3] Certainly, using aggregation methods like the fundamental tree method [30, 45] mentioned briefly in Chapter 2, even very large-scale block models can be reduced to a tractable number of aggregates. Nevertheless, it is desirable to use as large a number of aggregates as possible in order to retain a good approximation of the original block model.

A main time-consuming factor when applying a standard LP-based branch-and-cut algorithm is the solution of the LP-relaxation, especially at the root

[3]The data sets were primarily chosen for experiments with the aggregation approach from Chapter 5. For these, not necessarily the large number of aggregates, but rather the large number of blocks within each aggregate was relevant.

node. The experiments in Chapter 4 showed that for programme *D-MIP* the lagrangean dual arising from relaxation of the resource constraints can be solved within a fraction of the time necessary to solve the LP-relaxation, yielding practically the same bound. This can be used in various ways to improve the standard LP-based branch-and-cut approach.

First, solving the lagrangean dual, e.g. by a bundle algorithm as in Chapter 4 or by other means, such as stabilised column generation, can also provide a (possibly approximate) solution to the LP-relaxation by aggregating solutions of the lagrangean subproblems. Thus, one possibility is to solve the root relaxation this way (followed by a crossover to a simplex basis) instead of using standard simplex or interior point algorithms. This should already lead to a significant speed-up at the root node, especially for large-scale instances.

Considering the computational results of Chapter 4, the lagrangean approach could be of use even further by replacing the LP-relaxation also at the nodes of the branch-and-bound tree. To evaluate the viability of a completely lagrangean-based branch-and-cut approach, the author conducted preliminary experiments using the software SCIP 1.05 [3], a framework for solving constraint integer programmes. *Constraint integer programming* [1, 2] is a generalisation of mixed-integer programming and SCIP provides a highly customisable branch-cut-and-price framework for solving mixed-integer programmes. By default, SCIP is LP-based, but the LP-functionality can be turned off and customised methods for computing dual bounds can be integrated. Only when all integer variables are fixed, the remaining linear programme must be solved. Interfaces for several LP-solvers exist. The author used CPLEX 11.0 [29] in his experiments.

As for testing the heuristics in Tables 6.1 and 6.2, the experiments were conducted for the aggregated programme *B-MIP*$\left(\mathcal{B}^{0.01}\right)$. Thus, using the bundle algorithm as in Chapter 4, the dual bound at the root node could be computed even faster than in Table 4.1 for *D-MIP*. When solving the lagrangean dual at the nodes of the branch-and-bound tree, the optimal dual multipliers were saved for the last two nodes processed. This way, when selecting a child node after branching, the bundle algorithm could be warmstarted from the multipliers of the parent node. This yielded an even faster computation of the dual bound. By additionally storing the optimal multipliers at the root node, a warmstart was also possible when selecting a non-child-node in the tree.

Branching decisions on the integer variables are realised by fixing x-variables to 0 or 1 in *D-LR*(μ, π). For a down-branch, this carries over to the equivalent project scheduling formulation *PS-D-LR*(μ, π). For an up-branch, however, one must be careful: It can be shown that not the variable corresponding to the aggregate itself, but the variables for all of its predecessors

must be fixed to 1 in *PS-D-LR*(μ, π). In the minimum cut digraph D as described in Section 3.4.3, this can be enforced by setting capacities of certain assignment arcs to infinity.

At the nodes of the branch-and-bound tree, the lagrangean-based greedy heuristic was applied to a solution of the current lagrangean relaxation for optimal multipliers, always giving a feasible solution, possibly inferior to the incumbent, though. Applying the large neighbourhood search heuristic from Section 6.2, the incumbent solution was improved further. As stated above, it is crucial to keep control of the computational effort such that the overall search process profits from the provided solutions, but does not suffer from the computational overhead. In the experiments, this was achieved by applying the heuristics only at a certain frequency. Other possibilities include limiting the number of branch-and-bound nodes or the number of simplex iterations.

Because the lagrangean dual can also provide an LP-solution at each node, the same branching rules as in LP-based branch-and-bound could be applied. For the experiments though, variants of strong branching were used. These could be performed extensively due to the fast dual bound computation. Moreover, heuristics were run immediately at the root node. This gave a primal bound right from the start and was used during strong branching: If the local dual bound dropped below this primal bound for some branching, the branch was pruned and the corresponding x-variable fixed to the opposite value.

This approach proved very successful in creating high quality solutions, due to the good performance of the heuristics integrated. However, a pure branch-and-bound approach seems to be insufficient when it comes to proving optimality. Various branching and node selection rules were tried, but by branching alone the dual bound did not decrease during the early stages of the branch-and-bound process. This was confirmed by experiments where SCIP was run with standard settings, but cutting plane separators turned off: The dual bound remained at the level of the initial root dual bound.

One possibility to obtain tighter bounds is to try lagrangean relaxation of a different set of constraints such that a nonintegral feasible region is left. This would most likely yield tighter bounds than given by the LP-relaxation. On the other hand, solving the lagrangean subproblems will probably become more time-consuming.

Alternatively, we showed in Section 3.5 how to integrate valid inequalities into the lagrangean relaxation. SCIP provides routines for separation of several general mixed-integer cutting planes which do not depend on the simplex tableau, such as complemented mixed-integer rounding inequalities, see Wolter [51]. This allows for computing valid inequalities which are violated by a fractional LP-solution given by the bundle algorithm. If the coefficients of the x-variables in the valid inequality are nonnegative, it can be dualised

as described in Section 3.5. For specific combinatorial cutting planes, as for example discussed by Fricke [18], the latter condition is generally satisfied. A branch-and-cut algorithm of this type seems promising and worthwhile being considered for further research.

6.4 Conclusion

This chapter presented several heuristic methods proving to be successful in computing OPMPSP-solutions of high quality. Section 6.1 described two start heuristic based on a common generic greedy scheme: one proceeding by time periods and one based on a solution of the lagrangean relaxation from Section 3.4. The latter especially proved very successful in the experiments conducted and yielded high quality solutions with objective values within a 0.5% gap of the optimum.

In Section 6.1.4 we discussed how these heuristics can be improved by adding a more global perspective to the greedy steps. On one hand, these variants performed even more successful in the experiments with respect to the objective value. On the other hand, these gains in objective value came at the cost of highly increased running times. However, we noted potential to decrease these running times by a more careful implementation and utilisation of the lagrangean approach. Also, it must be pointed out that the improvement over the "standard" greedy scheme is not only a quantitative one meaning that it would result in a slightly increased objective value. It is of a different quality in the respect that it has a global perspective: At each greedy step, the entire orebody for all time periods is taken into consideration, of course in a relaxed form. Hence, it may be expected to yield consistently good results also in cases where typical shortsighted greedy approaches will fail: for example for open pit mines where all the high-value material is located towards the bottom of the pit.

With the effective large neighbourhood search heuristic described and tested in Section 6.2, the solutions obtained by the start heuristics could be improved even further. For all test instances, high quality solutions with objective value within 0.05% gap of the optimum were found.

Finally, an outlook on possible further research directions was given: The primal heuristics from this chapter can be brought together with the lagrangean dual bound computation from Chapter 4 in a customised branch-and-bound algorithm. To evaluate the effectiveness of this approach, the author conducted preliminary experiments using SCIP [3] as a branch-and-bound framework. The experiments were described in detail in Section 6.3. The results proved this approach successful in generating solutions with objective value very close to the optimum. However, it also became clear that branch-and-bound alone is insufficient to decrease the optimality gap in the

early stages of the branch-and-bound process. To obtain tighter dual bounds, either different lagrangean relaxations need to be applied, or valid inequalities integrated into the one already used. Testing this computationally seems to be a promising direction for further research.

Conclusion

Since the initial application of mathematical optimisation methods to mine planning in 1965, the Lerchs-Grossmann algorithm for computing the ultimate pit limit, operations researchers have worked on a variety of challenging problems in the area of open pit mining. This thesis focused on the open pit mining production scheduling problem, which was described in detail in Section 2.1. Historically, the open pit mining production scheduling problem was mainly addressed by means of heuristics. Exact methods were not applicable to large-scale real-world problem instances. Only through the improvement in general MIP-solving during recent years has integer programming become viable solution approach. This thesis treated several aspects of solving open pit mining production scheduling problems by integer programming.

A common method for making large-scale block models of open pit mines computationally tractable is to aggregate blocks to bigger units, which we called mining aggregates. Researchers have developed specified methods for this aggregation, which take into account the net present value objective as well as the precedence relations. One example briefly mentioned is the fundamental tree algorithm of Johnson, Dagdelen and Ramazan [30, 45]. Aggregation methods such as this yield mining aggregates with precedence relations aligned to the precedence relations of the underlying block model – the precedence structure is therefore approximated reasonably well. However, the standard integer programming formulations in the literature presuppose a homogeneous distribution of ore within each mining aggregate, and hence these formulations yield a rather bad approximation of the original block structure with respect to the ore distribution. Therefore, in this thesis we considered a new mixed-integer programming formulation first described by Boland et al. [6].

This model respects the in general heterogeneous ore distribution by separately deciding for each block whether it should be processed[4] or discarded as waste, depending on the oregrade. Thus, implicitly the model allows for *integrated cutoff grade optimisation*, i.e. for determining the optimal value of the oregrade below which material should be discarded as waste. The investigation

[4] In this context, "processing" means the further refinement and extraction of ore from material excavated from the open pit mine.

of this new model is the thrust of this thesis. Parts of the results presented, especially from Section 3.2 and Chapter 5, have also appeared in [6].

In Section 3.2 we formally proved the existence of so-called "split ratios", which perform the function of what are referred to as cutoff grades in mining terminology. Given values for these split ratios, the variables controlling the block processing decisions could be suitably removed, yielding a significantly smaller, but equivalent programme. However, we showed that computing split ratios is NP-hard. Therefore, Chapter 5 proposed a slightly less rigorous approach using column aggregation techniques. Split ratios for the LP-relaxation were used heuristically as approximations to exact split ratios, and blocks with similar oregrade were aggregated to so-called "bins". This approach proved to be very successful in computational experiments: The size of the programmes could be reduced drastically while incurring only a completely negligible decrease in objective value.

Another focus of the thesis was lagrangean relaxation: In Section 3.4, we relaxed the resource constraints in a lagrangean fashion, leaving an integral feasible region. Because of the close connection of open pit mining production scheduling to resource-constrained project scheduling, a result of Möhring et al. [41] could be applied to this lagrangean subproblem in a straightforward way: For fixed Lagrange multipliers, the lagrangean relaxation could be solved efficiently by computing minimum s-t-cuts in a suitably constructed network.

For Chapter 4 we conducted experiments solving the corresponding lagrangean dual by means of a bundle algorithm. By this method, the dual LP-bound and dual multipliers associated with the resource constraints could be computed near-optimally, taking only a fraction of the time needed to solve the LP-relaxation by one of the standard linear programming algorithms. Fast computation of optimal multipliers associated with the processing constraints proved to be of interest also for the aggregation approach in Chapter 5, as the very successful dual LP-binnings were based on them.

Chapter 6 proposed several heuristics specific to the open pit mining production scheduling problem. As start heuristics, we introduced a generic class of greedy sub-MIP heuristics and two specific algorithms of this type: one proceeding by time periods, another converting a solution of the lagrangean relaxation to a fully feasible solution. The latter especially showed very good performance in computational experiments and yielded solutions of high quality with objective values within a 0.5% gap of the optimum. We also proposed an interesting improvement of these start heuristics by adding a more global perspective to the single greedy steps. The computational results were promising: With respect to the objective value, this variant clearly outperformed the initially tested heuristics. The problem was that, considering that these were

heuristics, running times were too large. We noted, however, that this drawback might be overcome by a more sophisticated implementation and by integration of the lagrangean approach. A large neighbourhood search heuristic specific to the open pit mining production scheduling problem proved effective in improving solutions obtained by the start heuristics described above even further.

Moreover, the rapid solution of the lagrangean dual, as seen in computational results from Chapter 4, motivated preliminary experiments with a lagrangean-based branch-and-bound algorithm. Details of these experiments were described in Section 6.3. Due to the integration of the efficient heuristics described, the approach proved very effective in computing high quality solutions. However, the experiments also showed that branching alone seems insufficient to decrease the optimality gap in the early stages of the branch-and-bound algorithm. The dual bound remained at the level of the initial root bound. We concluded that either integrating valid inequalities or trying lagrangean relaxation with a different set of constraints is necessary to yield not only high-value primal solutions, but also a good proof of optimality. Testing this computationally seems to be a promising direction for further research.

List of frequently used notation

a_i	rock tonnage of block i
$\bar{a}_k$	rock tonnage of aggregate $\mathcal{K}_k$
$\mathcal{B}$	binning
$c_{i,t}$	cost of mining block i during time period t
$\bar{c}_{k,t}$	cost of mining aggregate $\mathcal{K}_k$ during time period t
f	objective function (of some programme)
f^*	optimal objective value (of some programme)
ϕ	(lagrangean) dual function
$\mathcal{J}$	precedence-feasible index set
$\kappa_{i,t}$	objective coefficient of variable $z_{i,t}$ in the lagrangean relaxation *D-LR*(μ, π)
K	number of aggregates
$\mathcal{K}_k$	aggregate number k, a subset of the set of blocks $\mathcal{N} = \{1, \ldots, N\}$
$\mathcal{K}^+_{k,t}$	subset of blocks i in aggregate $\mathcal{K}_k$ with positive $\kappa_{i,t}$
μ, π	vectors of dual multipliers for mining and processing constraints, respectively, $\mu, \pi \in \mathbb{R}^T_{\geqslant 0}$
$\mathcal{N}$	set of blocks (or block indices), $\mathcal{N} = \{1, \ldots, N\}$
$p_{i,t}$	profit from processing block i during time period t
$\mathcal{P}(i)$	set of immediate predecessor blocks (or block indices) of block i, $\mathcal{P}(i) \subseteq \mathcal{N}$
$\bar{\mathcal{P}}(k)$	set of immediate predecessor aggregates (or aggregate indices) of aggregate $\mathcal{K}_k$, $\bar{\mathcal{P}}(k) \subseteq \{1, \ldots, K\}$
σ	sequence of split ratios for *D-MIP* or *D-LP*, $\sigma = (\sigma_1, \ldots, \sigma_T)$
T	number of time periods
U^{m}_t	mining capacity for time period t
U^{p}_t	processing capacity for time period t

List of mathem. programmes

List of tables

List of figures

References

[1] Tobias Achterberg. *Constraint Integer Programming*. PhD thesis, Technische Universität Berlin, July 2007.

[2] Tobias Achterberg, Timo Berthold, Thorsten Koch and Kati Wolter. Constraint Integer Programming: a new approach to integrate CP and MIP. In: Laurent Perron and Michael A. Trick (eds.), *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 5015 of *LNCS*, pages 6–20. Springer-Verlag, 2008.

[3] Tobias Achterberg, Timo Berthold, Marc Pfetsch and Kati Wolter. SCIP (Solving Constraint Integer Programs), Documentation. `http://scip.zib.de/`.

[4] Timo Berthold. Heuristics of the branch-cut-and-price-framework SCIP. In: Jörg Kalcsics and Stefan Nickel (eds.), *Operations Research Proceedings 2007*, pages 31–36. Springer-Verlag, 2008.

[5] Timo Berthold. Primal heuristics for mixed integer programs. Master's thesis, Technische Universität Berlin, September 2006.

[6] Natashia Boland, Irina Dumitrescu, Gary Froyland and Ambros M. Gleixner. LP-based disaggregation approaches to solving the open pit mining production scheduling problem with block processing selectivity. *Computers and Operations Research*, to appear.

[7] Olivier Briant, Claude Lemaréchal, Philippe Meurdesoif, Sophie Michel, Nancy Perrot and François Vanderbeck. Comparison of bundle and classical column generation. Technical Report 5453, INRIA, January 2005.

[8] Louis Caccetta, Lou M. Giannini and P. Kelsey. On the implementation of exact optimization techniques for open pit design. *Asia-Pacific Journal of Operational Research*, 11:155–170, 1994.

[9] Louis Caccetta and Stephen P. Hill. An application of branch and cut to open pit mine scheduling. *Journal of Global Optimization*, 27:349–365, 2003.

[10] Louis Caccetta, P. Kelsey and Lou M. Giannini. Open pit mine production scheduling. In: *Proceedings of the Third Regional APCOM Symposium*, pages 65–72, Kalgoorlie, WA, Australia, 1998.

[11] Bala G. Chandran and Dorit S. Hochbaum. A computational study of the pseudoflow and push-relabel algorithms for the maximum flow problem. *Operations Research*, to appear.

[12] Bala G. Chandran and Dorit S. Hochbaum. *Pseudoflow Solver Version 3.21.* `http://riot.ieor.berkeley.edu/riot/Applications/Pseudoflow/maxflow.html` (accessed April 2008).

[13] E. Ward Cheney and Allen A. Goldstein. Newton's method for convex programming and tchebycheff approximation. *Numerische Mathematik*, 1:253–268, 1959.

[14] Emilie Danna, Edward Rothberg and Claude Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming A*, 102(1):71–90, 2004.

[15] George B. Dantzig and Philip Wolfe. Decomposition priciple for linear programs. *Operations Research*, 8(1):101–111, 1960.

[16] Matteo Fischetti and Andrea Lodi. Local branching. *Mathematical Programming B*, 98(1-3):23–47, 2003.

[17] Marshall L. Fisher. The lagrangian relaxation method for solving integer programming problems. *Management Science*, 27(1):1–18, 1981.

[18] Christopher Fricke. *Applications of Integer Programming in Open Pit Mining.* PhD thesis, University of Melbourne, August 2006.

[19] Gary Froyland, Merab Menabde, Peter Stone and Dave Hodson. The value of additional drilling to open pit mining projects. In: *Proceedings of the International Symposium on Orebody Modelling and Strategic Mine Planning: Uncertainty and Risk Management*, pages 169–176, Perth, WA, Australia, 2004.

[20] Arthur M. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical Programming Study*, 2:82–114, 1974.

[21] Jean-Louis Goffin, Alain Haurie and Jean-Philippe Vial. Decomposition and nondifferentiable optimization with the projective algorithm. *Management Science*, 38(2):284–302, 1992.

[22] Jean-Louis Goffin and Jean-Philippe Vial. Convex nondifferentiable optimization: A survey focused on the analytic center cutting plane method. *Optimization Methods and Software*, 17(5):805–867, 2002.

[23] Andrew V. Goldberg and Robert E. Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM*, 35:921–940, 1988.

[24] Michael Held and Richard M. Karp. The traveling salesman problem and minimum spanning trees. *Operations Research*, 18:1138–1162, 1970.

[25] Christoph Helmberg. *The ConicBundle Library for Convex Optimization Version 0.2d.* `http://www-user.tu-chemnitz.de/~helmberg/ConicBundle/` (accessed April 2008).

[26] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex Analysis and Minimization Algorithms.* Springer-Verlag, Berlin, Germany, 1993. Two volumes.

[27] Dorit S. Hochbaum. The pseudoflow algorithm and the pseudoflow-based simplex for the maximum flow problem. In: *Proceedings of Integer Programming and Combinatorial Optimization, 6th International IPCO conference*, pages 325–337, Houston, TX, USA, 1998.

[28] Dorit S. Hochbaum and Anna Chen. Performance analysis and best implementations of old and new algorithms for the open-pit mining problem. *Operations Research*, 48:894–914, 2000.

[29] Ilog Inc. *Ilog CPLEX 11.0 User's Manual*, September 2007.

[30] Thys B. Johnson, Kadri Dagdelen and Salih Ramazan. Open pit mine scheduling based on fundamental tree algorithm. In: *Proceedings 30th APCOM*, pages 147–159, Phoenix, AZ, USA, 2002.

[31] James E. Kelley Jr. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

[32] Hans Kellerer, Ulrich Pferschy and David Pisinger. *Knapsack Problems.* Springer-Verlag, Berlin, Germany, 2004.

[33] Alf Kimms. Maximizing the net present value of a project under resource constraints using a lagrangian relaxation based heuristc with tight upper bounds. *Annals of Operations Research*, 102:221–236, 2001.

[34] Jon Kleinberg and Éva Tardos. *Algorithm Design.* Addison-Wesley, Boston, MA, USA, 2005.

[35] Claude Lemaréchal. An algorithm for minimizing convex functions. In: Jack L. Rosenfeld (ed.), *Information Processing 74*, pages 552–556, North Holland, 1974.

[36] Claude Lemaréchal. Lagrangian relaxation. In: Michael Jünger and Denis Naddef (eds.), *Computational Combinatorial Optimization: Optimal Or Provably Near-optimal Solutions*, pages 112–156. Springer-Verlag, Berlin, Germany, 2001.

[37] Helmut Lerchs and Ingo F. Grossmann. Optimum design of open-pit mines. *Transactions CIM*, LXVIII:17–24, 1965.

[38] Igor Litvinchev and Vladimir Tsurkov. *Aggregation in Large-Scale Optimization*, volume 83 of *Applied Optimization*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2003.

[39] Gemcom Ltd. Gemcom Whittle. `http://www.gemcomsoftware.com/products/whittle/` (accessed May 2008).

[40] Merab Menabde, Gary Froyland, Peter Stone and Gavin Yeates. Mining schedule optimisation for conditionally simulated orebodies. In: *Proceedings of the International Symposium on Orebody Modelling and Strategic Mine Planning: Uncertainty and Risk Management*, pages 347–352, Perth, WA, Australia, 2004.

[41] Rolf H. Möhring, Andreas S. Schulz, Frederik Stork and Marc Uetz. Solving project scheduling problems by minimum cut computations. *Management Science*, 49(3):330–350, 2003.

[42] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, New York, USA, 1988.

[43] Ahmet H. Onur and Peter A. Dowd. Open-pit optimization – part 2: Production scheduling and inclusion of roadways. In: *Transactions of the Institution of Mining and Metallurgy (Section A: Mining Industry)*, volume 102, pages 105–113, 1993.

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

[45] Salih Ramazan. The new fundamental tree algorithm for production scheduling of open pit mines. *European Journal of Operational Research*, 177(2):1153–1166, 2007.

[46] David F. Rogers, Robert D. Plante, Richard T. Wong and James R. Evans. Aggregation and disaggregation techniques and methodology in optimization. *Operations Research*, 39(4):553–582, 1991.

[47] Jayaram K. Sankaran, Dennis L. Bricker and Shuw-Hwey Juang. A strong fractional cutting-plane algorithm for resource-constrained project scheduling. *International Journal of Industrial Engineering*, 6:99–111, 1999.

[48] M.L. Smith. Optimizing inventory stockpiles and mine production: An application of separable and goal programming to phosphate mining using AMPL/CPLEX. *CIM Bulletin*, 92:61–64, 1999.

[49] Hirofumi Uzawa. Iterative methods for concave programming. In: Kenneth J. Arrow, Leonid Hurwicz and Hirofumi Uzawa (eds.), *Studies in Linear and Non-Linear Programming*, pages 154–165. Stanford University Press, 1959.

[50] Laurence A. Wolsey. *Integer Programming*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, New York, USA, 1998.

[51] Kati Wolter. Implementation of cutting plane separators for mixed integer programs. Master's thesis, Technische Universität Berlin, December 2006.

[52] Paul H. Zipkin. Bounds on the effect of aggregating variables in linear programs. *Operations Research*, 28(2):403–418, 1980.