Mathematical Optimization and Public Transportation

Dr. Ralf Borndörfer

Kumulative Habilitationsschrift an der
Fakultät II – Mathematik und Naturwissenschaften
der Technischen Universität Berlin

Lehrgebiet:
Mathematik

Eröffnung des Verfahrens: 30.09.2009
Verleihung der Lehrbefähigung: 20.06.2010

Gutachter
Prof. Dr. Dr. h.c. mult. Martin Grötschel
Prof. Dr. Thomas Liebling
Prof. Dr. Rolf H. Möhring
Prof. Dr. Uwe Zimmermann

Berlin 2010

D 83

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Publications

[1] R. Borndörfer & R. Weismantel.
Set packing relaxations of some integer programs.
Mathematical Programming **88**:425–450, 2000.

[2] R. Borndörfer.
Combinatorial packing problems.
In M. Grötschel, (Ed.), *The Sharpest Cut – The Impact of Manfred Padberg and His Work*, pp. 19–32. SIAM, Philadelphia, 2004.

[3] R. Borndörfer, C. E. Ferreira & A. Martin.
Decomposing matrices into blocks.
SIAM Journal on Optimization **9**(1):236–269, 1998.

[4] R. Borndörfer, A. Löbel & S. Weider.
A bundle method for integrated multi-depot vehicle and duty scheduling in public transit.
In M. Hickman, P. Mirchandani & S. Voss, (Eds.), *Computer-aided Systems in Public Transport*, vol. 600 of *Lecture Notes in Economics and Mathematical Systems*, pp. 3–24. Springer-Verlag, 2008.

[5] R. Borndörfer & T. Schlechte.
Models for railway track allocation.
In C. Liebchen, R. K. Ahuja & J. A. Mesa, (Eds.), *Proceeding of the 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS 2007)*, Dagstuhl, Germany, 2007. Internationales Begegbnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

[6] R. Borndörfer, M. Grötschel & M. E. Pfetsch.
A column-generation approach to line planning in public transport.
Transportation Science **41**(1):123–132, 2007.

Introduction

I submit in this cumulative thesis the following six papers for obtaining the habilitation at the Technische Universität Berlin, Fakultät II – Mathematik und Naturwissenschaften:

(1) *Set packing relaxations of some integer programs.*
(2) *Combinatorial packing problems.*
(3) *Decomposing matrices into blocks.*
(4) *A bundle method for integrated multi-depot vehicle and duty scheduling in public transit.*
(5) *Models for railway track allocation.*
(6) *A column-generation approach to line planning in public transport.*

I have made some changes to the papers compared to the published versions. These pertain to layout unifications, i.e., common numbering, figure, table, and chapter head layout. I have not changed any notation or symbols, but I have eliminated some typos, updated the references, and added links and an index. The mathematical content is identical.

The papers are about the *optimization of public transportation systems*, i.e., bus networks, railways, and airlines, and its *mathematical foundations*, i.e., the theory of packing problems. The papers discuss mathematical models, theoretical analyses, algorithmic approaches, and computational aspects of and to problems in this area.

Papers 1, 2, and 3 are theoretical. They aim at establishing a theory of *packing problems* as a general framework that can be used to study traffic optimization problems. Indeed, traffic optimization problems can often be modelled as path packing, partitioning, or covering problems, which lead directly to *set packing*, *partitioning*, and *covering* models. Such models are used in papers 4, 5, and 6 to study a variety of problems concerning the planning of line systems, buses, trains, and crews. The common aim is always to exploit as many degrees of freedom as possible, both at the level of the individual problems by using large-scale integer programming techniques, as well as on a higher level by integrating hitherto separate steps in the planning process. I give in the following an outline of these papers; for surveys on optimization in public transportation I refer the reader to the papers (not included here) Borndörfer, Grötschel & Löbel (2000) [6]; Borndörfer, Grötschel & Pfetsch (2006) [9]; Borndörfer, Grötschel & Jaeger (2008) [11].

(1) Set Packing Relaxations of Some Integer Programs. *Packing*, i.e., finding a conflict free subset of some objects, is a basic task in combina-

torial optimization. The prototype of packing problems is *node packing in graphs*, an intensively investigated area which gave rise to deep and beautiful results and concepts such as antiblocking theory, the theory of perfect graphs and matrices, semidefinite programming, and polyhedral combinatorics. There is, in particular, a substantial body of knowledge about the *set packing polytope*. Many facet defining and/or polynomial time separable types of inequalities have been identified, among them the classes of clique, odd cycle, odd wheel, orthonormal representation, web, and antiweb inequalitites. All of these inequalities are potentially useful in cutting plane algorithms.

The idea of paper 1, jointly written with Robert Weismantel, is that the structures and construction principles that give rise to these inequalities can also be applied to other problems that involve a packing component. Just as one can build a clique or an odd cycle of nodes in a graph, one can build a clique or an odd cycle of general objects, where the edges correspond to pairwise conflicts. The paper shows that affine transformations can be used to associate, in an algebraically simple way, a *set packing relaxation* to another problem, that allows to transfer inequalities and separation algorithms from the set packing problem to the problem of interest.

In fact, this philosophy had, in a sense, already been applied before. Namely, many well-known classes of inequalities for a variety of combinatorial optimization problems can be explained in this way, e.g., the 2-chorded cycle inequalities for the clique partitioning problem, the fence and the Möbius ladder inequalities for the acyclic subdigraph problem, and the generalized wheel inequalities for the set packing problem. Using set packing relaxation techniques it was for the first time possible to show that the Möbius ladder inequalities can be separated in polynomial time. The possibilities for the construction of new classes are illustrated by creating a novel class of cycle-of-cylces inequalitites. Some more results in this direction can be found in the follow-on paper Borndörfer & Weismantel (2001) [4] (not included here).

(2) Combinatorial Packing Problems. An important class of general packing problems are couplings of individual combinatorial problems. Typical examples of such *combinatorial packing problems* are the multicommodity flow problem (packings of paths or flows), the multiple knapsack problem (packings of knapsacks), and the coloring problem (packings of stable sets).

Multicommodity flows are very close to traffic applications. They have integral subproblems, i.e., their difficulty stems solely from the coupling equations. Can such a problem be *naturally integral*, i.e., can it be that the integrality of the individual subproblems carries over to the entire problem? The answer of paper 2 is “yes, sometimes”, e.g., the bipartite 2-coloring

and the matroid packing problem are naturally integral. Moreover, general 2-packing problems have perfect conflict graphs such that their polytopes can be completely understood in terms of projections of clique inequalities from an associated extended *Dantzig-Wolfe formulation*.

(3) Decomposing matrices into blocks. The reverse approach is taken in paper 3, which is joint work with Carlos Ferreira and Alexander Martin. Here, we try to identify (automatically and without any problem specific knowledge) a block structure (of individual problems) with a minimal coupling part in a general MIP. Such a *block decomposition*, if it can be discovered, is useful in several ways: It paves the way for the application of parallel linear algebra methods such as *parallel LU factorizations* or *parallel Cholesky factorizations*, it allows the application of polyhedral knowledge from standard problems with block structure such as the *generalized assignment* or *multiple knapsack problem*, and it can be used in a *branch-and-bound* approach to break up a large problem into independent subproblems that can also be solved in parallel. General MIPs offer potential in this direction, because they often model individual components of a process, machine, or company that are coupled by a number of linking constraints.

The *matrix decomposition problem* is closely related to graph decomposition problems such as the *node separator problem*, to the *set packing*, and the *set covering problem*. It features, in particular, a natural *set packing relaxation* in the original variables, and has been, in this way, a precursor to paper 1. However, knapsack structures and symmetry breaking also play a major role in our branch-and-cut approach to the solution of this interesting problem. It turns out that indeed not all, but quite a number of general MIP instances from the `Netlib` and the `Miplib` problem libraries can be well decomposed into 2 and 4 blocks of roughly the same size.

(4) A Bundle Method for Integrated Multi-Depot Vehicle and Duty Scheduling in Public Transit. *Vehicle and crew scheduling* are prime examples for the successful application of mathematical optimization techniques in practice. The need to solve such problems was among the driving forces behind the development of the multicommodity flow, set partitioning, and column generation machinery that is now used with great success in numerous applications ranging from network design to political districting. Today, all major airlines, public transit companies, and for a short time also the large railways use or are beginning to use mathematical vehicle and crew optimizers on a routine basis to deploy their vehicles and personnel. I was involved in the development of the vehicle scheduling optimizer VS-OPT for public transit as well as the crew scheduling optimizers DS-OPT for public transit and CS-OPT for airlines, that are part of the

commercial scheduling systems ivu.plan (formerly known as MICROBUS II) by IVU Traffic Technologies AG and NetLine by Lufthansa Systems Berlin GmbH (under the brand xOPT Crew). These optimizers are used by many companies all over the world, e.g., at Berlin's public transit company BVG, at the Italian railway Trenitalia, and at the Brasilian airline GOL.

Solving two, otherwise sequentially tackled, planning problems *simultaneously* discloses further optimization potentials that can be used to improve the service and to cut costs. A simple idea, but easier said than done. Setting up two models and adding suitable coupling constraints is not the hard part, and, in fact, many such models have been published. However, not much is known about methods that work for solving integrated real-world scheduling problems. As far as I know, paper 4, which is joint work with Andreas Löbel and Steffen Weider, is the first to report on the solution of industrial large-scale *integrated scheduling problems*, namely, on *integrated vehicle and duty scheduling problems* in public transit. Integrated scheduling is necessary in regional transit, not only for cost optimization, but simply for feasibility. Our method is part of the commercial scheduling system ivu.plan by IVU Traffic Technologies AG, and used, e.g., by the DB Stadtverkehr group (the former "Bahnbusse"), the largest provider of regional public transit in Germany.

The key to the solution of this problem is the *proximal bundle method*, an algorithm for the solution of non-differentiable convex optimization problems. The bundle method can be seen as an advancement of subgradient or volume methods. It has a very similar computational performance, but superior convergence properties, in particular, automatic dual stabilization, and it produces not only a converging series of dual estimates, but also primal values, that are extremely useful in heuristics. Another advantage is that it can also be used to handle quadratic objectives, which come up, e.g., in crew rostering problems. This feature is also very handy in order to get some control on the integrality of 0/1 variables by putting terms such as $x(1-x)$ into the objective function. "Whenever you think of using subgradient methods, you should use the bundle method", said Christoph Helmberg when he introduced the method to us some years ago. This is indeed true. It turns out that the method can deal with the very large coupling systems that come up in integrated scheduling, that it can exploit separability of the objective, that one can make it work with inexact information, and that it provides sufficient synchronization of the individual problems in a superordinate Lagrangean relaxation approach.

I find it surprising that, until today, the bundle method has not gained the acceptance and popularity in the optimization community that it deserves, even though very good open-source implementations are readily available, e.g., on Christoph Helmberg's homepage (http://www-user.tu-chemnitz.

de/~helmberg/ConicBundle/). Interesting research directions are crossover techniques and higher-order extensions. An additional publication on integrated vehicle and duty scheduling (not included here) is Borndörfer, Löbel & Weider (2002) [7], in German, complementary papers on crew scheduling (not included here) are Borndörfer et al. (1999) [5], in German, Borndörfer, Grötschel & Löbel (2003) [8], and Borndörfer et al. (2006) [10].

(5) Models for Railway Track Allocation. The bundle method plays also a major role in our approach to the solution of *track allocation problems*, i.e., the packing of train paths in a railway network. As most railway problems, technical constraints make this problem much harder than its public transit or airline counterparts, such that it is, today, still a challenge to optimize railway traffic through a path-type system such as a tunnel, a mountain pass such as the Brenner, or a corridor like the Rhine route. In fact, several groups in Germany and Italy are currently working on exactly these scenarios.

The most popular approach is to set up a direct packing model, eliminating conflicts by cutting planes. Thomas Schlechte and I propose in paper 5 a different formulation that is based on *track configuration variables*, that model the feasible assignments of slots to trains over a time horizon on a single track. It can be shown that this model produces the same LP bound as the standard model including all clique inequalities and that it is algorithmically tractable (configurations can be priced solving a shortest path problem). In our computationas, it was clearly the best model that allows to solve scenarios involving several hundred trains.

The work is part of a project on railway track auctioning. Additional publications (not included here) are Borndörfer & Schlechte (2008) [2]; Borndörfer & Schlechte (2008) [3]; Borndörfer, Mura & Schlechte (2009) [14].

(6) A Column-Generation Approach to Line Planning in Public Transport. Large effects could be achieved if the use of mathematical optimization could be established in the design phase of transportation systems in a similar way as it is already the case in the subsequent steps of resource allocation. This is currently not the case; today, strategic traffic infrastructure and system design decisions are taken by hand planning using rules of thumb and, in good case, simulation tools. One reason for this lack of mathematical support in *service design* is certainly that the planning problems are very difficult such that often the available methods are not yet adequate to live up to the requirements from practice. Problems in service design involve multiple parties and are therefore inherently integrated, they feature multiple, contradicting objectives such as cost vs. quality, and they

have stochastic and game theoretic traits, e.g., in the prediction of user behaviour. Considering the importance of service design, however, these difficulties can be no reason to defer working on these problems.

Paper 6, coauthored by Martin Grötschel and Marc Pfetsch, investigates the *line planning problem* to set up a cost efficient and high-quality line system in public transit. Decisions involve the routing of lines, and the frequency of service. The passengers choose their travel routes with respect to the line system, i.e., the model integrates line planning and passenger routing. In fact, passengers are routed in a system optimum, which, in this case, is also a user equilibrium, i.e., the routing is relatively realisitc.

As far as I know, this paper was the first in proposing and investigating a mathematical optimization approach to this integrated problem. We show that the LP-relaxation of our line planning model can be solved in polynomial time, if the lines have logarithmic lengths. A computational study for ViP Verkehr in Potsdam GmbH, the public transit company of Potsdam, shows that it is indeed possible to compute line plans for medium sized cities in this way. Further publications on line planning (not included here) are Borndörfer & Liebchen (2008) [1]; Borndörfer, Grötschel & Pfetsch (2008) [12]; Borndörfer, Neumann & Pfetsch (2008) [13]; Pfetsch & Borndörfer (2006) [15]; Torres et al. (2008) [16, 17].

*

There are several people that played an important role in the development of this thesis. I want to thank Martin Grötschel, who got me into combinatorial optimization, taught me a lot, and had the vision to bring integer programming into traffic planning. My friends and associates Andreas Löbel and Steffen Weider share these ideas and are ideal partners. Thanks to my other coauthors Marc Pfetsch, Thomas Schlechte, and Robert Weismantel, and to the traffic group at ZIB. Special thanks to Henry Thieme, the former administrative director of ZIB, for supporting me through a long time.

I am indebted to the Zuse Institute Berlin, to BMBF, BMWi, and MATHEON for their financial support, and to the companies Berliner Verkehrsbetriebe AöR, DB Stadtverkehr GmbH, IVU Traffic Technologies AG, Lufthansa Systems Berlin GmbH, Regensburger Verkehrsbetriebe GmbH, and ViP Verkehr in Potsdam GmbH.

I dedicate this thesis to the memory of my father.

Berlin, 21.08.2009, Ralf Borndörfer

References

[1] R. Borndörfer & C. Liebchen. *When Periodic Timetables are Suboptimal*. In J. Kalcsics & S. Nickel, (Eds.), *Operations Research Proceedings 2007*, pp. 449–454. Springer Verlag, Berlin, 2008. ZIB Report 07-29 available at http://opus.kobv.de/zib/volltexte/2007/1056/. Cited on page xiii.

[2] R. Borndörfer & T. Schlechte. *Solving railway track allocation problems*. In J. Kalcsics & S. Nickel, (Eds.), *Operations Research Proceedings 2007*, pp. 117–122. Springer Verlag, Berlin, 2008. ZIB Report 07-20 available at http://opus.kobv.de/zib/volltexte/2007/974/. Cited on page xii.

[3] R. Borndörfer & T. Schlechte. *Balancing efficiency and robustness*. In M. Ehrgott, B. Naujoks, T. Stewart & J. Wallenius, (Eds.), *MCDM for Sustainable Energy and Transportation Systems*, Lecture Notes in Economics and Mathematical Systems. Springer Verlag, Berlin, 2008. ZIB Report 08-22 available at http://opus.kobv.de/zib/volltexte/2008/1105/. Cited on page xii.

[4] R. Borndörfer & R. Weismantel. *Discrete relaxations of combinatorial programs*. Discrete Appl. Math. **112**(1–3):11–26, 2001. ZIB preprint SC 97-54 available at URL http://opus.kobv.de/zib/volltexte/1997/324/. Cited on page ix.

[5] R. Borndörfer, A. Löbel, U. Strubbe & M. Völker. *Zielorientierte Dienstplanoptimierung*. In M. Boltze, (Ed.), *Heureka '99: Optimierung in Verkehr und Transport*, pp. 171–194. VDV, Forschungsgesellschaft für Strassen- und Verkehrswesen, Köln, 1999. ZIB Report 98-41 available at http://opus.kobv.de/zib/volltexte/1998/385/. Cited on page xii.

[6] R. Borndörfer, M. Grötschel & A. Löbel. *Der schnellste Weg zum Ziel*. In M. Aigner & E. Behrends, (Eds.), *Alles Mathematik*, pp. 45–76. Vieweg Verlag, Braunschweig/Wiesbaden, 2000. ISBN 3-528-03131-X. ZIB Report 99-32 available at http://opus.kobv.de/zib/volltexte/1999/421/. Cited on page viii.

[7] R. Borndörfer, A. Löbel & S. Weider. *Integrierte Umlauf- und Dienstplanung im öffentlichen Nahverkehr*. In M. Boltze, (Ed.), *Heureka '02*, pp. 77–98. VDV, Forschungsgesellschaft für Strassen- und Verkehrswesen, Köln, 2002. ZIB Report 02-10 available at http://opus.kobv.de/zib/volltexte/2002/678/. Cited on page xii.

[8] R. Borndörfer, M. Grötschel & A. Löbel. *Duty scheduling in public transit*. In W. Jäger & H.-J. Krebs, (Eds.), *MATHEMATICS – Key Technology for the Future*, pp. 653–674. Springer Verlag, Berlin,

2003. ISBN 3-540-44220-0. ZIB Report 01-02 available at http://opus.kobv.de/zib/volltexte/2001/629/. Cited on page xii.

[9] R. Borndörfer, M. Grötschel & M. E. Pfetsch. *Public transport to the fORe!* OR/MS Today **33**(2):30–40, 2006. ZIB Report 05-22 available at http://opus.kobv.de/zib/volltexte/2005/856/. Cited on page viii.

[10] R. Borndörfer, U. Schelten, T. Schlechte & S. Weider. *A column generation approach to airline crew scheduling.* In H. Haasis, (Ed.), *Operations Research Proceedings 2005*, pp. 343–348. Springer Verlag, Berlin, 2006. ZIB Report 05-37 available at http://opus.kobv.de/zib/volltexte/2005/871/. Cited on page xii.

[11] R. Borndörfer, M. Grötschel & U. Jaeger. *Planungsprobleme im öffentlichen Verkehr.* In M. Grötschel, K. Lucas & V. Mehrmann, (Eds.), *PRODUKTIONSFAKTOR MATHEMATIK – Wie Mathematik Technik und Wirtschaft bewegt*, acatech DISKUTIERT, pp. 127–153. acatech – Deutsche Akademie der Technikwissenschaften und Springer, 2008. ISBN 978-3-540-89434-6. ZIB Report 08-20 available at http://opus.kobv.de/zib/volltexte/2008/1103/. Cited on page viii.

[12] R. Borndörfer, M. Grötschel & M. E. Pfetsch. *Models for line planning in public transport.* In M. Hickman, P. Mirchandani & S. Voss, (Eds.), *Computer-aided Systems in Public Transport (CASPT 2004)*, vol. 600 of *Lecture Notes in Economics and Mathematical Systems*, pp. 363–378. Springer Verlag, Berlin, 2008. ZIB Report 04-10 available at http://opus.kobv.de/zib/volltexte/2004/786/. Cited on page xiii.

[13] R. Borndörfer, M. Neumann & M. E. Pfetsch. *Angebotsplanung im öffentlichen Nahverkehr.* In M. Boltze, (Ed.), *Heureka '08*. VDV, Forschungsgesellschaft für Strassen- und Verkehrswesen, Köln, 2008. ZIB Report 08-04 available at http://opus.kobv.de/zib/volltexte/2008/1084/. Cited on page xiii.

[14] R. Borndörfer, A. Mura & T. Schlechte. *Vickrey auctions for railway tracks.* In B. Fleischmann, K. H. Borgwardt, R. Klein & A. Tuma, (Eds.), *Operations Research Proceedings 2008*, pp. 551–556. Springer Verlag, Berlin, 2009. ZIB Report 08-34 available at http://opus.kobv.de/zib/volltexte/2008/1122/. Cited on page xii.

[15] M. E. Pfetsch & R. Borndörfer. *Routing in line planning for public transportation.* In H. Haasis, (Ed.), *Operations Research Proceedings 2005*, pp. 405–410. Springer Verlag, Berlin, 2006. ZIB Report 05-36 available at http://opus.kobv.de/zib/volltexte/2005/870/. Cited on page xiii.

[16] L. M. Torres, R. Torres, R. Borndörfer & M. E. Pfetsch. *Line planning on paths and tree networks with applications to the Quito*

Trolebus system. In M. Fischetti & P. Widmayer, (Eds.), *ATMOS 2008 - 8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*, Dagstuhl, Germany, 2008. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany. ZIB Report 08-35 available at http://drops.dagstuhl.de/opus/volltexte/2008/1580. Cited on page xiii.

[17] L. M. Torres, R. Torres, R. Borndörfer & M. E. Pfetsch. On the line planning problem in tree networks. ZIB Report 08-52, ZIB, Takustr. 7, 14195 Berlin, 2008. URL http://opus.kobv.de/zib/volltexte/2008/1147/. Cited on page xiii.

Paper I

Set Packing Relaxations of Some Integer Programs

R. Borndörfer & R. Weismantel.
Set packing relaxations of some integer programs.
Mathematical Programming **88**:425–450, 2000.

Abstract. This paper is about *set packing relaxations* of combinatorial optimization problems associated with acyclic digraphs and linear orderings, cuts and multicuts, and set packings themselves. Families of inequalities that are valid for such a relaxation as well as the associated separation routines carry over to the problems under investigation.

Keywords. Set packing, polyhedral combinatorics, cutting planes, integer programming

Mathematics Subject Classification (MSC 2000). 90C10, 90C57

1 Introduction

This paper is about *relaxations* of certain combinatorial optimization problems in the form of a set packing problem and the use of such relaxations in a polyhedral approach.

The *set packing* or *stable set problem* (SSP) is to find, in a graph $G = (V, E)$ with node weights c, a *set packing* or *stable set*, i.e., a set of pairwise non-adjacent nodes, of maximum weight. Set packing problems are among the best studied combinatorial optimization problems with beautiful theories connecting this area of research to Fulkerson's antiblocking theory, see Fulkerson (1971) [12], the theory of perfect graphs, see Lovász (1971) [19], perfect and balanced matrices, see Padberg (1973) [26] and Berge (1971) [4],

semidefinite programming, see Grötschel, Lovász & Schrijver (1988) [16], and other fields, see Grötschel, Lovász & Schrijver (1988) [16] for a survey. Likewise, the *set packing polytope*, i.e., the convex hull of all set packings of a graph, see, e.g., Padberg (1973) [26], plays a prominent role in polyhedral combinatorics not only because large classes of (facet defining) inequalities are known. Perhaps even more important, many of them can be separated in polynomial time. In particular, odd cycle, orthonormal representation, and (superclasses of) odd antihole constraints are polynomial time separable, see again Grötschel, Lovász & Schrijver (1988) [16] and Lovász & Schrijver (1991) [20].

Our aim in this paper is to transfer some of these results to other combinatorial optimization problems. We show that the acyclic subdigraph and the linear ordering problem, the max cut, the k-multicut, and the clique partitioning problem, and the set packing problem itself have interesting *combinatorial relaxations* in form of a set packing problem. Families of inequalities that are valid for these relaxations and the associated separation routines carry over to the problems under investigation. The procedure is an application of a more general method to construct relaxations of combinatorial optimization problems by means of *affine transformations* that we discuss in a forthcoming paper. This method is in the tradition of projection techniques such as Balas & Pulleyblank (1989) [1], Pulleyblank & Shepherd (1993) [29], and Chopra & Rao (1994) [7, 8] and, in particular, Padberg & Sung (1991) [28] approach to use affine transformations for the comparison of TSP formulations.

The paper is subdivided into four parts. Section 2 lists notation and results on set packing for future reference. It also recalls three earlier frameworks that give results similar, yet not identical, to ours; we shall point out similarities and differences throughout the article. Section 3 is devoted to a study of the acyclic subdigraph and the linear ordering problem, see Grötschel, Jünger & Reinelt (1985) [14, 15]. A main result in this section is that a class of Möbius ladders with dicycles of *arbitrary* length belong to a (larger) class of odd cycles of an appropriate set packing relaxation; this superclass is polynomial time separable. Section 4 deals with set packing relaxations of the clique partitioning, the k-multicut, and the max cut problem, see Grötschel & Wakabayashi (1990) [13] and Deza & Laurent (1997) [10]. We introduce a class of "inequalities from odd cycles of lower triangle inequalities" that contains the 2-chorded cycle inequalities. Section 5 treats the set packing problem *itself*. We show, in particular, that the wheel inequalities of Barahona & Mahjoub (1994) [3] and Cheng & Cunningham (1997) [6] are odd cycle inequalities of a suitable set packing relaxation. We also introduce a new family of facet defining inequalities for the set packing polytope: The "cycle of cycles" inequalities. This class can be separated in polynomial time.

2 Terminology

The subsequent sections resort to the following notation and results about the set packing problem. The *set packing problem* for a graph G with node weights c can be formulated as an *integer program*

$$\max c^T x \qquad Ax \leq \mathbf{1}, \quad x \in \{0,1\}^V, \tag{SSP}$$

where $A = A(G) \in \{0,1\}^{E \times V}$ is the *edge-node incidence matrix* of G and $\mathbf{1}$ a *vector of all ones* of compatible dimension. Associated with this program is the *stable set polytope* or *set packing polytope*

$$P_I := \operatorname{conv}\big\{\chi^S : \ S \text{ is a stable set in } G\big\} = \operatorname{conv}\big\{x \in \{0,1\}^n : \ Ax \leq \mathbf{1}\big\},$$

the convex hull of all incidence vectors of stable sets in G or, equivalently, of all solutions of (SSP). Occasionally, we will denote this polytope also by $P_I(G)$. For technical reasons, we will actually not work with the stable set polytope P_I itself, but with its *antidominant*

$$\breve{P}_{\mathrm{SSP}} := P_I - \mathbb{R}^V_+ = \big\{x \in \mathbb{R}^V : \exists y \in P_I : x \leq y\big\}.$$

This construction allows to consider vectors with arbitrary negative coordinates *without* destroying the polyhedral structure of P_I: Obviously, the valid inequalities for $\breve{P}_{\mathrm{SSP}}$ are exactly the valid inequalities for P_I of the form $a^{\mathrm{T}}x \leq \alpha$ *with non-negative coefficients*. Since the stable set polytope P_I is down-monotone, its non-trivial valid inequalities all have non-negative coefficients. We can thus work with $\breve{P}_{\mathrm{SSP}}$ as well as with P_I.

We list some terminology to state five fundamental results on P_I (or $\breve{P}_{\mathrm{SSP}}$). A *clique* in a graph G is a set of pairwise adjacent nodes. A *walk* consists of a sequence $v_1, e_1, v_2, e_2, \ldots, e_k, v_{k+1}$ of nodes v_i and edges e_i such that $e_i = v_i v_{i+1}$, $i = 1, \ldots, k$. A *closed walk* has $v_1 = v_{k+1}$. A walk is a *path* if all its nodes are different, except possibly the first and the last, which can be identical; in this case, the path is called a *cycle*. The analogous concepts for digraphs, with the additional stipulation that all arcs have to be "oriented in the same direction", are called *diwalk*, *closed diwalk*, *dipath*, and *dicycle*, respectively. A (di)walk (and hence a (di)path or (di)cycle) is *odd* if k is odd, i.e., if it contains an odd number of nodes, *even* otherwise. An edge that joins two nodes of a cycle, but is not a member of the cycle, is a *chord*. A *2-chord* is a chord of the form $v_i v_{i+2}$ (indices $> k$ taken modulo k). A chordless cycle is a *hole*. For convenience of notation, we will occasionally consider (di)paths and (di)cycles as *sets* of nodes, edges, or arcs, and we will denote edges *as well as* arcs with the symbols ij and (i,j); the latter will be used in cases like $(i, i+1)$. Finally, $\operatorname{supp}(x) = \{i \in V : x_i \neq 0\}$ is the *support* of a vector $x \in \mathbb{R}^V$.

The results on P_I that we need are summarized in the following two theorems; we state them for $\breve{P}_{\text{SSP}}$.

Theorem 2.1 (Clique, Odd Cycle, and Orthonormal Representation Inequalities, Padberg (1973) [26], Grötschel, Lovász & Schrijver (1988) [16]).
Let $G = (V, E)$ be a graph and $\breve{P}_{\text{SSP}}$ the antidominant of the associated set packing polytope.

(i) If Q is a clique in G, the clique inequality $\sum_{i \in Q} x_i \leq 1$ *is valid for $\breve{P}_{\text{SSP}}$; it is facet defining if and only if Q is a maximal clique (with respect to set inclusion).*

(ii) If C is an odd cycle in G, the odd cycle inequality $\sum_{i \in C} x_i \leq (|C| - 1)/2$ *is valid for $\breve{P}_{\text{SSP}}$.*

(iii) Let $u \in \mathbb{R}^V$ be an orthonormal representation *of G, i.e., $|u_i| = 1$ for all $i \in V$ and $u_i^{\mathrm{T}} u_j = 0$ holds for all $ij \notin E$, and let $c \in \mathbb{R}^V$ be an additional arbitrary vector with $|c| = 1$. The* orthonormal representation inequality $\sum_{i \in V} (c^{\mathrm{T}} u_i)^2 x_i \leq 1$ *is valid for $\breve{P}_{\text{SSP}}$.*

Separation of clique inequalities is $\mathcal{NP}$-hard. But the clique inequalities belong to the more general class of orthonormal representation inequalities which can be separated in polynomial time.

Theorem 2.2 (Orthonormal Representation & Odd Cycle Inequalities, Grötschel, Lovász & Schrijver (1988) [16]).
Let $G = (V, E)$ be a graph, $\breve{P}_{\text{SSP}}$ the antidominant of the associated set packing polytope, and $x \in \mathbb{Q}^V$. Suppose that $x_i + x_j \leq 1$ holds for all edges $ij \in E$. Then:

(i) Orthonormal representation inequalities violated by x can be separated in polynomial time.

(ii) Odd cycle inequalities violated by x can be separated in polynomial time.

The literature has three frameworks that give results similar to this article:

- The *independence system approach*, see Nemhauser & Trotter (1973) [24], Sekiguchi (1983) [32], Euler, Jünger & Reinelt (1987) [11], Laurent (1989) [18], Nobili & Sassano (1989) [25], and others.
- The *transitive packing approach* of Müller (1996) [21], Müller & Schulz (1995) [22]; Müller & Schulz (1996) [23], Schulz (1996) [31].
- The $\{0, \frac{1}{2}\}$-*Chvátal-Gomory cuts* of Caprara & Fischetti (1996) [5].

We recall these concepts for later comparisons with our approach. We do neither discuss the relation to projection techniques nor to Padberg & Sung (1991) [28] here; this would blast the scope of this article.

Independence System Approach. An *independence system* (IS) arises from a set system $\mathfrak{C} \subseteq 2^V$ of *circuits* on a finite ground set V of elements i with weights w_i. A subset I of V is *independent* if it contains no circuit. The *independence system problem* (ISP) asks for an independent set of maximum weight. An integer programming formulation of the ISP reads

$$\begin{aligned} \max \quad & \sum_{i \in V} w_i x_i & & \text{(ISP)} \\ \text{(i)} \quad & \sum_{i \in C} x_i \leq |C| - 1 & \forall \text{ circuits } C \in \mathfrak{C} \\ \text{(ii)} \quad & x_i \in \{0,1\} & \forall\, i \in V. \end{aligned}$$

The set packing problem, the acyclic subdigraph problem, and the knapsack problem are prominent examples of independence system problems, others, such as the set covering problem, can be transformed into this setting, see, e.g., Laurent (1989) [18] or Nobili & Sassano (1989) [25]. This means that, in principle, these problems and their associated polytopes can be understood completely in terms of the IS framework.

Facets for the *independence system polytope* P_{ISP} include *generalized clique*, *generalized cycle*, *generalized anticycle*, and *generalized antiweb inequalities*, see, e.g., Nemhauser & Trotter (1973) [24], Sekiguchi (1983) [32], Euler, Jünger & Reinelt (1987) [11], Laurent (1989) [18], Nobili & Sassano (1989) [25]. These results unify and/or extend individual results on special independence system polytopes such as the matroid and the set packing polytope (Nemhauser & Trotter (1973) [24], Laurent (1989) [18]), the knapsack polytope (Padberg (1975) [27], Laurent (1989) [18]), the acyclic subdigraph polytope (Euler, Jünger & Reinelt (1987) [11], Nobili & Sassano (1989) [25]), etc.

On the algorithmic side, virtually no polynomial separation algorithms for general classes of IS inequalities seem to be known. There are, on the contrary, many negative results on the separation of subclasses, e.g., the $\mathcal{NP}$-hardness of the separation of fence inequalities, which happen to be generalized clique inequalities for the acyclic subdigraph polytope, see Müller (1996) [21]. To the best of our knowledge, the only tractable general IS inequalities are those that happen to be polynomial time separable $\{0, \frac{1}{2}\}$-Chvátal-Gomory cuts, see below.

Transitive Packing Approach. An *extended set system* $\mathfrak{D} \subseteq 2^{V \times V}$ consists of pairs $\big(C, \operatorname{tr}(C)\big)$ of (hyper)edges $C \subseteq V$ and associated sets of *tran-*

sitive elements $\mathrm{tr}(C) \subseteq V \setminus C$. V is a finite ground set of elements i with weights w_i. A subset I of V is a *transitive packing* (TP) if it contains, for every $\big(C, \mathrm{tr}(C)\big)$ such that $C \subseteq I$, at least one element from $\mathrm{tr}(C)$. The *transitive packing problem* (TPP) is to find a transitive packing of maximum weight. An integer programming formulation is

$$\begin{aligned} &\max \quad \sum_{i \in V} w_i x_i && && \text{(TPP)} \\ &\text{(i)} \quad \sum_{i \in C} x_i - \sum_{i \in \mathrm{tr}(C)} x_i \leq |C| - 1 && \forall \ \big(C, \mathrm{tr}(C)\big) \in \mathfrak{D} \\ &\text{(ii)} \quad x_i \in \{0,1\} && \forall \ i \in V. \end{aligned}$$

Every ISP with circuit system $\mathfrak{C}$ is a TPP with extended set system $\mathfrak{D} := \{(C, \emptyset) : C \in \mathfrak{C}\}$. This means that transitive packing captures all independence system problems. It subsumes, however, additional combinatorial optimization problems that are not ISPs, among them the clique partitioning problem, the max cut problem, the transitive acyclic subdigraph problem, and the interval order problem, see Müller & Schulz (1996) [23], Schulz (1996) [31].

All types of IS inequalities have been extended to classes of valid inequalities for the *transitive packing polytope* P_{TPP}. This unifies and/or generalizes results on inequalities for individual transitive packing and related polytopes, among them the clique partitioning polytope and several subpolytopes of the acyclic subdigraph polytope, namely, the transitive acyclic subdigraph polytope, the interval order polytope of a digraph, and the linear ordering polytope, see Müller (1996) [21], Müller & Schulz (1995) [22]; Müller & Schulz (1996) [23], Schulz (1996) [31].

While the general classes of TP inequalities are as difficult to separate as their IS antecedents, polynomial time separation algorithms are known for interesting subclasses of generalized cycle inequalities for special transitive packing polytopes. Polynomial time separable instances of these *weak odd closed walk inequalities* (later called *weak generalized* $(k, 2)$*-cycle inequalities*) include (and generalize) well studied classes such as the 2-chorded cycle and the odd wheel inequalities for the clique partitioning polytope, see Müller (1996) [21], and most of the known types of inequalities for the linear ordering polytope, see Müller & Schulz (1995) [22]. Polynomial time separation of weak odd closed walk inequalities carries over to these classes.

$\{0, \frac{1}{2}\}$-Chvátal-Gomory Cuts. This concept applies to integer linear systems

$$Ax \leq b, \quad x \in \mathbb{Z}^n, \qquad \text{(IP)}$$

where $A \in \mathbb{Z}^{m\times n}$ and $b \in \mathbb{Z}^m$. A $\{0, \frac{1}{2}\}$-*Chvátal-Gomory cut* for IP is an inequality of the form

$$\lambda^{\mathrm{T}} A x \leq \lfloor \lambda^{\mathrm{T}} b \rfloor,$$

where $\lambda^{\mathrm{T}} A \in \mathbb{Z}^n$ and $\lambda \in \{0, \frac{1}{2}\}^m$, hence the name. Separation of $\{0, \frac{1}{2}\}$-Chvátal-Gomory cuts, although $\mathcal{NP}$-hard in general, is polynomial in important cases, e.g., when the system IP has at most two odd coefficients per row , see Caprara & Fischetti (1996) [5].

LU weakening is a way to proceed when the system IP does not have this property. We may assume that IP contains the bounds $0 \leq x \leq u$, $u \in (\mathbb{Z} \cup \{\infty\})^n$. Denote by $O_i := \{1 \leq j \leq n : a_{ij} \text{ odd }\}$ the index set of the odd coefficients in row i for $i = 1, \ldots, m$. With each row i of IP with $|O_i| \geq 3$ one associates $2^{|O_i|(|O_i|-1)/2}$ (weaker) inequalities by adding upper and lower bound constraints

$$(A_i.x \leq b_i) + \sum_{j\in L_i} (-x_j \leq 0) + \sum_{j\in U_i} (x_j \leq u_j),$$

where (L_i, U_i) runs through all possible partitions of $O_i \setminus \{k, \ell\}$, for all index pairs of odd coefficients $k, \ell \in O_i$, $k \neq \ell$. Note that all of these so-called *LU weakenings* of row i have exactly 2 odd coefficients. Let IP$'$ be the system that arises from IP by replacing each row with $|O_i| \geq 3$ by all its LU weakenings. IP$'$ has, in general, an exponential number of rows. Caprara & Fischetti (1996) [5], however, have shown that one can separate in time polynomial in the encoding length of the *original system* IP over all $\{0, \frac{1}{2}\}$-Chvátal-Gomory cuts that can be obtained from its LU weakening IP$'$.

Well known classes of inequalities for combinatorial optimization problems including the clique partitioning problem, the acyclic subdigraph problem, and the asymmetric travelling salesman problem are $\{0, \frac{1}{2}\}$-Chvátal-Gomory cuts from appropriate IP formulations or their LU weakenings. Whenever IP has polynomial size, these cuts can be separated in polynomial time. This applies, among others, to the 2-chorded cycle and the odd wheel inequalities for the clique partitioning polytope, and to a large class of facet defining Möbius ladder inequalities for the linear ordering polytope.

3 The Acyclic Subdigraph and the Linear Ordering Problem

Our aim in this section is to construct a set packing relaxation of the acyclic subdigraph and the linear ordering problem in a space of exponential dimension. It will turn out that clique and odd cycle inequalities of this relaxation

give rise to (and generalize) several classes of inequalities for the acyclic subdigraph and the linear ordering problem, namely, fence and Möbius ladder inequalities. References are Grötschel, Jünger & Reinelt (1985) [14] for the acyclic subdigraph problem and Grötschel, Jünger & Reinelt (1985) [15] for the linear ordering problem, see also the monographs of Jünger (1985) [17] and Reinelt (1985) [30].

The acyclic subdigraph and the linear ordering problem involve a complete digraph $D_n = (V, A)$ on n nodes with integer weights w_a on its arcs $a \in A$. An *acyclic arc set* in A contains no dicycle. The *acyclic subdigraph problem* (ASP) asks for an acyclic arc set of maximum weight. Acyclic arc sets that contain, for any pair of nodes i and j, either the arc ij or the arc ji, are called *tournaments*. The *linear ordering problem* (LOP) is to find a tournament of maximum weight. Integer programming formulations for the ASP and the LOP read as follows:

$$
\begin{array}{lll}
\max & \displaystyle\sum_{ij \in A} w_{ij} x_{ij} & \\
\text{(ii)} & \displaystyle\sum_{ij \in C} x_{ij} \leq |C| - 1 & \forall \text{ dicycles } C \subseteq A \\
\text{(iii)} & -x_{ij} \leq 0 & \forall\, ij \in A \\
\text{(iv)} & x_{ij} \leq 1 & \forall\, ij \in A \\
\text{(v)} & x_{ij} \in \mathbb{Z} & \forall\, ij \in A
\end{array}
\qquad \text{(ASP)}
$$

$$
\begin{array}{lll}
\max & \displaystyle\sum_{ij \in A} w_{ij} x_{ij} & \\
\text{(i)} & x_{ij} + x_{ji} = 1 & \forall\, i, j \in V, i \neq j \\
\text{(ii)} & \displaystyle\sum_{ij \in C} x_{ij} \leq |C| - 1 & \forall \text{ dicycles } C \subseteq A : |C| = 3 \\
\text{(iii)} & -x_{ij} \leq 0 & \forall\, ij \in A \\
\text{(iv)} & x_{ij} \leq 1 & \forall\, ij \in A \\
\text{(v)} & x_{ij} \in \mathbb{Z} & \forall\, ij \in A.
\end{array}
\qquad \text{(LOP)}
$$

It can be shown that (ASP) is a relaxation of (LOP) and, even more, that the *linear ordering polytope* P_{LOP} is a face of the *acyclic subdigraph polytope* P_{ASP}. In particular, all inequalities that are valid for P_{ASP} are also valid for P_{LOP}. Two such classes of inequalities for both the ASP and the LOP are the *k-fence* and the *Möbius ladder inequalities*, see Grötschel, Jünger & Reinelt (1985) [14].

A *simple k-fence* has disjoint sets of "upper" and "lower" nodes $\{u_1, \ldots, u_k\}$ and $\{l_1, \ldots, l_k\}$ that are joined by a set of k *pales* $P_i^{\downarrow} := \{u_i l_i\}$, $i = 1, \ldots, k$. The pales are oriented "downward". The k-fence is completed by adding all "upward" *pickets* $P_{ij}^{\uparrow} := \{l_i u_j\}$ with the exception of the antiparallel pales. A (general) *k-fence* is obtained from a simple one by repeated subdivision

of arcs, replacing pale and picket *arcs* by *dipaths*. Figure 1 shows a simple 4-fence.

A *Möbius ladder* consists of an odd number $2k+1$ of dicycles $C_0, \ldots, C_{2k}$ such that C_i and C_{i+1} (indices taken modulo $2k+1$) have a dipath P_i in common, see Figure 2.

Fences and Möbius ladders give rise to valid inequalities for P_{ASP}: For a k-fence F_k and a Möbius ladder M of $2k+1$ dicycles we have

$$\sum_{ij \in F_k} x_{ij} \leq |F_k| - k + 1 \quad \text{and} \quad \sum_{i=0}^{2k} \sum_{ij \in C_i \setminus P_i} x_{ij} \leq \left(\sum_{i=0}^{2k} |C_i \setminus P_i| \right) - (k+1).$$

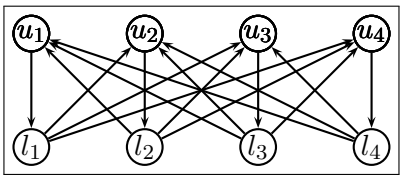


Figure 1: A 4-Fence.

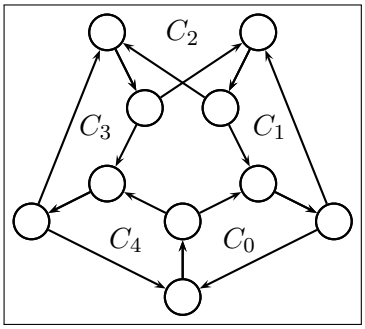


Figure 2: A Möbius Ladder of 5 Dicycles.

A Möbius ladder inequality as above has coefficients larger than one if an arc is contained in more than one of the dipaths $C_i \setminus P_i$. In this situation of *arc repetition*, there is a difference to Grötschel, Jünger & Reinelt (1985) [14] original definition, where the coefficients take only values of zero and one, the right hand side is smaller, and the Möbius ladder must meet a number of additional technical requirements to support a valid inequality. The definitions coincide if and only if there is no arc repetition.

We will show now that fences and Möbius ladders are cliques and odd cycles, respectively, in an (exponential) *conflict graph* $\mathfrak{G}(D_n) = (\mathfrak{V}, \mathfrak{E})$. $\mathfrak{G}$ has the set of all acyclic arc sets of D_n as its nodes. We draw an edge $\mathfrak{uv}$ between two acyclic arc-set nodes $\mathfrak{u}$ and $\mathfrak{v}$ if their union contains a dicycle. In this case, we say that $\mathfrak{u}$ *and* $\mathfrak{v}$ *are in conflict*, because they can not be simultaneously contained in a solution to (ASP).

It is now easy to identify the fences and Möbius ladders with cliques and odd cycles of $\mathfrak{G}$. To obtain a k-fence F_k, we look at the k acyclic arc sets F_k^i that consist of a pale $P_i^{\downarrow}$ and the pickets $P_{ij}^{\uparrow}$ that go up from l_i, for $i = 1, \ldots, k$. We call such a configuration a *k-fork*. Any two forks F_k^i and F_k^j, $i \neq j$, are in conflict (they contain a dicycle). Hence, all of them together form

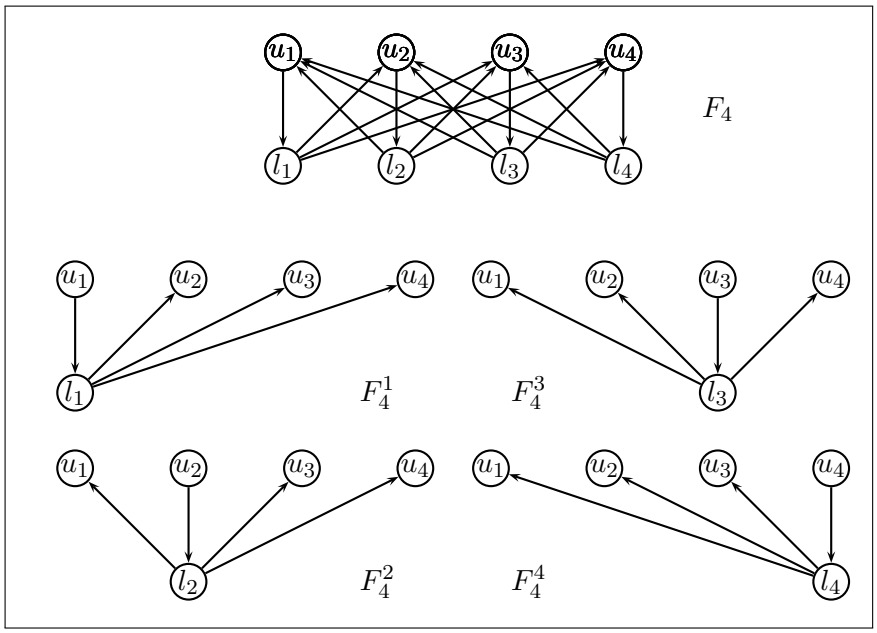


Figure 3: A Fence Clique of Forks.

a clique in $\mathfrak{G}$. Figure 3 illustrates this construction. Likewise, the Möbius ladders correspond to odd cycles of conflicting dipaths, namely, the dipaths $C_i \setminus P_i$, see Figure 4.

The next step to obtain the fence and the Möbius ladder inequalities from the clique and odd cycle inequalities of the (antidominant of the) set packing polytope $\breve{P}_{\mathrm{SSP}}(\mathfrak{G})$ associated with the conflict graph $\mathfrak{G}$, is to construct a set packing relaxation of the ASP. To this purpose, consider the function π :

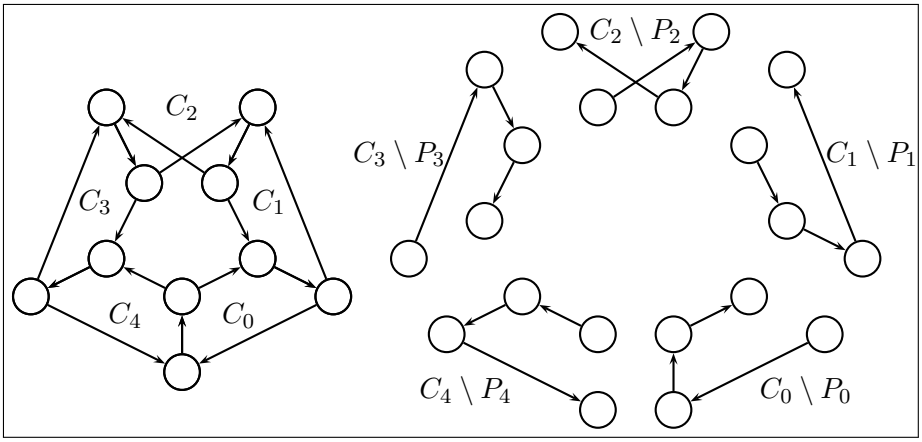


Figure 4: A Möbius Cycle of Dipaths.

$\mathbb{R}^A \to \mathbb{R}^{\mathfrak{V}}$ defined as

$$\pi_{\mathfrak{v}}(x) := \sum_{ij \in \mathfrak{v}} x_{ij} - (|\mathfrak{v}| - 1) \qquad \forall \text{ acyclic arc sets } \mathfrak{v} \in \mathfrak{V}. \tag{1}$$

$\pi(x)$ is integral for all integral $x \in \mathbb{R}^A$. Moreover, for every incidence vector $x \in P_{\text{ASP}}$ of an acyclic arc set $\text{supp}(x)$ in D_n, we have that $\pi(x)$ attains its maximum value of one in component $\pi_{\mathfrak{v}}(x)$ if and only if $\mathfrak{v}$ is contained in $\text{supp}(x)$. Since two conflicting acyclic arc sets can not simultaneously be contained in $\text{supp}(x)$, we have that

$$\mathfrak{uv} \in \mathfrak{E} \iff \pi_{\mathfrak{u}}(x) + \pi_{\mathfrak{v}}(x) \leq 1 \qquad \forall\, x \in P_{\text{ASP}} \cap \mathbb{Z}^A$$

and, by convexity, also for all $x \in P_{\text{ASP}}$. This argument proves that $\breve{P}_{\text{SSP}}(\mathfrak{G})$ is a set packing relaxation of P_{ASP} in the sense that

Lemma 3.1 (Set Packing Relaxation of the ASP).
$\pi\left(P_{\text{ASP}}\right) \subseteq \breve{P}_{\text{SSP}}\big(\mathfrak{G}(D_n)\big)$.

Note that it is not possible to replace $\breve{P}_{\text{SSP}}$ with P_I, because the components of π can take negative values. More precisely, $\pi(x)$ is in general *not* the incidence vector of a stable set in $\breve{P}_{\text{SSP}}(\mathfrak{G})$, but $\max\{\mathbf{0}, \pi(x)\}$, with the maximum taken in every component, is.

Lemma 3.1 allows us to *expand* an inequality $a^T\pi \leq \alpha$ which is valid for $\breve{P}_{\text{SSP}}$ into the inequality $a^T\pi(x) \leq \alpha$ which is valid for P_{ASP}. Our next theorem states that, with this terminology, the fence and Möbius ladder inequalities are expansions of clique and odd cycle inequalities, respectively.

Theorem 3.2 (Fence and Möbius Ladder Inequalities).
Let D_n be the complete digraph on n nodes, P_{ASP} the corresponding acyclic subdigraph polytope, $\mathfrak{G}$ the conflict graph associated with D_n, and $\breve{P}_{\text{SSP}}(\mathfrak{G})$ the set packing relaxation of P_{ASP}.

(i) Every k-fence inequality for P_{ASP} is the expansion of a clique inequality for $\breve{P}_{\text{SSP}}(\mathfrak{G})$.

(ii) Every Möbius ladder inequality for P_{ASP} is the expansion of an odd cycle inequality for $\breve{P}_{\text{SSP}}(\mathfrak{G})$.

Proof. (i) Let F_k be a k-fence. The forks F_k^i, $i = 1, \ldots, k$, defined on page 9, are acyclic arc sets and they form a clique in $\mathfrak{G}$, see the discussion on the previous page. An expansion of the corresponding clique inequality yields

the desired k-fence inequality:

$$\begin{aligned}
& \sum_{i=1}^{k} \pi_{F_k^i}(x) \leq 1 \\
\iff & \sum_{i=1}^{k} \left(\sum_{ij \in F_k^i} x_{ij} - (|F_k^i| - 1) \right) = \sum_{ij \in F_k} x_{ij} - |F_k| + k \leq 1 \\
\iff & \sum_{ij \in F_k} x_{ij} \leq |F_k| - k + 1.
\end{aligned}$$

(ii) Let M be a Möbius ladder consisting of an odd number $2k+1$ of dicycles $C_0, \ldots, C_{2k}$ such that C_i and C_{i+1} have a dipath P_i in common. The argument on page 10 showed that the dipaths $C_i \setminus P_i$ form an odd cycle of $2k+1$ acyclic arc sets in $\mathfrak{G}$. Expanding the corresponding odd cycle inequality for $\breve{P}_{\mathrm{SSP}}(\mathfrak{G})$, one obtains the Möbius ladder inequality for M:

$$\begin{aligned}
& \sum_{i=0}^{2k} \pi_{C_i \setminus P_i}(x) \leq k \\
\iff & \sum_{i=0}^{2k} \left(\sum_{ij \in C_i \setminus P_i} x_{ij} - (|C_i \setminus P_i| - 1) \right) \leq k \\
\iff & \sum_{i=0}^{2k} \sum_{ij \in C_i \setminus P_i} x_{ij} \leq \left(\sum_{i=0}^{2k} |C_i \setminus P_i| \right) - (k+1).
\end{aligned}$$

□

Looking at the *separation of Möbius ladder inequalities*, we notice that the construction that we just presented to prove Theorem 3.2 (ii) yields a class of *odd cycle of dipath inequalities* that coincides with the Möbius ladder inequalities as defined in this paper and subsumes Grötschel, Jünger & Reinelt (1985) [14] Möbius ladder inequalities without arc repetition. Generalizing this class further by allowing the paths $C_i \setminus P_i$ to intersect themselves on nodes and/or arcs, i.e., by substituting in the definition of a Möbius ladder on page 9 *diwalk* for dipath and *closed diwalk* for dicycle, we obtain an even larger class of *odd cycle of diwalk inequalities* for the acyclic subdigraph polytope. Note that these inequalities do in general *not* correspond to odd cycles of conflicting acyclic arc sets in the graph $\mathfrak{G}$, because diwalks do not have to be acyclic (they may contain dicycles). This obstacle can be overcome by extending $\mathfrak{G}$ in an appropriate way (including only certain relevant diwalks). At this point, however, we do not want to enter this formalism and defer the details of the extension to the proof of Theorem 3.3.

We can devise a polynomial time separation algorithm for odd cycle of diwalk inequalities, even though the number of diwalks is, in fact, infinite and their

length is not even bounded. The idea is to construct a *most violated* cycle of diwalks out of properly interlinked *longest* diwalks. Suppose that M is an odd cycle of diwalks (we want to denote these diwalks with a slight extension of our notation by $C_i \setminus P_i$) that induces a violated inequality, and consider the diwalk P_i linking the two (successive) closed diwalks C_i and C_{i+1}. Rearranging, we can isolate the contribution of P_i in the constraint as

$$|P_i| - \sum_{ij \in P_i} x_{ij} < \sum_{j \neq i+1} \Big(\sum_{ij \in C_j \setminus P_j} x_{ij} - |C_j \setminus P_j| \Big) + \sum_{ij \in C_{i+1} \setminus (P_i \cup P_{i+1})} x_{ij} - |C_{i+1} \setminus (P_{i+1} \cup P_i)| + (k+1).$$

(Here, all sets are supposed to be multisets. We have $<$ because the constraint is, by assumption, violated.)

Replacing P_i with a diwalk P that has the same endpoints, but is shorter with respect to the length function

$$|P| - \sum_{ij \in P} x_{ij} = \sum_{ij \in P} (1 - x_{ij}), \tag{2}$$

we get a more violated cycle of diwalks inequality. If we think of any closed diwalk C_i as being composed out of four diwalks, namely, the diwalk $P_i^1 := P_i$ that C_i has in common with the succeeding closed diwalk C_{i+1}, the diwalk P_i^2 from P_i^1's head to the diwalk $P_i^3 := P_{i-1}$, that C_i has in common with the preceding closed diwalk C_{i-1}, and the remaining diwalk P_i^4 from P_i^3's head to P_i^1's tail, the same argument holds for any of these diwalks. This observation allows us to show

Theorem 3.3 (Polynomial Separability of Odd Cycle of Diwalk Inequalities).
Let D_n be the complete digraph on n nodes and P_{ASP} the associated acyclic subdigraph polytope. Suppose that $x \in \mathbb{Q}^A$ satisfies the dicycle and bound constraints (ASP) (ii)–(iv). Then:

Odd cycle of diwalk inequalities violated by x can be separated in polynomial time.

Proof. Using Dijkstra's algorithm, we can compute a shortest diwalk $P(u, v)$ with respect to the length (2) from any node u to any node v of D_n. We can assume these diwalks $P(u, v)$ w.l.o.g. to be of polynomial length (actually we could even assume them to be dipaths). This yields a polynomial number of (2)-shortest diwalks of polynomial length and, moreover, (not every, but) *a* most violated cycle of diwalks inequality will consist only of these shortest diwalks.

We can find a set of them forming an odd cycle of diwalks as follows. We think of all diwalks $P(u,v)$ as a possible common diwalk P_i of two successive closed diwalks C_i and C_{i+1} in a cycle of diwalks. To get the diwalks $C_i \setminus P_i$ as the pieces of the cycle, we compute for any two diwalks P_i and P_j the (2)-shortest diwalk $P_i\langle P_j\rangle$ that starts at P_i's head, contains P_j, and ends at P_i's tail. Such a diwalk $P_i\langle P_j\rangle$ will link (on P_j) properly with another diwalk $P_j\langle P_k\rangle$ to form a cycle of diwalks. Computation of the $P_i\langle P_j\rangle$ can be performed in polynomial time and yields, in particular, a polynomial number of $n(n-1)\big(n(n-1)-1\big) = O(n^4)$ diwalks of polynomial length. Again, (not every, but) *a* most violated cycle of diwalks inequality will consist *only* of these diwalks $P_i\langle P_j\rangle$.

We can construct a graph that has these diwalks $P_i\langle P_j\rangle$ as its nodes with node weights equal to the values $\sum_{ij\in P_i\langle P_j\rangle} x_{ij} - \big(|P_i\langle P_j\rangle| - 1\big)$ (see (1) on page 11) and that has all edges of the form $(P_i\langle P_j\rangle, P_j\langle P_k\rangle)$. A most violated cycle of diwalks inequality corresponds to a most violated odd cycle inequality in the $P_i\langle P_j\rangle$-graph. Note that this means, in particular, that there is a most violated cycle of diwalks inequality that consists of a *polynomial* number of diwalks, even though the total number of diwalks is infinite.

The node weights on an edge in the $P_i\langle P_j\rangle$-graph never exceed one because x satisfies the dicycle inequalities (ASP) (ii). Hence, we can find a most violated odd cycle inequality there with the algorithm of Grötschel, Lovász & Schrijver (1988) [16], Lemma 9.1.11. □

Corollary 3.4 (Separation of Möbius Ladder Inequalities).
A superclass of the Möbius ladder inequalities can be separated in polynomial time.

The same technique can be used for the *separation of (general) k-fence inequalities* for fixed, but arbitrary k. Note that one can not enumerate these constraints because general k-fences can contain long dipaths. We sketch the construction. Generalizing k-forks to *k-forkings* by allowing for arbitrary diwalks as pales and pickets, we can construct classes of *clique of k-forking inequalities*, or even *k-forking orthonormal representation constraints*, that subsume the k-fence inequalities. Similar to the proof of Theorem 3.3, one can show that not every, but *a* most violated clique of k-forkings inequality, or *a* most violated k-forking orthonormal representation constraint, will consist of k-forkings that are solely composed from (2)-shortest pickets and pales. For fixed k, these k-forkings can be enumerated in polynomial time and turned into the nodes of a forking conflict graph of polynomial size that has an edge for any two such k-forkings R_1 and R_2 that contain a closed diwalk of the form "pale(R_1)-picket(R_1)-pale(R_2)-picket(R_2)". Associating a weight of $x(R) - (|R| - 1)$ to each forking-node R (where R is a multiset), we enumerate a most violated k-clique or use Grötschel, Lovász & Schrijver

(1988) [16] techniques to separate orthonormal representation constraints. These arguments prove

Theorem 3.5 (Separation of k-Fence Inequalities for Fixed k). *A superclass of the k-fence inequalities can be separated in polynomial time for fixed, but arbitrary k.*

Fence and Möbius ladder inequalities have been discussed in the contexts of independence systems, transitive packings, and $\{0, \frac{1}{2}\}$ Chvátal-Gomory cuts in the literature.

Before we start a comparison of results, we want to draw the readers attention to the following subtle difference between the ASP and the LOP. While the length of the dicycles in a facetial Möbius ladder inequality without node and arc repetition for the *acyclic subdigraph polytope* can be arbitrarily large, the same constraint can only define a facet for the *linear ordering polytope* if the length of each dicycle is either three or four, see Grötschel, Jünger & Reinelt (1985) [14]. This characteristic is responsible for differences in Möbius ladder separation between the LOP and the ASP. No complete facet characterizations for Möbius ladders, neither for P_{ASP} nor for P_{LOP}, are known in the presence of node and/or arc repetition. Reinelt (1985) [30], Definition 2.4.1 gave the best known *sufficient conditions* for a Möbius ladder to be a facet of P_{LOP}; the LOP literature focusses on these (Reinelt's) Möbius ladders.

Euler, Jünger & Reinelt (1987) [11] have shown that the (simple) fences are generalized cliques of the *independence system* of acyclic arc sets of a complete digraph, and that the Möbius ladders without node and arc repetitions are generalized cycles of this IS. In both cases, their arguments prove faceteness of the associated inequalities for P_{ASP}, but they do, per se, not lead to separation algorithms.

Müller & Schulz (1996) [23] and Schulz (1996) [31] give similar results in the context of *transitive packing*. They also show that general fences are generalized cliques as well. A slight extension of their cutting plane constructions and inequality classes, with appropriate modifications to allow for arc repetitions, would classify our cliques of forkings as "extended generalized cliques" and our cycles of diwalks as "extended weak generalized $(k, 2)$-cycles".

Müller (1996) [21] has proved that fence separation is $\mathcal{NP}$-hard. Müller & Schulz (1995) [22] and Schulz (1996) [31], extending results of Müller (1996) [21], give a polynomial algorithm to separate a superclass (of so-called odd closed walk inequalities) of the Möbius ladder inequalities of Reinelt (1985) [30] for the linear ordering polytope.

Caprara & Fischetti (1996) [5] exhibit the Möbius ladder inequalities of

Reinelt (1985) [30] for the linear ordering polytope as $\{0, \frac{1}{2}\}$ *Chvátal-Gomory cuts* from an LU weakening of the system (LOP) (i)–(iv), which is of polynomial size. This already establishes polynomial separability of this class.

To a limited extent, similar results hold for the separation of Möbius ladder inequalities for the acyclic subdigraph polytope. Caprara & Fischetti (1996) [5] separate a class of inequalities similar to Möbius ladder inequalities (their cut (10)) that consists of $\{0, \frac{1}{2}\}$ Chvátal-Gomory cuts from an LU weakening of the system (ASP) (ii)–(iv), where (ASP) (ii) is restricted to a polynomial number of dicycle inequalities, e.g., those from dicycles of some arbitrary, but bounded length. In case of arc repetition, Caprara & Fischetti (1996) [5] cut (10) is stronger than a corresponding cycle of diwalks inequality whenever the structure of the latter gives also rise to the former. Möbius ladders with dicycles of arbitrary length, however, can not be separated in this way.

4 The Clique Partitioning Problem

In this section, we investigate set packing relaxations of combinatorial optimization problems in connection with cuts: The clique partitioning, the k-multicut, and the max cut problem. We will see that the 2-chorded cycle inequalities for the clique partitioning polytope can be seen as cycles of "lower" triangle inequalities. As a reference to the clique partitioning problem, we suggest Grötschel & Wakabayashi (1990) [13], see also Wakabayashi (1986) [33], for the multicut and the max cut problem Deza & Laurent (1997) [10].

The three cut problems of this section come up on a complete graph $K_n = (V, E)$ on n nodes with integer weights $w : E \to \mathbb{Z}$ on its edges. The *clique partitioning problem* (CPP) is to find a partition of V into an arbitrary number k of cliques $V = C_1 \cup \ldots \cup C_k$ (where $\cup$ denotes a union of disjoint sets), such that the sum of the weights of the edges that run between different cliques is maximal. In other words, we are trying to find a *multicut* $\delta(C_1 : \cdots : C_k)$ of maximum weight, where the number k of (non-empty) members C_i of the *clique partition* $C_1 \cup \ldots \cup C_k$ is arbitrary. One obtains the *k-multicut problem* (k-MCP) from this formulation by restricting the number of cliques to be less than or equal to some given number k, and the *max cut problem* (MCP) by prescribing $k = 2$. Thus, any (max) cut is a k-multicut ($k \geq 2$), and any k-multicut comes from a clique partition. We remark that the CPP is commonly stated in an equivalent version to find a clique partition that minimizes the sum of the edge weights *inside* the cliques.

Denote by $\mathfrak{V}_\Delta$ the set of all *ordered* triples $(i, j, k) \in V^3$ of distinct nodes of

K_n, such that, in particular, (i, j, k) forms a triangle. With this notation, integer programs for the CPP and the k-MCP read

$$\begin{array}{lll} \max & \sum_{ij \in E} w_{ij} x_{ij} & \\ \text{(ii)} & x_{ij} - x_{jk} - x_{ik} \leq 0 & \forall\ (i,j,k) \in \mathfrak{V}_\Delta \\ \text{(iii)} & -x_{ij} \leq 0 & \forall\ ij \in E \\ \text{(iv)} & x_{ij} \leq 1 & \forall\ ij \in E \\ \text{(v)} & x_{ij} \in \mathbb{Z} & \forall\ ij \in E \end{array} \tag{CPP}$$

$$\begin{array}{lll} \max & \sum_{ij \in E} w_{ij} x_{ij} & \\ \text{(i)} & \sum_{ij \in E(W)} x_{ij} \leq |E(W)| - 1 & \forall\ W \subseteq V : |W| = k+1 \\ \text{(ii)} & x_{ij} - x_{jk} - x_{ik} \leq 0 & \forall\ (i,j,k) \in \mathfrak{V}_\Delta \\ \text{(iii)} & -x_{ij} \leq 0 & \forall\ ij \in E \\ \text{(iv)} & x_{ij} \leq 1 & \forall\ ij \in E \\ \text{(v)} & x_{ij} \in \mathbb{Z} & \forall\ ij \in E. \end{array} \tag{k-MCP}$$

Inequalities (CPP) and (k-MCP) (ii) are called "lower" triangle inequalities (their normal vectors are oriented "downward" such that the induced face is on the "downside" of the polytope). Setting k to 2, inequalities (k-MCP) (i) turn out to be the "upper" triangle inequalities $x_{ij} + x_{jk} + x_{ik} \leq 2$ for all $(i, j, k) \in \mathfrak{V}_\Delta$, and (2-MCP) is an integer programming formulation for the max cut problem. For $k = n$, on the other hand, (k-MCP) (i) becomes void and (n-MCP) coincides with (CPP). Hence, (CPP) is a relaxation of (k-MCP) which in turn is a relaxation of (MCP) and the associated polytopes P_{CPP}, $P_{k-\text{MCP}}$, and P_{MCP} satisfy

$$P_{\text{CPP}} \supseteq P_{k-\text{MCP}} \supseteq P_{\text{MCP}}.$$

In particular, any valid inequality for the clique partitioning polytope is also valid for the k-multicut and the max cut polytope. One such family are the *2-chorded cycle inequalities* of Grötschel & Wakabayashi (1990) [13].

A *2-chorded cycle* is an odd cycle C of K_n together with its set of 2-chords $\overline{C}$, see Figure 5. The associated inequality states that

$$\sum_{ij \in \overline{C}} x_{ij} - \sum_{ij \in C} x_{ij} \leq (|C| - 1)/2.$$

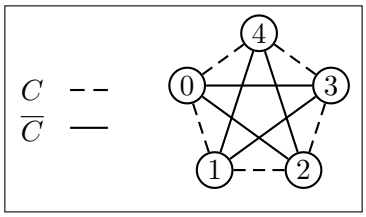


Figure 5: A 2-Chorded Cycle.

We show next that the 2-chorded cycle inequalities arise from odd cycle inequalities of a set packing relaxation of the clique partitioning (or k-multicut or max cut) problem. Our arguments establish the polynomial time separability of this class in an alternative way to earlier proofs of Müller (1996) [21] and Caprara & Fischetti (1996) [5].

The relaxation involves a "lower triangle" *conflict graph* $\mathfrak{G}_\Delta(K_n) = (\mathfrak{V}_\Delta, \mathfrak{E}_\Delta)$. $\mathfrak{V}_\Delta$ consists of all *ordered* triples $(i,j,k) \in V^3$ of distinct nodes of K_n, the edges $\mathfrak{E}$ of $\mathfrak{G}$ are of the form $(i,j,k)(l,i,j)$, $(i,j,k)(l,j,i)$, $(i,j,k)(l,i,k)$, and $(i,j,k)(l,k,i)$ (the meaning of this definition will become clear in a second).

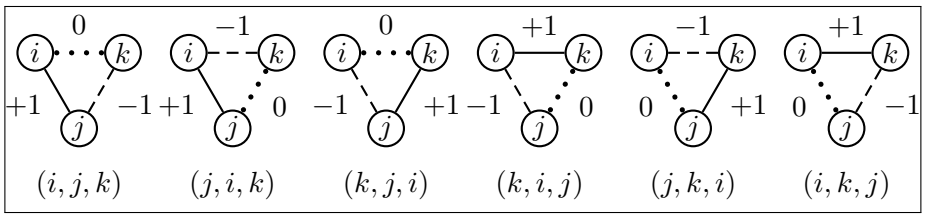


Figure 6: Labeling Lower Triangles.

To construct a set packing relaxation of the clique partitioning problem with this graph, we define a mapping $\pi : \mathbb{R}^E \to \mathbb{R}^{\mathfrak{V}_\Delta}$ as

$$\pi_{(i,j,k)}(x) := x_{ij} - x_{jk} \qquad \forall \text{ ordered triples } (i,j,k) \in \mathfrak{V}_\Delta.$$

$\pi_{(i,j,k)}(x)$ is integral if $x \in \mathbb{R}^E$ is integral. Moreover, for every multicut $x \in P_{\text{CPP}}$, the component $\pi_{(i,j,k)}(x)$ attains its maximum value of one if and only if the nodes j and k belong to the same clique ($x_{jk} = 0$), but node i does not ($x_{ij} = x_{ik} = 1$). The reader may think of the triples (i,j,k) as "edge-labelled triangles" as shown in Figure 6; then, it is easy to see that

$$\mathfrak{uv} \in \mathfrak{E}_\Delta \iff \pi_{\mathfrak{u}}(x) + \pi_{\mathfrak{v}}(x) \leq 1 \qquad \forall\, x \in P_{\text{CPP}} \cap \mathbb{Z}^E$$

and thus for all $x \in P_{\text{CPP}}$. In other words, $\mathfrak{E}_\Delta$ was defined in such a way that two triples are joined by an edge if and only if it is impossible that both attain their maximum value of one under π simultaneously. This argument shows that $P_I(\mathfrak{G}_\Delta)$ is a "lower triangle" set packing relaxation of P_{CPP}:

Lemma 4.1 (Set Packing Relaxation of the CPP).
$\pi(P_{\text{CPP}}) \subseteq P_I\big(\mathfrak{G}_\Delta(K_n)\big)$.

The construction is called a "lower triangle set packing relaxation", because one obtains the components $\pi_{(i,j,k)}(x) = x_{ij} - x_{jk} \leq 1$ of π from the lower triangle inequalities (CPP) (ii) by setting $x_{ik} = 1$:

$$x_{ij} - x_{jk} - x_{ik} \leq 0 \iff x_{ij} - x_{jk} \leq x_{ik}.$$

We are now ready to state our result that the 2-chorded cycle inequalities are expansions (see the definition on page 11) of odd cycle inequalities of $\breve{P}_{\text{SSP}}(\mathfrak{G}_\Delta)$.

Theorem 4.2 (2-Chorded Cycle Inequalities).
Let K_n be the complete graph on n nodes, P_{CPP} the corresponding clique partitioning polytope, $\mathfrak{G}_\Delta$ the lower triangle conflict graph, and $\breve{P}_{\text{SSP}}(\mathfrak{G}_\Delta)$ the lower triangle set packing relaxation of P_{CPP}.

Every 2-chorded cycle inequality for P_{CPP} is the expansion of an odd cycle inequality for $\breve{P}_{\text{SSP}}(\mathfrak{G}_\Delta)$.

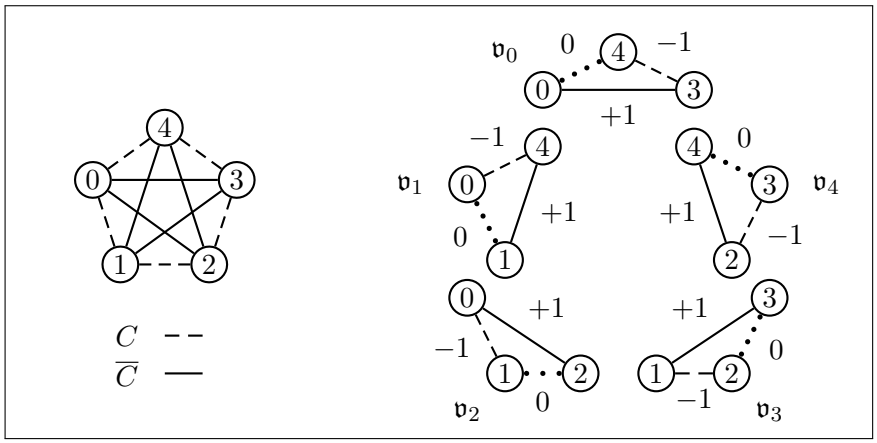


Figure 7: An Odd Cycle of Lower Triangle Inequalities.

Proof. Let $C \cup \overline{C}$ be a 2-chorded cycle in K_n with node set $\{0,\dots,2k\}$. By definition, $C = \{ij : i = 0,\dots,2k,\ j = i+1\}$ and $\overline{C} = \{ij : i = 0,\dots,2k,\ j = i+2\}$ (where indices are taken modulo $2k+1$).

Consider the $2k+1$ triples $\mathfrak{v}_i := (i, i-2, i-1)$, $i = 0,\dots,2k$ (indices modulo $2k+1$). One verifies that $\mathfrak{v}_i\mathfrak{v}_{i+1} \in \mathfrak{E}$ are in conflict and form the edge set of an odd cycle in $\mathfrak{G}_\Delta$, see Figure 7 for an example. The associated odd cycle inequality expands to the 2-chorded cycle inequality in question:

$$\sum_{i=0}^{2k} \pi_{(i,i-2,i-1)}(x) = \sum_{i=0}^{2k} x_{(i,i-2)} - x_{(i-2,i-1)} = \sum_{ij\in\overline{C}} x_{ij} - \sum_{ij\in C} x_{ij} \le (|C|-1)/2.$$

□

Calling the expansions of odd cycle inequalities for $\breve{P}_{\text{SSP}}(\mathfrak{G}_\Delta)$ *inequalities from odd cycles of lower triangle inequalities*, we obtain

Corollary 4.3 (Separation of Inequalities from Odd Cycles of Lower Triangle Inequalities).
Let K_n be the complete graph on n nodes and P_{CPP} the associated clique

partitioning polytope. Suppose $x \in \mathbb{Q}^E$ satisfies the constraints (CPP) (ii)–(iv). Then:

Inequalities from odd cycles of lower triangle inequalities violated by x can be separated in polynomial time.

Proof. The conflict graph $\mathfrak{G}_\Delta$ has $6 \times \binom{n}{3} = O(n^3)$ triple-nodes. Its size is polynomial. The sum of the node weights on an edge never exceeds one. Applying the algorithm of Grötschel, Lovász & Schrijver (1988) [16], Lemma 9.1.11, we can find a most violated odd cycle inequality in $\mathfrak{G}_\Delta$ in polynomial time. □

Corollary 4.4 (Separation of 2-Chorded Cycle Inequalities).
A superclass of the 2-chorded cycle inequalities can be separated in polynomial time."

Note that the conflicts between two successive triples $\mathfrak{v}_i = (i, i-2, i-1)$ and $\mathfrak{v}_{i+1} = (i+1, i-1, i)$ in a 2-chorded cycle stem from the common edge connecting nodes i and $i-1$, that has a coefficient of -1 in $\pi_{\mathfrak{v}_{i+1}}$ and 0 in $\pi_{\mathfrak{v}_i}$. But conflicts arise also from common edges with $+1$ and -1 coefficients. Thus, besides possible node/edge repetitions and the like, odd cycle of lower triangle inequalities give rise to inequalities that do not correspond to 2-chorded cycle inequalities.

Müller (1996) [21] obtained similar results in a transitive packing context. He showed that the 2-chorded cycles belong to a larger class of odd closed walk inequalities and gave a polynomial time separation algorithm.

Caprara & Fischetti (1996) [5] derived the 2-chorded cycle inequalities as $\{0, \frac{1}{2}\}$ Chvátal-Gomory cuts from an LU weakening of the polynomial sized system (CPP) (ii)–(iv), thereby proving their polynomial separability.

5 The Set Packing Problem

We have demonstrated in the examples of the preceding sections that certain combinatorial optimization problems have interesting set packing relaxations. Perhaps a bit surprising, we show now that the set packing problem *itself* also has interesting set packing relaxations! These considerations yield alternative derivation and separation techniques for several classes of wheel inequalities, including two classes introduced by Barahona & Mahjoub (1994) [3] and Cheng & Cunningham (1997) [6], as well as a new class of cycle of cycles inequalities. A survey on results for the set packing problem can be found in Grötschel, Lovász & Schrijver (1988) [16].

The examples of this sections are based on a "rank" set packing relaxation that we introduce now. Given a set packing problem (SSP) on a graph $G = (V, E)$, the associated *conflict graph* $\mathfrak{G} = (\mathfrak{V}, \mathfrak{E})$ of the relaxation has the set $\mathfrak{V} := \{G[H] : H \subseteq V\}$ of all node induced subgraphs of G as its nodes. In order to define the set of edges, we consider the mapping $\pi : \mathbb{R}^V \to \mathbb{R}^{\mathfrak{V}}$ defined as

$$\pi_{G[H]}(x) = \sum_{i \in G[H]} x_i - (\alpha(G[H]) - 1) \qquad \forall \text{ node induced subgraphs } G[H] \in \mathfrak{V},$$

where $\alpha(G[H])$ denotes the *rank*, i.e., the maximum cardinality of a stable set, of $G[H]$. We draw an edge between two subgraphs $G[H]$ and $G[W]$ if there is no stable set in G such that its restrictions to $G[H]$ and $G[W]$ are simultaneously stable sets of maximum cardinality in $G[H]$ and $G[W]$, i.e.,

$$G[H]G[W] \in \mathfrak{E} \iff \pi_{G[H]}(x) + \pi_{G[W]}(x) \leq 1 \qquad \forall\, x \in P_I(G) \cap \mathbb{Z}^V.$$

Well known arguments show that $\breve{P}_{\mathrm{SSP}}(\mathfrak{G})$ is a set packing relaxation of P_I in the exponential space $\mathbb{R}^{\mathfrak{V}}$:

Lemma 5.1 (Rank Set Packing Relaxation of the SSP).
$\pi(P_I) \subseteq \breve{P}_{\mathrm{SSP}}(\mathfrak{G})$.

5.1 Wheel Inequalities

One method to derive polynomial time separable expansions of inequalities from the rank relaxation is to consider subgraphs of $\mathfrak{G}$ of polynomial size. A natural idea is to restrict the set of nodes of $\mathfrak{G}$ to

$$\mathfrak{V}_k := \{G[H] : H \subseteq V : |H| \leq k\},$$

the node induced subgraphs $G[H]$ of G with bounded numbers of nodes $|H| \leq k$ for some arbitrary, but fixed bound k. The smallest interesting case is $k = 2$, where $G[H]$ ($|H| \leq 2$) is either empty, a singleton, an edge, or a coedge (complement of an edge). The odd cycle inequalities that one obtains from this restricted relaxation $\breve{P}_{\mathrm{SSP}}(\mathfrak{G}[\mathfrak{V}_2])$ contain, among other classes, the *odd wheel inequalities* of the set packing polytope.

A $2k+1$-*wheel* is an odd cycle C of $2k+1$ nodes $\{0 \ldots, 2k\}$, say, plus an additional node $2k+1$ that is connected to all nodes of the cycle C. C is the *rim* of the wheel, node $2k+1$ is the *hub*, and the edges connecting the node $2k+1$ and i, $i = 0, \ldots, 2k$, are called *spokes*. For such a configuration, the following inequality holds:

$$kx_{2k+1} + \sum_{i=0}^{2k} x_i \leq k.$$

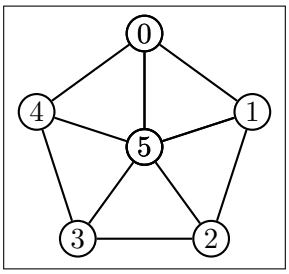


Figure 8: A 5-Wheel.

An odd wheel inequality can be obtained by a sequential lifting of the hub into the odd cycle inequality that corresponds to the rim. This can be used to construct a polynomial time separation algorithm for wheel inequalities which tries all possible hubs. An alternative derivation is

Theorem 5.2 (Odd Wheel Inequalities).
Let $G = (V, E)$ be a graph, P_I the associated set packing polytope, $\mathfrak{G}$ the rank conflict graph, and $\breve{P}_{\text{SSP}}(\mathfrak{G})$ the rank set packing relaxation of P_I.

Every odd wheel inequality for P_I is the expansion of an odd cycle inequality for $\breve{P}_{\text{SSP}}(\mathfrak{G}[\mathfrak{V}_2])$.

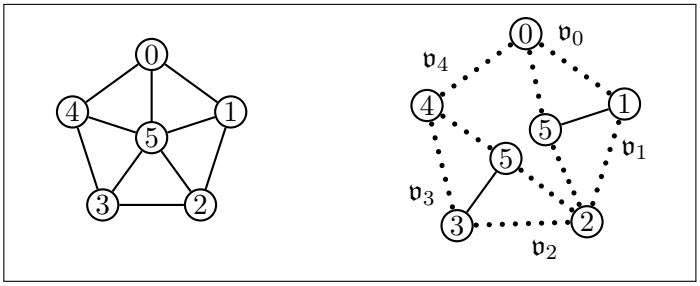


Figure 9: A Cycle of Nodes and Edges.

Proof. Consider a $2k+1$ wheel with rim $C = \{0, \ldots, 2k\}$ and hub node $2k+1$. The subgraphs $\mathfrak{v}_i := G[\{i, 2k+1\}]$, $i = 1, 3, \ldots, 2k-1$, induced by the spokes with odd rim nodes, and the subgraphs $\mathfrak{v}_i = G[\{i\}]$, $i = 0, 2, \ldots, 2k$, induced by the even rim nodes, form an odd cycle in $\mathfrak{G}$, see Figure 9 (the original wheel is on the left, the nodes of the conflict graph are right, the dotted edges indicate conflicts). Expanding the associated odd cycle inequality yields the wheel inequality:

$$\sum_{i=0}^{2k} \pi_{\mathfrak{v}_i(x)} = \sum_{i=1,3,\ldots,2k-1} (x_i + x_{2k+1}) + \sum_{i=0,2,\ldots,2k} x_i = kx_{2k+1} + \sum_{i=0}^{2k} x_i \leq k.$$

□

Proposition 5.3 (Separation of Inequalities from Odd Cycles of Nodes, Edges & Coedges).
Let $G = (V, E)$ be a graph and P_I the associated set packing polytope. Suppose $x \in \mathbb{Q}^V$ satisfies all edge constraints $x_i + x_j \leq 1$, $ij \in E$, and the bounds $0 \leq x_i \leq 1$, $i \in V$. Then:

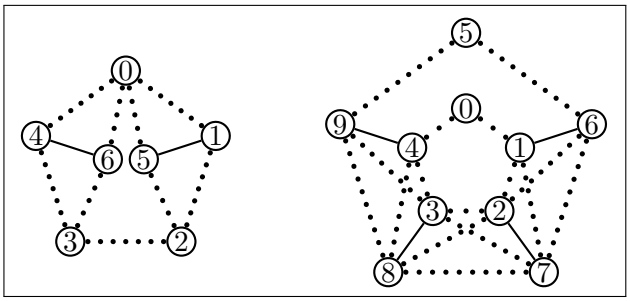
Inequalities from odd cycles of nodes, edges, and coedges violated by x can be separated in polynomial time.

We show now two *examples* of cycles of nodes, edges, and coedges that give rise to *facetial* inequalities that do not correspond to odd wheels. The cycle on the left side of Figure 10 consists of the nodes 0, 2, and 3 and the edges $(1, 5)$ and $(4, 6)$, the one on the right of the edges $(1, 6)$, $(2, 7)$, $(3, 8)$, and $(4, 9)$ and the coedge $(0, 5)$. The associated inequalities are

$$x_0 + (x_5 + x_1) + x_2 + x_3 + (x_6 + x_4) \leq 2 \Longleftrightarrow \sum_{i=0}^{6} x_i \leq 2$$

$$(x_5 + x_0 - 1) + (x_6 + x_1) + (x_7 + x_2) + (x_8 + x_3) + (x_9 + x_4) \leq 2 \Longleftrightarrow \sum_{i=0}^{9} x_i \leq 3.$$

$\sum_{i=0}^{6} x_i \leq 2$



$\sum_{i=0}^{9} x_i \leq 3$

Figure 10: Two Generalizations of Odd Wheel Inequalities.

Another generalization of odd wheel inequalities was given by Barahona & Mahjoub (1994) [3] and Cheng & Cunningham (1997) [6]. They introduce two classes of inequalities that have *subdivisions* of odd wheels as support graphs, where each face cycle must be odd, see Figure 11. Formally, a generalized $2k + 1$-wheel consists of an odd number $2k + 1$ of *spoke(path)s* S_i, $i = 0, \ldots, 2k$, that all have one common endnode, the *hub* h. The opposite endnodes of any two successive spokes S_i and S_{i+1} (indices taken modulo $2k + 1$) are joined by a *rim path* R_i, $i = 0, \ldots, 2k$, such that the *face cycle* formed by S_i, R_i and S_{i+1} is odd. Following for the remainder of this subsection the terminology of Cheng & Cunningham (1997) [6], a spoke is called *even* and *odd* if it has an even and odd number of *edges* (not of nodes!), respectively. (We temporarily override here the node oriented parity definition of the introduction for notational consistency with the literature.)

Let $\mathcal{E}$ and $\mathcal{O}$ be the endnodes of the even and odd spokes of an odd wheel W of this kind with some number $2k+1$ of faces, and let h be the hub. A *wheel inequality of type I* states that

$$kx_h + \sum_{i \in W \setminus \{h\}} x_i + \sum_{i \in \mathcal{E}} x_i \leq \frac{|W| + |\mathcal{E}|}{2} - 1. \tag{3}$$

A second variant of *wheel inequalities* (*of type II,* associated with the same wheel) states that

$$(k+1)x_h + \sum_{i \in W \setminus \{h\}} x_i + \sum_{i \in \mathcal{O}} x_i \leq \frac{|W| + |\mathcal{O}| - 1}{2}. \tag{4}$$

We remark that these wheels do in general not arise from cycles of subgraphs of bounded size because they contain potentially very long paths.

Theorem 5.4 (Odd Wheel Inequalities).
Let $G = (V, E)$ be a graph, P_I the corresponding set packing polytope, $\mathfrak{G}$ the rank conflict graph, and $\breve{P}_{\mathrm{SSP}}(\mathfrak{G})$ the rank set packing relaxation of P_I.

Every odd wheel inequality of type I and II for P_I is the expansion of an odd cycle inequality for $\breve{P}_{\mathrm{SSP}}(\mathfrak{G})$.

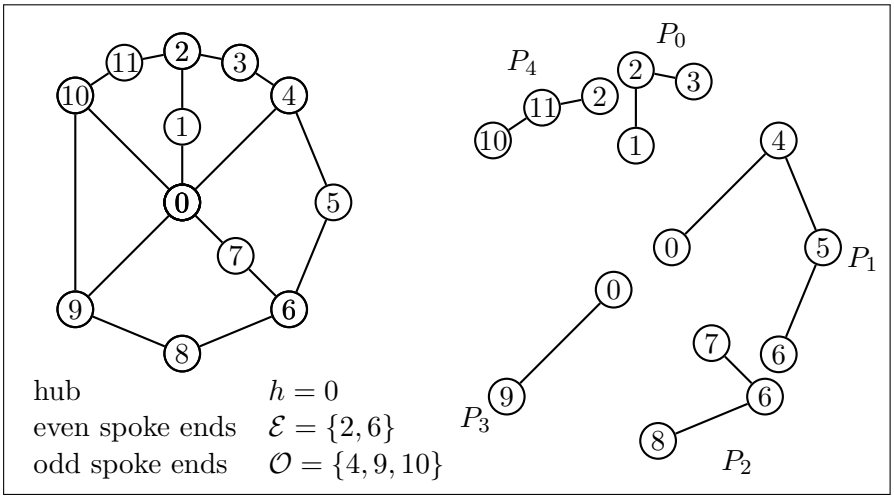


Figure 11: A 5-Wheel of Type I and a 5-Cycle of Paths.

Proof. (i) Wheel inequalities of type I.

The idea of the proof is to obtain the wheel inequality (3) of type I as a cycle of paths, namely, the paths

$$P_i := S_i \cup \begin{Bmatrix} R_i, & \text{if } S_{i+1} \text{ is even} \\ R_i \setminus S_{i+1}, & \text{if } S_{i+1} \text{ is odd} \end{Bmatrix} \setminus \begin{Bmatrix} \emptyset, & \text{if } i \text{ is odd} \\ \{h\}, & \text{if } i \text{ is even} \end{Bmatrix}, \quad i = 0, \ldots, 2k,$$

see Figure 11. By definition, a path P_i consists of the spoke S_i plus minus the hub depending on i, and the *full* rim path R_i if the end node of the next spoke (in clockwise order) is even, or the rim path R_i *without* the end of the next spoke in case this spoke is odd. In this way, the even spoke ends, having a coefficient of two in the wheel inequality, appear in two paths, the odd spoke ends in one. (Recall that *in this context* a spoke was odd/even if it contained an odd/even number of *edges*.) It is not hard to see that any two successive paths P_i and P_{i+1} are in pairwise conflict: The subpaths $P_i \setminus \{h\}$ with the hub removed are all odd and in pairwise conflict, and, likewise, the hub is in conflict with any of these subpaths. The odd cycle inequality corresponding to the paths P_i expands into the odd wheel inequality (3):

$$\begin{aligned}
&\sum_{i=0}^{2k} \pi_{P_i}(x) \leq k \\
\iff \quad &\sum_{i=0}^{k} \left(\sum_{j \in P_{2i}} x_j - (|P_{2i}| - 1)/2 \right) + \sum_{i=0}^{k-1} \left(\sum_{j \in P_{2i+1}} x_j - (|P_{2i+1}| - 2)/2 \right) \leq k \\
\iff \quad &k x_h + \sum_{j \in W \setminus \{h\}} x_j + \sum_{j \in \mathcal{E}} x_j - \frac{|W| - 1 + k + |\mathcal{E}| - (k+1) - 2k}{2} \leq k \\
\iff \quad &k x_h + \sum_{j \in W \setminus \{h\}} x_j + \sum_{j \in \mathcal{E}} x_j \leq \frac{|W| + |\mathcal{E}| - 2k - 2}{2} + k = \frac{|W| + |\mathcal{E}|}{2} - 1.
\end{aligned}$$

(Here, $|P_i|$ denotes the number of nodes in path P_i).

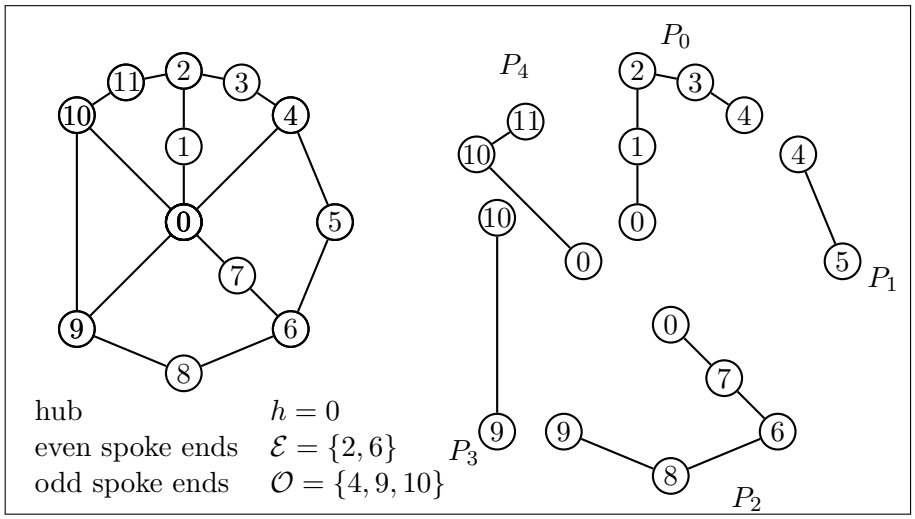


Figure 12: A 5-Wheel of Type II and a 5-Cycle of Paths.

(ii) Wheel inequalities of type II.

The wheel inequalities (4) of type II can be derived in much the same way as their relatives of type I. For the sake of completeness, we record the path

decomposition

$$P_i := S_i \cup \begin{Bmatrix} R_i, & \text{if } S_{i+1} \text{ is odd} \\ R_i \setminus S_{i+1}, & \text{if } S_{i+1} \text{ is even} \end{Bmatrix} \setminus \begin{Bmatrix} \emptyset, & \text{if } i \text{ is even} \\ \{h\}, & \text{if } i \text{ is odd} \end{Bmatrix}, \quad i = 0, \ldots, 2k.$$

One can verify that, again, any two successive paths are in conflict. A final calculation to expand the resulting odd cycle inequality yields the wheel inequality (4) of type II:

$$\begin{aligned} & \sum_{i=0}^{2k} \pi_{P_i}(x) \leq k \\ \iff & \sum_{i=0}^{k} \left(\sum_{j \in P_{2i}} x_j - (|P_{2i}| - 1)/2 \right) + \sum_{i=0}^{k-1} \left(\sum_{j \in P_{2i+1}} x_j - (|P_{2i+1}| - 2)/2 \right) \leq k \\ \iff & (k+1)x_h + \sum_{j \in W \setminus \{h\}} x_j + \sum_{j \in \mathcal{O}} x_j - \frac{|W| - 1 + (k+1) + |\mathcal{O}| - (k+1) - 2k}{2} \leq k \\ \iff & (k+1)x_h + \sum_{j \in W \setminus \{h\}} x_j + \sum_{j \in \mathcal{O}} x_j \leq \frac{|W| + |\mathcal{O}| - 2k - 1}{2} + k = \frac{|W| + |\mathcal{O}| - 1}{2}. \end{aligned}$$

□

One can also derive polynomial time separation algorithms of much the same flavour as for the odd cycle of diwalk inequalities; Cheng & Cunningham (1997) [6] give such procedures.

5.2 A New Family of Facets for the Stable Set Polytope

The rank relaxation of the set packing problem offers ample possibilities to define new classes of polynomially separable inequalities for the set packing problem. We discuss, as one such example, a class of *cycle of cycles* inequalities.

The way to construct a cycle of cycles inequality is to link an odd number $2k + 1$ of odd cycles $C_0, \ldots, C_{2k}$ to a circular structure, such that any two successive cycles are in pairwise conflict, i.e., $\pi_{C_i}(x) + \pi_{C_{i+1}}(x) \leq 1$ (indices taken modulo $2k + 1$).

One way to do this is to select from each cycle C_i three successive nodes $L_i \subseteq C_i$ that will serve as a part of the inter-cycle links yet to be formed. The *link* L_i has the property that $\pi_{C_i}(x) = 1$ implies that at least one of the nodes in L_i is contained in the stable set $\text{supp}(x)$, i.e.,

$$\pi_{C_i}(x) = 1 \Longrightarrow \sum_{j \in L_i} x_j \geq 1.$$

If we make sure that any two successive links L_i and L_{i+1} are joined by the edge set of the complete bipartite graph $K_{3,3}$, then the inequality

$$\sum_{j \in L_i} x_j + \sum_{j \in L_{i+1}} x_j \leq 1$$

holds for all incidence vectors x of stable sets in G. But then, the corresponding two successive cycles C_i and C_{i+1} are in conflict, i.e., $\pi_{C_i}(x) + \pi_{C_{i+1}}(x) \leq 1$, and the cycles C_i form an odd cycle in $\mathfrak{G}$, see Figure 13.

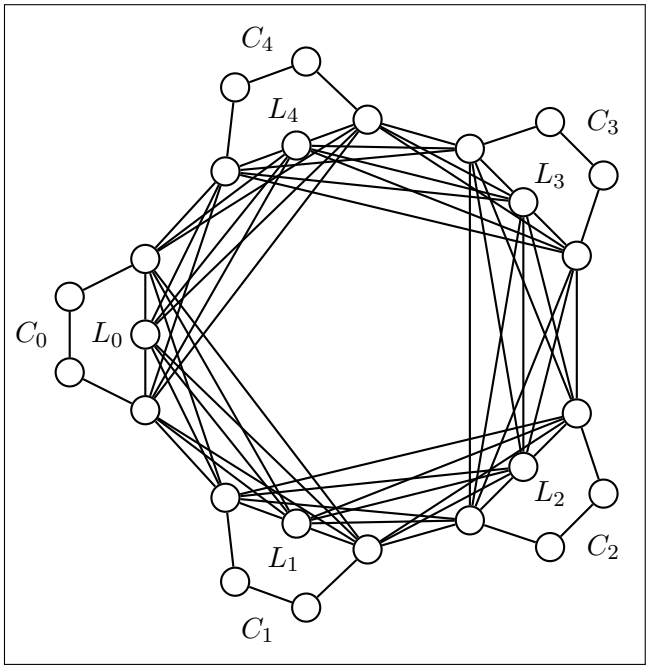


Figure 13: A 5-Cycle of 5-Cycles.

Theorem 5.5 (Cycle of Cycles Inequality).
Let $G = (V, E)$ be a graph and P_I be the corresponding set packing polytope. Let C_i, $i = 0, \ldots, 2k$, be an odd cycle and $L_i \subseteq C_i$, $i = 0, \ldots, 2k$, a set of three successive nodes in C_i. Assume further that L_i and L_{i+1}, $i = 0, \ldots, 2k$, are joined by a complete $K_{3,3}$.

Then the following cycle of cycles inequality is valid for P_I:

$$\sum_{i=0}^{2k} \sum_{j \in C_i} x_j \leq \sum_{i=0}^{2k} (|C_i| - 1)/2 - (k+1).$$

Proof.

$$\begin{aligned}
& \sum_{i=0}^{2k} \pi_{C_i}(x) \leq k \\
\iff & \sum_{i=0}^{2k} \left(\sum_{j \in C_i} x_j - \big((|C_i| - 1)/2 - 1\big) \right) \leq k \\
\iff & \sum_{i=0}^{2k} \sum_{j \in C_i} x_j \leq \sum_{i=0}^{2k} \big((|C_i| - 1)/2 - 1\big) + k = \sum_{i=0}^{2k} (|C_i| - 1)/2 - (k+1).
\end{aligned}$$

□

A cycle of cycles inequality will in general not be facet inducing, for example, if one of the cycles has a chord that does not join two nodes of its link. But one can come up with conditions that ensure this property. The most simple case is where the cycles C_i are holes, all node disjoint, and the only edges that run between different holes belong to the links, i.e., we have a "hole of holes".

Theorem 5.6 (Facet Inducing Cycle of Cycles Inequalities).
If every cycle in a cycle of cycles inequality is a hole, all node disjoint, and the only edges that run between different holes emerge from the links, then the cycle of cycles inequality is facet inducing.

Proof. The proof is based on a sufficiency criterion for the faceteness of rank inequalities by Chvátal (1975) [9]. It is based on the notion of *critical edges* in a graph $G = (V, E)$: An edge $ij \in E$ is critical if its removal increases G's rank, i.e., if $\alpha(G - ij) = \alpha(G) + 1$. The criterion states that if the graph $G^* := (V, E^*)$ is connected, where E^* is the set of critical edges of G, the rank inequality $\sum_{i \in V} x_i \leq \alpha(G)$ is facet defining for $P_I(G)$. It is easy to see that this condition holds in this case. □

Theorem 5.7 (Separation of Cycle of Cycles Inequalities).
Let $G = (V, E)$ be a graph and P_I the associated set packing polytope. Suppose $x \in \mathbb{Q}^V$ satisfies all bound, edge, and odd cycle constraints. Then:

Cycle of cycles inequalities violated by x can be separated in polynomial time.

Proof. The number of potential links L_i is polynomial of order $O(|V|^3)$. We set up a *link graph*, that has the links as its nodes; this device will, in a second, turn out to be a subgraph of $\mathfrak{G}$. Two links are connected by an edge if and only if they are joined by a $K_{3,3}$. To assign weights to the links, we calculate for each link L_i the shortest even path P_i in G that connects the two endpoints of the link (see, e.g., Barahona & Mahjoub (1985) [2] how to

find even paths); here, shortest means shortest with respect to the length function

$$(1 - x_i - x_j)/2 \qquad \forall \text{ edges } ij \in E.$$

$L_i \cup P_i$ forms an odd cycle C_i through L_i. We set the weight of link L_i to the value $\pi_{C_i}(x)$, obtain the link graph as a subgraph of $\mathfrak{G}[\{C_i\}]$ (some edges that correspond to "non-link conflicts" are possibly missing), and detect a violated odd cycle inequality in the link graph if and only if a violated cycle of cycles inequality in G exists. □

Acknowledgment. The work of the second author was partially supported by the Gerhard Hess Forschungsförderpreis of the Deutsche Forschungsgemeinschaft, the Kultusministerium Sachsen Anhalt, and an EU Donet Project. We thank Adam Letchford, Akiyoshi Shioura, and four anonymous referees for helpful suggestions and comments.

References

[1] E. Balas & W. Pulleyblank. *The perfectly matchable subgraph polytope of an arbitrary graph.* Combinatorica **9**:321–327, 1989. Cited on page 2.

[2] F. Barahona & A. Mahjoub. *On the cut polytope.* Math. Programming **36**:157–173, 1985. Cited on page 28.

[3] F. Barahona & A. Mahjoub. *Compositions of graphs and polyhedra II: Stable sets.* SIAM J. Discrete Math. **7**:359–371, 1994. Cited on pages 2, 20, 23.

[4] C. Berge. *Balanced matrices.* Math. Programming **2**:19–31, 1971. Cited on page 1.

[5] A. Caprara & M. Fischetti. $\{0, \frac{1}{2}\}$*-Chvátal-Gomory cuts.* Math. Programming **74**(3):221–235, 1996. Cited on pages 4, 7, 15, 16, 18, 20.

[6] E. Cheng & W. H. Cunningham. *Wheel inequalities for stable set polytopes.* Math. Programming **77**(3):389–421, 1997. Cited on pages 2, 20, 23, 26.

[7] S. Chopra & M. Rao. *The steiner tree problem I: Formulations, compositions, and extensions of facets.* Math. Programming **64**:209–229, 1994. Cited on page 2.

[8] S. Chopra & M. Rao. *The steiner tree problem II: Properties and classes of facets.* Math. Programming **64**:231–246, 1994. Cited on page 2.

[9] V. Chvátal. *On certain polytopes associated with graphs.* J. Combin. Theory Ser. B **18**:138–154, 1975. Cited on page 28.

[10] M. Deza & M. Laurent. *Geometry of Cuts and Metrics*. Springer Verlag, Berlin, 1997. Cited on pages 2, 16.

[11] R. Euler, M. Jünger & G. Reinelt. *Generalizations of cliques, odd cycles and anticycles and their relation to independence system polyhedra*. Math. Oper. Res. **12**(3):451–462, August 1987. Cited on pages 4, 5, 15.

[12] D. Fulkerson. *Blocking and Anti-Blocking Pairs of Polyhedra*. Math. Programming **1**:168–194, 1971. Cited on page 1.

[13] M. Grötschel & Y. Wakabayashi. *Facets of the clique partitioning polytope*. Math. Programming **47**(3):367–387, 1990. Cited on pages 2, 16, 17.

[14] M. Grötschel, M. Jünger & G. Reinelt. *On the acyclic subgraph polytope*. Math. Programming **33**:28–42, 1985. Cited on pages 2, 8, 9, 12, 15.

[15] M. Grötschel, M. Jünger & G. Reinelt. *Facets of the linear ordering polytope*. Math. Programming **33**:43–60, 1985. Cited on pages 2, 8.

[16] M. Grötschel, L. Lovász & A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, vol. 2 of *Algorithms and Combinatorics*. Springer Verlag, Berlin, 1988. ISBN 3-540-13624-X, 0-387-13624-X (U.S.). Cited on pages 2, 4, 14, 15, 20.

[17] M. Jünger. *Polyhedral Combinatorics and the Acyclic Subdigraph Problem*. Heldermann Verlag, Berlin, 1985. Cited on page 8.

[18] M. Laurent. *A generalization of antiwebs to independence systems and their canonical facets*. Math. Programming **45**:97–108, 1989. Cited on pages 4, 5.

[19] L. Lovász. *Normal hypergraphs and the perfect graph conjecture*. Discrete Math. **2**:253–267, 1971. Cited on page 1.

[20] L. Lovász & A. Schrijver. *Cones of matrices and set-functions and 0-1 optimization*. SIAM J. Optim. **1**:166–190, 1991. Cited on page 2.

[21] R. Müller. *On the partial order polytope of a digraph*. Math. Programming **73**(1):31–49, 1996. Cited on pages 4, 5, 6, 15, 18, 20.

[22] R. Müller & A. S. Schulz. *The interval order polytope of a digraph*. In E. Balas & J. Clausen, (Eds.), *Integer Programming and Combinatorial Optimiziation*, Proc. 4th Int. IPCO Conf., pp. 50–64, 1995. Cited on pages 4, 6, 15.

[23] R. Müller & A. S. Schulz. *Transitive packing*. In W. H. Cunningham, S. T. McCormick & M. Queyranne, (Eds.), *Integer Programming and Combinatorial Optimization*, Proc. 5th Int. IPCO Conf., pp. 430–444, 1996. Cited on pages 4, 6, 15.

[24] G. L. Nemhauser & L. E. Trotter, Jr. *Properties of vertex packing*

and independence system polyhedra. Math. Programming **6**:48–61, 1973. Cited on pages 4, 5.

[25] P. Nobili & A. Sassano. *Facets and lifting procedures for the set covering polytope*. Math. Programming **45**:111–137, 1989. Cited on pages 4, 5.

[26] M. W. Padberg. *On the facial structure of set packing polyhedra*. Math. Programming **5**:199–215, 1973. Cited on pages 1, 2, 4.

[27] M. W. Padberg. *A note on zero-one programming*. Oper. Res. **23**(4): 833–837, 1975. Cited on page 5.

[28] M. W. Padberg & T.-Y. Sung. *An analytical comparison of different formulations of the travelling salesman problem*. Math. Programming **52**(2):315–357, 1991. Cited on pages 2, 5.

[29] W. Pulleyblank & F. Shepherd. *Formulations for the stable set polytope of a claw-free graph*. In G. Rinaldi & L. Wolsey, (Eds.), *Integer programming and combinatorial optimization 3*, Proc. of the 3rd Int. IPCO Conf., pp. 267–279, 1993. Cited on page 2.

[30] G. Reinelt. *The linear ordering problem: Algorithms and applications*, vol. 8 of *Res. and Exposition in Math.* Heldermann Verlag, Berlin, 1985. Cited on pages 8, 15, 16.

[31] A. S. Schulz. *Polytopes and Scheduling*. PhD thesis, Technische Universität Berlin, 1996. Cited on pages 4, 6, 15.

[32] Y. Sekiguchi. *A note on node packing polytopes on hypergraphs*. Oper. Res. Lett. **2**(5):243–247, 1983. Cited on pages 4, 5.

[33] Y. Wakabayashi. *Aggregation of Binary Relations: Algorithmic and Polyhedral Investigations*. PhD thesis, Universität Augsburg, 1986. Cited on page 16.

Paper II

Combinatorial Packing Problems

R. Borndörfer.
Combinatorial packing problems.
In M. Grötschel, (Ed.), *The Sharpest Cut – The Impact of Manfred Padberg and His Work*, pp. 19–32. SIAM, Philadelphia, 2004.

Abstract. This article investigates a certain class of combinatorial packing problems and some polyhedral relations between such problems and the set packing problem.

Mathematics Subject Classification (MSC 2000). 90C27, 90C57

Keywords. Packing problems, polyhedral combinatorics

1 Introduction

Packing constraints are one of the most common problem characteristics in combinatorial optimization. They come up in problems of vehicle and crew scheduling, VLSI and network design, and frequency assignment, see Frank (1990) [16]; Padberg (1979) [39] for surveys. The pure form is the *set packing* or *stable set problem* (SPP) in a graph $G = (V, E)$ with node weights w; it asks for a maximum weight set of mutually non-adjacent nodes. This problem has been studied extensively, and deep structural and algorithmic results have been achieved in areas such as anti-blocking theory, the theory of perfect graphs, perfect and balanced matrix theory, and semidefinite programming, see Balas & Padberg (1976) [6]; Borndörfer (1998) [8]; Grötschel, Lovász & Schrijver (1988) [20]; Nemhauser & Wolsey (1988) [34] for surveys. There is, in particular, a substantial structural and algorithmic knowledge

of the set packing polytope, with many classes of strong and polynomial time separable inequalities such as odd hole, odd antihole, orthonormal representation constraints and other classes, see Grötschel, Lovász & Schrijver (1988) [20]; Nemhauser & Trotter (1973) [33]; Padberg (1973) [36]; Padberg (1977) [38]; Trotter (1975) [45].

Several research directions try to translate some of these results to broader settings. A first line investigates generalizations of set packing such as node packing in hypergraphs, see Sekiguchi (1983) [42], independence systems, see Euler, Jünger & Reinelt (1987) [13]; Laurent (1989) [26]; Nemhauser & Trotter (1973) [33]; Padberg (1975) [37], transitive packing, see Müller (1996) [30]; Müller & Schulz (1995) [31]; Müller & Schulz (1996) [32]; Schulz (1996) [41], and mixed integer packing, see Atamturk, Nemhauser & Savelsbergh (1998) [2]; Atamtrk, Nemhauser & Savelsbergh (2000) [3]. This work aims at a unified polyhedral theory. A second direction is the theory of matrix cuts, see Lovász & Schrijver (1991) [27], which generalizes the semidefinite separation machinery that had been developed for the solution of the stable set problem in perfect graphs, see Grötschel, Lovász & Schrijver (1988) [20], to arbitrary 0/1 programs. A third approach is the construction of discrete set packing relaxations, see Borndörfer & Weismantel (2000) [9]; Borndörfer & Weismantel (2001) [10] and also Padberg & Sung (1991) [40]. This technique allows to transfer set packing inequalities and separation algorithms to other combinatorial problems.

Our aim in this paper is to continue in this general direction. We consider a class of combinatorial optimization problems of packing type where a Dantzig-Wolfe decomposition gives rise to a canonical, yet exponential, set packing formulation, namely, the formulation that one would use in a column generation approach. This alternative formulation allows, at least in principle, to understand combinatorial packing problems completely in terms of set packing theory. We show that such Dantzig-Wolfe set packing formulations of combinatorial packing problems have structural properties that relate them to the original formulation and make them interesting sources of cutting planes.

The article consists of two parts. In Section 2 we introduce the concept of combinatorial packing. We give two examples of such problems, namely, on packings of two stable sets in bipartite graphs and independent sets in any number of matroids, which are naturally integral. Dantzig-Wolfe set packing formulations of combinatorial packing problems are discussed in Section 3. It is shown that such formulations give rise to cutting planes and that the intersection graphs associated with Dantzig-Wolfe formulations of combinatorial 2-packing problems are perfect.

2 Combinatorial Packing

We introduce in this section the notion of combinatorial packing. This concept subsumes a variety of combinatorial optimization problems, among them the Steiner tree packing problem, the multicommodity flow problem with unit capacities, the multiple knapsack problem, and the coloring problem. It will turn out that for some problems of this type, namely, the 2-coloring problem in bipartite graphs and the matroid packing problem, the integrality of the individual subproblems carries over to the packing composition.

Consider a family of some number k of combinatorial optimization problems

$$(\mathrm{IP}^i) \quad \max\ {c^i}^{\mathrm{T}} x^i, \quad M^i x^i \le b^i,\ 0 \le x^i \le \mathbf{1},\ x^i \in \mathbb{Z}^E, \quad i = 1, \dots, k \tag{1}$$

on the same ground set E. These are the *individual problems*. Associated with each of them is an *individual polytope* $P^I_{\mathrm{IP}^i} = \mathrm{conv}\{x^i \in \{0,1\}^E \mid M^i x^i \le b^i\}$ and its fractional relaxation $P_{\mathrm{IP}^i} = \{0 \le x^i \le \mathbf{1} \mid M^i x^i \le b^i\}$. An individual problem with the property $P^I_{\mathrm{IP}^i} = P_{\mathrm{IP}^i}$ is called *integral*.

A *packing* is a collection of individual solutions $x^1, \dots, x^k$ of $\mathrm{IP}^1, \dots, \mathrm{IP}^k$, respectively, such that each element of the ground set is contained in at most one solution. The problem to find a maximum weight packing is the *combinatorial packing problem* (CPP) associated with the individual problems IP^i, $i = 1, \dots, k$. A CPP with k individual problems is a *(combinatorial) k-packing problem*. The IP formulation of a CPP reads

$$\begin{array}{lll} (\mathrm{CPP}) & \max \sum_{i=1}^k {c^i}^{\mathrm{T}} x^i & \\ \text{(i)} & M^i x^i \le b^i, & i = 1, \dots, k \\ \text{(ii)} & x^i \ge 0, & i = 1, \dots, k \\ \text{(iii)} & \sum_{i=1}^k x^i \le \mathbf{1} & \\ \text{(iv)} & x^i \in \mathbb{Z}^E, & i = 1, \dots, k. \end{array} \tag{2}$$

We call CPP (iii) the *packing constraints*. It will be convenient to use the notation $x^{\mathrm{T}} = ({x^1}^{\mathrm{T}}, \dots, {x^k}^{\mathrm{T}})$ and $c^{\mathrm{T}} = ({c^1}^{\mathrm{T}}, \dots, {c^k}^{\mathrm{T}})$. Likewise, we shall view the ground set of a combinatorial k-packing problem as a disjoint union $\bigcup E^i = E^1 \cup \dots \cup E^k$ of copies of the ground sets of the individual problems, where E^i is the copy of the ground set of problem IP^i. Associated with the CPP are finally the *combinatorial packing polytope* and its fractional relaxation

$$\begin{aligned} P_{\mathrm{CPP}} &= \mathrm{conv}\{x \in \{0,1\}^{\bigcup E^i} \mid \textstyle\sum_{i=1}^k x^i \le \mathbf{1},\ M^i x^i \le b^i,\ i = 1, \dots, k\}. \\ P_{\mathrm{CPP}} &= \{x \in [0,1]^{\bigcup E^i} \mid \textstyle\sum_{i=1}^k x^i \le \mathbf{1},\ M^i x^i \le b^i,\ i = 1, \dots, k\}. \end{aligned} \tag{3}$$

A CPP is *integral* if $P_{\text{CPP}} = P_{\text{CPP}}$. If all individual problems as well as CPP itself are integral, we say that CPP is *naturally integral*.

2.1 Examples of Combinatorial Packing Problems

The Multicommodity Flow Problem with Unit Capacities involves a *supply digraph* $D_S = (V, A_S)$ and a *demand digraph* $D_D = (V, A_D)$, both on the same node set V. We denote an arc from a node s to a node t in these digraphs by st. There are non-negative weights $w \in \mathbb{Q}_+^{A_S}$ on the arcs A_S of the supply digraph. A *multiflow* is a collection of pairwise arc disjoint directed st-paths in D_S, one for each arc $st \in A_D$ of the demand digraph. The *multicommodity flow problem with unit capacities* (MCFP) asks for a multiflow of minimum weight, see Ahuja, Magnanti & Orlin (1989) [1]; Deza & Laurent (1997) [12]; Frank (1990) [16].

The MCFP is a combinatorial path packing problem. The individual problems are shortest path problems, one for each demand arc $st \in A_D$:

$$\begin{aligned} \min\ & w^{\mathrm{T}} x^{st} \\ & x^{st}(\delta^+(v)) - x^{st}(\delta^-(v)) = e_s - e_t, \qquad \forall v \in V \\ 0 \le\ & x^{st} \le \mathbf{1} \\ & x^{st} \in \mathbb{Z}^{A_S}. \end{aligned} \tag{4}$$

Combining the shortest path problems in a CPP adds the packing constraints $\sum_{st \in A_S} x^{st} \le \mathbf{1}$ that model the edge disjointness of the paths.

The Steiner Tree Packing Problem involves a graph $G = (V, E)$, some number k of sets of *terminal* nodes $T^1, \dots, T^k \subseteq V$, and non-negative edge weights $w^1, \dots, w^k \in \mathbb{Q}_+^E$. The *Steiner tree packing problem* (PST) is to find a collection of Steiner trees $S^1, \dots, S^k$ spanning the terminals $T^1, \dots, T^k$, respectively, such that no two Steiner trees have an edge in common, see Grötschel, Martin & Weismantel (1996) [21, 22, 23, 24]; Martin (1992) [29]. Note that terminal sets of two nodes will be joined by paths such that the PST subsumes the MCFP.

The PST is a combinatorial packing problem. The individual problems, one for each terminal set T^i, $i = 1, \dots, k$, are Steiner tree problems

$$\begin{aligned} \min\ & {w^i}^{\mathrm{T}} x^i \\ & x^i(\delta(W)) \ge 1, \qquad \forall W \subseteq V :\ W \cap T^i \neq \emptyset \neq (V \setminus W) \cap T^i \\ 0 \le\ & x^i \le \mathbf{1} \\ & x^i \in \mathbb{Z}^E. \end{aligned} \tag{5}$$

Combining the problems in a CPP forces the Steiner trees to be edge disjoint.

The Generalized Assignment Problem deals with a set of *jobs* J to be processed by a set of *machines* I with capacities α^i. There are resource demands a^i_j and profits w^i_j for the assignment of job j to machine i. The *generalized assignment problem* (GAP) is to find a maximum profit assignment of jobs to machines, see Gottlieb & Rao (1990) [18]; Martello & Toth (1990) [28]. The special case where the resource demands and availabilities do not depend on the machines, i.e., when $a^i = a^k$ and $\alpha^i = \alpha^k$ for all $i, k \in I$, is known as the *multiple knapsack problem* (MKP), see Ferreira (1994) [14]; Ferreira, Martin & Weismantel (1996) [15]; Martello & Toth (1990) [28].

The GAP models combinatorial packings of job-machine assignments. There is an individual knapsack problem for each of the machines $i \in I$

$$\max\ {w^i}^{\mathrm{T}} x^i, \quad {a^i}^{\mathrm{T}} x^i \leq \alpha^i,\ 0 \leq x^i \leq \mathbf{1},\ x^i \in \mathbb{Z}^J. \tag{6}$$

The packing constraints forbid assignments of jobs to more than one machine.

The k-Coloring Problem involves a graph $G = (V, E)$ with node weights $w \in \mathbb{Q}^V_+$ and some number $k \in \mathbb{N}$ of colors. The *k-coloring problem* (k-COL) asks for a collection of k mutually disjoint stable sets (color classes) of maximum weight, see Toft (1995) [44].

A combinatorial packing formulation of the k-coloring problem is based on k individual stable set problems

$$\max\ w^{\mathrm{T}} x^i, \quad x^i_u + x^i_v \leq 1 \quad \forall uv \in E,\ 0 \leq x^i \leq \mathbf{1},\ x^i \in \mathbb{Z}^V, \tag{7}$$

one for each color $1 \leq i \leq k$. The packing constraints $\sum_{i=1}^{k} x^i \leq \mathbf{1}$ guarantee that each node can take at most one color.

We finish our list of examples here and remark that, in the same way, graph decomposition problems, constrained path packing problems that arise, e.g., in vehicle routing and duty scheduling, and a variety of other problems are also combinatorial packing problems.

2.2 Natural Integrality

The example of the multicommodity flow problem shows that combinatorial packing problems can be hard even if all of the individual subproblems are easy and, in particular, even if complete descriptions of the individual polyhedra are explicitly known. There are, however, cases where the integrality of the individual problems carries over to the entire combinatorial packing problem. We give now two examples of combinatorial packing problems that have this natural integrality property.

The Bipartite 2-Coloring Problem (BIP-2-COL) is the special case of the 2-coloring problem where $G = (V, E)$ is a bipartite graph G. The individual problems are two set packing problems in this graph G. Their IP formulations can be stated as

$$\max\ w^{\mathrm{T}}x^i, \quad Ax^i \leq \mathbf{1}, x^i \geq 0,\ x^i \in \mathbb{Z}^V, \qquad (i = 1, 2) \tag{8}$$

where $A = A(G)$ denotes the edge-node incidence matrix of G. It is well known (see, e.g., Nemhauser & Wolsey (1988) [34], III.1., Corollary 2.9) that the edge-node incidence matrices of bipartite graphs are totally unimodular. Hence, the individual coloring problems are integral.

The IP formulation of the entire bipartite 2-coloring problem reads

$$\begin{array}{lll} \text{(BIP-2-COL)} & \max\ w^{\mathrm{T}}x^1 + w^{\mathrm{T}}x^2 & \\ \text{(i)} & x^1_u + x^1_v \leq 1 & \forall uv \in E \\ \text{(ii)} & x^2_u + x^2_v \leq 1 & \forall uv \in E \\ \text{(iii)} & x^1_v + x^2_v \leq 1 & \forall v \in V \\ \text{(iv)} & x^1_v, x^2_v \geq 0 & \forall v \in V \\ \text{(v)} & x^1_v, x^2_v \in \{0, 1\}^V & \forall v \in V. \end{array} \tag{9}$$

Propositiondd 2.1. *The bipartite 2-coloring problem is naturally integral.*

Proof. We show that the constraint matrix of the bipartite 2-coloring problem is totally unimodular. This is easily done be noting that BIP-2-COL can again be seen as a set packing problem in a larger bipartite graph H. Using the convention to view the ground set of a combinatorial packing problem as a disjoint union of the ground sets of the individual problems, this graph H has as its node set the ground set $V^1 \cup V^2$ of the bipartite 2-coloring problem, where V^1 is a copy of the node set of the first individual coloring problem, and V^2 a copy of the second node set. For every constraint BIP-2-COL (i), there is an edge u^1v^1 between the first copies u^1 and v^1 of nodes u and v; this edge is a copy of the respective edge uv in the first individual problem. Analogously, there is an edge u^2v^2 between the second copies u^2 and v^2 of nodes u and v for every constraint BIP-2-COL (ii); this edge is a copy of the respective edge uv in the second individual problem. The graph H contains thus two disjoint copies G^1 and G^2 of G, one on the nodes V^1, the other one the nodes V^2. The only additional edges between these copies come from the constraints BIP-2-COL (iii). There is an edge v^1v^2 that joins the two copies of each original node for every packing constraint.

Let $X \cup Y$ be a bipartition of the nodes of G. The nodes of H can be partitioned into corresponding copies X^1, Y^1, X^2, and Y^2. Edges run between

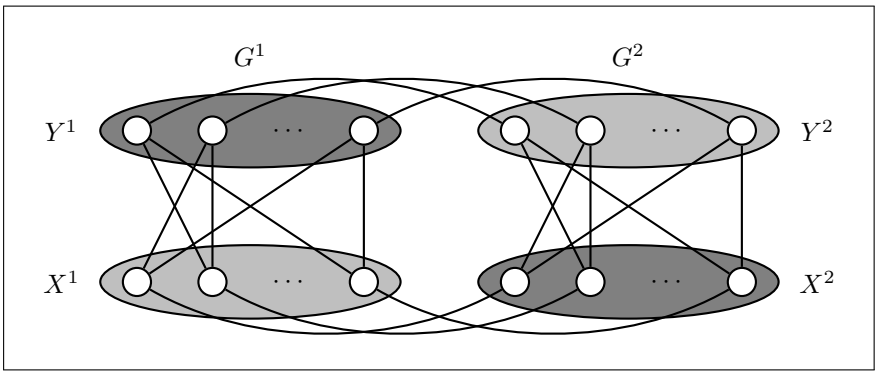


Figure 1: Bipartite 2-Coloring.

X^1 and Y^1 (first copy G^1 of G), X^2 and Y^2 (second copy G^2 of G), X^1 and X^2 (packing constraints on the copies of X), and Y^1 and Y^2 (packing constraints on the copies of Y), see Figure 1. It follows that $(X^1 \cup Y^2) \cup (X^2 \cup Y^1)$ is a bipartition of H. □

The Matroid Packing Problem involves some number k of not necessarily identical matroids on the same ground set E with not necessarily identical non-negative weights $w^1, \ldots, w^k \in \mathbb{Q}_+^E$. The *matroid packing problem* (MPP) is to find a maximum weight collection of independent sets, one from each matroid, such that no two independent sets intersect on a common element.

The matroid packing problem can be stated as the following integer program:

$$\begin{array}{lll} \text{(MPP)} & \max \sum {w^i}^{\mathrm{T}} x^i & \\ \text{(i)} & x^i(F) \leq r^i(F) \quad \forall F \subseteq E, & i = 1, \ldots, k \\ \text{(ii)} & x^i \geq 0 & i = 1, \ldots, k \\ \text{(iii)} & \sum x^i \leq \mathbf{1} & \\ \text{(iv)} & x^i \in \{0,1\}^E & i = 1, \ldots, k. \end{array} \tag{10}$$

Here, r^i denotes the rank function of matroid i. It is known (see, e.g., Nemhauser & Wolsey (1988) [34], Theorem 3.5) that the individual matroid problems are integral.

Propositiondd 2.2. *The matroid packing problem is naturally integral.*

Proof. The reason for the natural integrality of the matroid packing problem is that this problem can be reinterpreted as a matroid intersection problem involving two matroids. Both of these matroids have $E^1 \cup \ldots \cup E^k$ as their ground set. The first matroid is simply the disjoint union of the k

individual matroids. The second matroid is also a disjoint union of k matroids, namely, the $|E|$ uniform matroids that are induced by the packing constraints MPP (iii). Consider the packing constraint $\sum_{i=1}^{k} x_e^i \leq 1$ for element e. The matroid that is associated with this constraint has as its ground set the set $\{e^1, \ldots, e^k\}$ of copies of the element e. The non-trivial independent sets of this matroid are precisely the one-element sets $\{e^1\}, \ldots, \{e^k\}$. The disjoint union of these $|E|$ uniform matroids forms the second matroid.

By definition, MPP (i) and (ii) are a complete polyhedral description for the first matroid. Trivially, MPP (iii) and (ii) are also a complete polyhedral description of the second matroid. It is, however, well known (see, e.g., Nemhauser & Wolsey (1988) [34], III.3., Theorem 5.9) that the union of two such systems is a complete description of the polytope that is associated with the intersection of two matroids. □

Having seen two examples of naturally integral CPPs, a "converse" question that comes up is whether the integrality of the individual problems is a necessary condition for the natural integrality of a CPP. This is true if the individual problems are down monotone. The following example shows, however, that this is not true in general.

Exampledd 2.3. Consider the combinatorial 2-packing problem

$$\begin{array}{rrrrrrrrrrr}
\max & x1 & + & x2 & + & x3 & + & x4 & & \\
 & x1 & + & x2 & & & & & \geq & 1 \\
 & 2x1 & + & 2x2 & & & & & \leq & 3 \\
 & & & & & x3 & + & x4 & \geq & 1 \\
 & & & & & 2x3 & + & 2x4 & \leq & 3 \\
 & & & & & x1, & x2, & x3, x4 & \geq & 0 \\
 & x1 & & & + & x3 & & & \leq & 1 \\
 & & & x2 & & & + & x4 & \leq & 1 \\
 & & & & & x1, & x2, & x3, x4 & \in & \mathbb{Z}.
\end{array} \tag{11}$$

The individual problems produce the polytopes $P_{\mathrm{IP}^i} = \operatorname{conv}\left(\begin{smallmatrix} 0 & \frac{1}{2} & 1 & 1 \\ 1 & 1 & 0 & \frac{1}{2} \end{smallmatrix}\right)$, $i = 1, 2$, which have fractional vertices. The entire CPP is, however, integral; its associated polytope is $P_{\mathrm{CPP}} = \operatorname{conv}\left(\begin{smallmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{smallmatrix}\right)^{\mathrm{T}} = P_{\mathrm{CPP}}$.

3 Dantzig-Wolfe Set Packing Formulations

Combinatorial packing problems give rise to a natural alternative set packing formulation via Dantzig-Wolfe decomposition. This connection creates a possibility to study combinatorial packing problems in terms of set packing

theory. We show in this section that such Dantzig-Wolfe set packing formulations have interesting structural properties that make them potentially useful sources of cutting planes for combinatorial packing problems.

Consider a combinatorial packing problem (2). Let $M^i \in \{0,1\}^{E \times \mathfrak{V}^i}$ be a matrix whose columns are the incidence vectors of the 0/1 solutions of the individual problem IP^i, $i = 1, \ldots, k$. Let us identify the index $\mathfrak{v} \in \mathfrak{V}^i$ of such a column $M^i_{\cdot\mathfrak{v}}$ with the set associated with that column, i.e., we view $\mathfrak{v}$ as a subset of the ground set E^i whose incidence vector is $M^i_{\cdot\mathfrak{v}}$ (i.e., $\chi^{\mathfrak{v}} = M^i_{\cdot\mathfrak{v}}$).

A Dantzig-Wolfe decomposition subject to the substitutions

$$x^i = M^i\lambda^i, \quad \mathbf{1}^{\mathrm{T}}\lambda^i = 1,\ \lambda^i \geq 0,\ \lambda^i \in \{0,1\}^{\mathfrak{V}^i}, \qquad i = 1, \ldots, k, \tag{12}$$

transforms (2) into the form

$$\begin{array}{lrll} \text{(XPP)} & \max \sum_{i=1}^k {w^i}^{\mathrm{T}} M^i\lambda^i & & \\ \text{(i)} & \mathbf{1}^{\mathrm{T}}\lambda^i = 1, & & i = 1, \ldots, k \\ \text{(ii)} & \sum_{i=1}^k M^i\lambda^i \leq \mathbf{1} & & \\ \text{(iii)} & \lambda^i \geq 0, & & i = 1, \ldots, k \\ \text{(iv)} & \lambda^i \in \{0,1\}^{\mathfrak{V}^i}, & & i = 1, \ldots, k. \end{array} \tag{13}$$

We call XPP the *Dantzig-Wolfe formulation* associated with CPP. Constraints XPP (i) are the *convexity constraints*, and constraints XPP (ii) are the *packing constraints*. Introducing the notation $\lambda^{\mathrm{T}} = ({\lambda^1}^{\mathrm{T}}, \ldots, {\lambda^k}^{\mathrm{T}})$, $M = (M^1, \ldots, M^k)$, $C = \mathrm{diag}(\mathbf{1}^{\mathrm{T}})$, $w^{\mathrm{T}} = ({w^1}^{\mathrm{T}}, \ldots, {w^k}^{\mathrm{T}})$, and $\mathfrak{V} = \mathfrak{V}^1 \cup \ldots \cup \mathfrak{V}^k$, XPP becomes

$$\text{(XPP)} \quad \max\ w^{\mathrm{T}}M\lambda, \quad C\lambda = \mathbf{1},\ M\lambda \leq \mathbf{1},\ \lambda \geq 0,\ \lambda \in \{0,1\}^{\mathfrak{V}}. \tag{14}$$

XPP is closely related to the set packing problem

$$\text{(SPP)} \quad \max\ w^{\mathrm{T}}M\lambda, \quad C\lambda \leq \mathbf{1},\ M\lambda \leq \mathbf{1},\ \lambda \geq 0,\ \lambda \in \{0,1\}^{\mathfrak{V}}. \tag{15}$$

In fact, XPP arises from SPP by forcing the relaxed convexity constraints $C\lambda \leq \mathbf{1}$ to equality. This is, however, not an essential change. XPP can, e.g., be transformed into the form SPP by adding a suitably large constant $M \cdot \mathbf{1}$ to the objective. As a (modified) packing problem, XPP can be restated in graph theoretical language in terms of the intersection graph $\mathfrak{G} = (\mathfrak{V}, \mathfrak{E})$ that is associated with the constraint matrix $A = \begin{pmatrix} C \\ M \end{pmatrix}$. This graph $\mathfrak{G}$ has a node $\mathfrak{v} \in \mathfrak{V} = \mathfrak{V}^1 \cup \ldots \cup \mathfrak{V}^k$ for each individual 0/1 solution. There is an edge $\mathfrak{u}\mathfrak{v}$ for any two individual solutions $\mathfrak{u}$ and $\mathfrak{v}$ that can not simultaneously be contained in a packing. This is the case when either $\mathfrak{u}$ and $\mathfrak{v}$ are both solutions of the same individual problem such that the columns $A_{\cdot\mathfrak{u}}$ and $A_{\cdot\mathfrak{v}}$ intersect on a convexity row, or when $\mathfrak{u}$ and $\mathfrak{v}$ contain both the same element

$e \in E$, i.e., $A._{\mathfrak{u}}$ and $A._{\mathfrak{v}}$ intersect on the packing row associated with the element e. In terms of $\mathfrak{G}$, XPP is the problem to find a maximum weight packing in $\mathfrak{G}$ such that each "convexity clique" is covered exactly once. This connection to set packing has polyhedral consequences. Consider the polytopes

$$\begin{aligned} P^I_{\text{XPP}} &= \{\lambda \in \{0,1\}^{\mathfrak{V}} : C\lambda = \mathbf{1}, M\lambda \leq \mathbf{1}\} \\ P^=_I &= \{\lambda \in \{0,1\}^{\mathfrak{V}} : A\lambda \leq \mathbf{1}\} \end{aligned} \tag{16}$$

associated with XPP and SPP and their respective fractional relaxations P_{XPP} and P_{SPP}. The polytope $P^=_I$ is the set packing polytope associated with $\mathfrak{G}$ and P^I_{XPP} is a face of $P^=_I$. The combinatorial packing polytope can be obtained from P^I_{XPP} by projection.

Propositiondd 3.1. *P_{CPP} is the projection of the "extended set packing polytope"*

$$\{x \mid x = \text{diag}(M^i)\lambda,\ \lambda \in P^I_{\text{XPP}}\} \tag{17}$$

on the space of the x-variables.

Proposition 3.1 states that all facets of the combinatorial packing polytope are projections of set packing inequalities in some high dimensional space. This means that it is, at least in principle, possible to study combinatorial packing problems in terms of set packing theory. We remark that such a study is necessary because a Dantzig-Wolfe formulation *per se* does only contain information on the individual problems, but not on packings. Namely, Proposition 3.1 implies the following relationship between CPP and XPP (see, e.g., Sol (1994) [43], Section 2.3 for essentially the same result):

Corollarydd 3.2. *Let CPP be a combinatorial packing problem with integral individual problems and let XPP be its Dantzig-Wolfe formulation. Then:*

The value of the LP relaxation of CPP is equal to the value of the LP relaxation of XPP.

For combinatorial packing problems with integral individual problems such as the multicommodity flow problem, one can therefore not gain much from just restating the problem in column generation form.

The natural way to exploit Proposition 3.1 algorithmically is by using *lift-and-project* techniques, see Balas (1979) [4]; Balas, Ceria & Cornuéjols (1993) [7]. Suppose we want to check some point $\overline{x}$ for membership in P_{CPP}. Suppose also for the moment that we have a complete description $D\lambda \leq d$ of

P^I_{XPP} at hand. Then, by the Farkas lemma,

$$\begin{aligned} & \overline{x} \notin P_{\text{CPP}} \\ \iff\ & \{\lambda \mid D\lambda \leq d, \operatorname{diag}(M^i)\lambda = \overline{x}\} = \emptyset \\ \iff\ & \exists a, b : a^{\mathrm{T}}D + b^{\mathrm{T}}\operatorname{diag}(M^i) \geq 0,\ a \geq 0,\ a^{\mathrm{T}}d + b^{\mathrm{T}}\overline{x} < 0. \end{aligned} \tag{18}$$

However, as $0 \leq a^{\mathrm{T}}D\lambda + b^{\mathrm{T}}\operatorname{diag}(M^i)\lambda \leq a^{\mathrm{T}}d + b^{\mathrm{T}}x$ is valid for any $x \in P_{\text{CPP}}$, the inequality

$$a^{\mathrm{T}}d + b^{\mathrm{T}}x \geq 0 \tag{19}$$

is a valid inequality for P_{CPP} that is violated by $\overline{x}$; such a cut can be determined by solving an appropriate LP (involving an additional normalization constraint to bound the recession cone).

Ignoring the technical difficulty of this projection process for the moment, the success of the procedure clearly depends on the quality of the description $D\lambda \leq d$ for P^I_{XPP}. Knowledge of a *complete* description of P^I_{XPP} is surely an elusive goal in general. There are, however, significant cases where such a complete description is, in some sense, in fact available.

Propositiondd 3.3. *The intersection graph associated with the Dantzig-Wolfe formulation of a combinatorial 2-packing problem is perfect.*

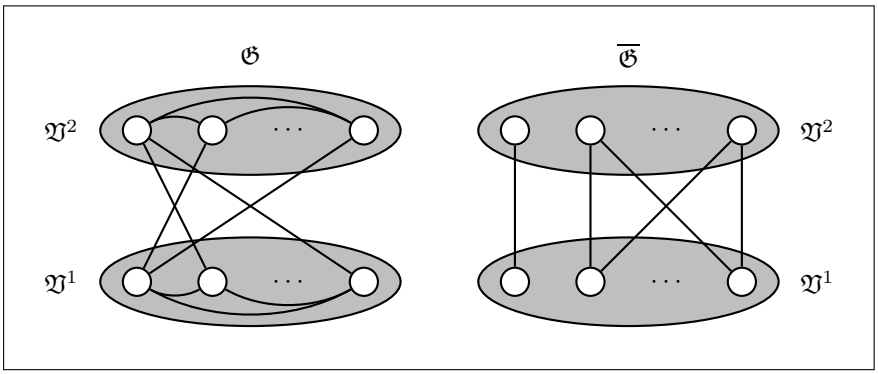


Figure 2: Intersection Graph of a Combinatorial 2-Packing Problem.

Proof. We show that $\mathfrak{G}$ is the complement of a bipartite graph. The nodes of $\mathfrak{G}$ consist of the two sets $\mathfrak{V} = \mathfrak{V}^1 \cup \mathfrak{V}^2$ that correspond to the solutions of the first and the second individual problem, respectively. As there can be only one solution of each individual problem in a packing, the nodes of $\mathfrak{V}^1$ and $\mathfrak{V}^2$ form two cliques in $\mathfrak{G}$. These cliques are joined by the remaining edges, connecting solutions that have elements from the ground set in common, see Figure 2. In the complement graph $\overline{\mathfrak{G}}$, the sets $\mathfrak{V}^1$ and $\mathfrak{V}^2$ form two stable sets. Therefore, they induce a bipartition in $\overline{\mathfrak{G}}$, and hence $\overline{\mathfrak{G}}$ is perfect. □

Proposition 3.3 shows that all facets of combinatorial 2-packing polytopes are projections of *clique inequalities*, see Padberg (1973) [36]. The clique inequalities are subsumed by the larger class of *orthonormal representation constraints* which can be separated in polynomial time, see Grötschel, Lovász & Schrijver (1988) [20]. Proposition 3.3 suggests that such separation techniques, combined with lift-and-project methods, are potentially useful tools for the solution of combinatorial 2-packing problems. We remark that such techniques can, however, not lead to polynomial time algorithms for general combinatorial 2-packing problems, because this class contains $\mathcal{NP}$-hard problems such as the 2-commodity flow problem with unit capacities, see Garey & Johnson (1979) [17], Problem ND38.

A practical use of lift-and-project cutting planes from Dantzig-Wolfe formulations can not be that one builds up a larger and larger description of P^I_{XPP} in the exponential space $\mathbb{R}^{\mathfrak{V}}$, adding more and more cutting planes and columns. Doing so would be equivalent to a combined column generation and cutting plane approach to combinatorial packing problems with its well-known difficulties. Instead, we propose to accumulate cutting planes only in the compact original space $\mathbb{R}^{\dot{\cup} E^i}$, and to use the Dantzig-Wolfe formulation solely as a separation tool.

The straightforward way to do that is as follows. Suppose we are given a point $\overline{x}$ to be tested for membership in P_{CPP}. The first step is to express $\overline{x}$ as a convex combination of individual solutions in the form $\overline{x}^i = M^i\lambda^i$, $i = 1, \ldots, k$. By Caratheodory's theorem, this can be done in such a way that the resulting multipliers λ^i have at most $|E| + 1$ nonzero components each. We then set up a subproblem of XPP that consists of the columns that appear in these convex combinations, apply whatever separation algorithms we have at hand, and add the resulting cuts. Projecting back, we have to be careful that our cut is dual feasible for the global XPP, i.e., we potentially have to lift a number of additional variables (this can happen because there may be more than one way to express $\overline{x}$ as a convex combination of 0/1 solutions). When this process results in a violated cutting plane for P_{CPP}, we add it to our current description of P_{CPP}, resolve, and iterate. The procedure that we have just sketched is admittedly expensive, but it points into a direction of a possible future algorithmic use of structural results such as Proposition 3.3.

We close this paper with an example that is supposed to avoid a possible misunderstanding. Proposition 3.3 does *not* make a statement that would relate perfection of the constraint matrix A of a Dantzig-Wolfe formulation or its intersection graph $\mathfrak{G}$ with natural integrality of the original formulation. The obstacle that prevents us from establishing such a connection is that the LP relaxation of a Dantzig-Wolfe formulation can have fractional vertices that correspond to integral packings.

Exampledd 3.4. Consider the following combinatorial packing problem with two uniform matroids of rank 2.

$$\begin{array}{lcccccccccccl}
\max & x_1^1 & + & x_2^1 & + & x_3^1 & + & x_1^2 & + & x_2^2 & + & x_3^2 & \\
 & x_1^1 & + & x_2^1 & + & x_3^1 & & & & & & & \leq 2 \\
 & & & & & & & x_1^2 & + & x_2^2 & + & x_3^2 & \leq 2 \\
 & x_1^1 & & & & & + & x_1^2 & & & & & \leq 1 \\
 & & & x_2^1 & & & & & + & x_2^2 & & & \leq 1 \\
 & & & & & x_3^1 & & & & & + & x_3^2 & \leq 1 \\
 & x_1^1 & & x_2^1, & & x_3^1, & & x_1^2, & & x_2^2, & & x_3^2 & \geq 0 \\
 & x_1^1 & & x_2^1, & & x_3^1, & & x_1^2, & & x_2^2, & & x_3^2 & \in \mathbb{Z}.
\end{array} \tag{20}$$

By Proposition 2.2, the problem is naturally integral. The Dantzig-Wolfe formulation is

$$\begin{array}{l}
\max\ (0,1,1,1,2,2,2)\lambda^1 + (0,1,1,1,2,2,2)\lambda^2 \\
\begin{pmatrix}
1 & \colorbox{lightgray}{1} & \colorbox{lightgray}{1} & 1 & 1 & 1 & 1 & & & & & & & \\
 & & & & & & & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
 & \colorbox{lightgray}{1} & & & 1 & 1 & & & 1 & & & \colorbox{lightgray}{1} & 1 & \\
 & & \colorbox{lightgray}{1} & & 1 & & 1 & & & 1 & & \colorbox{lightgray}{1} & & 1 \\
 & & & 1 & & 1 & 1 & & & & 1 & & 1 & 1
\end{pmatrix}
\begin{pmatrix} \lambda^1 \\ \lambda^2 \end{pmatrix}
\begin{array}{l} = 1 \\ = 1 \\ \leq 1 \\ \leq 1 \\ \leq 1 \end{array} \\
\hfill \lambda^1, \lambda^2 \ \geq 0 \\
\hfill \lambda^1, \lambda^2 \ \in \mathbb{Z}^7.
\end{array} \tag{21}$$

The constraint matrix A of this formulation is not perfect. The perfect clique matrix associated with the intersection graph of the Dantzig-Wolfe formulation is

$$\begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & & & & & & & \\
 & & & & & & & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
 & 1 & & & 1 & 1 & & & 1 & & & 1 & 1 & \\
 & & 1 & & 1 & & 1 & & & 1 & & 1 & & 1 \\
 & & & 1 & & 1 & 1 & & & & 1 & & 1 & 1 \\
\hline
 & & & & & & 1 & & & 1 & 1 & 1 & 1 & 1 \\
 & & & & & 1 & & & 1 & & 1 & 1 & 1 & 1 \\
 & & & & & 1 & 1 & & & & 1 & 1 & 1 & 1 \\
 & & & & 1 & & & & 1 & 1 & & 1 & 1 & 1 \\
 & & & & 1 & & 1 & & & 1 & & 1 & 1 & 1 \\
 & & & & 1 & 1 & & & 1 & & & 1 & 1 & 1 \\
 & & & & 1 & 1 & 1 & & & & & 1 & 1 & 1 \\
 & & & 1 & 1 & 1 & 1 & & & & & & 1 & 1 \\
 & & 1 & & 1 & 1 & 1 & & & & & 1 & & 1 \\
 & & 1 & 1 & 1 & 1 & 1 & & & & & & & 1 \\
 & 1 & & & 1 & 1 & 1 & & & & & 1 & 1 & \\
 & 1 & & 1 & 1 & 1 & 1 & & & & & & 1 & \\
 & 1 & 1 & & 1 & 1 & 1 & & & & & 1 & &
\end{pmatrix}. \tag{22}$$

This matrix adds 13 missing cliques to A. The clique in the last row contains the highlighted columns of A.

Similarly, one can verify that the 3-packing problem associated with three uniform matroids of rank 2 has an imperfect intersection graph.

Acknowledgement. I would like to thank an anonymous referee for helpful comments and suggestions, among them the idea to investigate the question behind Example 2.3. I would also like to thank Robert Weismantel and Alexander Martin for helpful comments.

References

[1] R. K. Ahuja, T. L. Magnanti & J. B. Orlin. *Network Flows*, chap. IV, pp. 211–369. In vol. 1 of Nemhauser, Rinnooy Kan & Todd (1989) [35], 1989. Cited on page 35.

[2] A. Atamturk, G. L. Nemhauser & M. W. P. Savelsbergh. *Conflict graphs in solving integer programming problems*. European J. Oper. Res. **121**(1):40–55, 1998. Cited on page 33.

[3] A. Atamtrk, G. L. Nemhauser & M. W. P. Savelsbergh. *The mixed vertex packing problem*. Math. Programming **89**:2000, 2000. Cited on page 33.

[4] E. Balas. *Disjunctive programming*. Ann. Discrete Math. **5**:3–51, 1979. Cited on page 41.

[5] E. Balas & J. Clausen, (Eds.). *Integer Programming and Combinatorial Optimiziation*, Proc. 4th Int. IPCO Conf., 1995. Cited on page 47.

[6] E. Balas & M. Padberg. *Set partitioning: A survey*. SIAM Rev. **18**: 710–760, 1976. Cited on page 32.

[7] E. Balas, S. Ceria & G. Cornuéjols. *A lift-and-project cutting plane algorithm for mixed 0-1 programs*. Math. Programming **58**:295–324, 1993. Cited on page 41.

[8] R. Borndörfer. *Aspects of Set Packing, Partitioning, and Covering*. Berichte aus der Mathematik. Shaker Verlag, Aachen, 1998. ISBN 3-8265-4351-3. URL http://www.shaker.de/Online-Gesamtkatalog/Details.idc?ISBN=3-8265-4351-3. PhD thesis, Technische Universität Berlin, 1998. Available at http://www.zib.de/bib/books/borndoerfer.thesis.ps. Cited on page 32.

[9] R. Borndörfer & R. Weismantel. *Set packing relaxations of some integer programs*. Math. Programming **88**:425–450, 2000. Preprint ZIB Report 97-30 available at URL http://opus.kobv.de/zib/volltexte/1997/300/. Cited on page 33.

[10] R. Borndörfer & R. Weismantel. *Discrete relaxations of combinatorial programs*. Discrete Appl. Math. **112**(1–3):11–26, 2001.

ZIB preprint SC 97-54 available at URL http://opus.kobv.de/zib/volltexte/1997/324/. Cited on page 33.

[11] W. H. Cunningham, S. T. McCormick & M. Queyranne, (Eds.). *Integer Programming and Combinatorial Optimization*, Proc. 5th Int. IPCO Conf., 1996. Cited on page 47.

[12] M. Deza & M. Laurent. *Geometry of Cuts and Metrics*. Springer Verlag, Berlin, 1997. Cited on page 35.

[13] R. Euler, M. Jünger & G. Reinelt. *Generalizations of cliques, odd cycles and anticycles and their relation to independence system polyhedra*. Math. Oper. Res. **12**(3):451–462, August 1987. Cited on page 33.

[14] C. E. Ferreira. *On Combinatorial Optimization Problems Arising in Computer System Design*. PhD thesis, Technische Universität Berlin, 1994. URL http://www.zib.de/bib/books/Ferreira.disser.ps. Cited on page 36.

[15] C. E. Ferreira, A. Martin & R. Weismantel. *Solving multiple knapsack problems by cutting planes*. SIAM J. Optim. **6**:858–877, 1996. Cited on page 36.

[16] A. Frank. *Packing paths, circuits, and cuts – a survey*. In Korte et al. (1990) [25], chap. 2, pp. 47–100. Cited on pages 32, 35.

[17] M. Garey & D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979. Cited on page 43.

[18] E. Gottlieb & M. Rao. *The generalized assignment problem: Valid inequalities and facets*. Math. Programming **46**:31–52, 1990. Cited on page 36.

[19] R. Graham, M. Grötschel & L. Lovász, (Eds.). *Handbook of Combinatorics*. Elsevier Science B. V., Amsterdam, 1995.

[20] M. Grötschel, L. Lovász & A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, vol. 2 of *Algorithms and Combinatorics*. Springer Verlag, Berlin, 1988. ISBN 3-540-13624-X, 0-387-13624-X (U.S.). Cited on pages 32, 33, 43.

[21] M. Grötschel, A. Martin & R. Weismantel. *Packing Steiner trees: Polyhedral investigations*. Math. Programming **72**:101–123, 1996. ZIB preprint SC 92-08 available at http://opus.kobv.de/zib/volltexte/1992/77/. Cited on page 35.

[22] M. Grötschel, A. Martin & R. Weismantel. *Packing Steiner trees: A cutting plane algorithm and computational results*. Math. Programming **72**:125–145, 1996. ZIB preprint SC 92-09 available at http://opus.kobv.de/zib/volltexte/1992/78/. Cited on page 35.

[23] M. Grötschel, A. Martin & R. Weismantel. *Packing Steiner trees: Further facets*. European J. Combin. **17**:39–52, 1996. ZIB

preprint SC 93-01 available at http://opus.kobv.de/zib/volltexte/1992/96/. Cited on page 35.

[24] M. Grötschel, A. Martin & R. Weismantel. *Packing Steiner trees: Separation algorithms.* SIAM J. Discrete Math. **9**:233–257, 1996. ZIB preprint SC 93-02 available at http://opus.kobv.de/zib/volltexte/1992/97/. Cited on page 35.

[25] B. Korte, L. Lovász, H. Prömel & A. Schrijver, (Eds.). *Paths, Flows, and VLSI-Layout.* Springer Verlag, Berlin, 1990. Cited on page 46.

[26] M. Laurent. *A generalization of antiwebs to independence systems and their canonical facets.* Math. Programming **45**:97–108, 1989. Cited on page 33.

[27] L. Lovász & A. Schrijver. *Cones of matrices and set-functions and 0-1 optimization.* SIAM J. Optim. **1**:166–190, 1991. Cited on pages 2, 33.

[28] S. Martello & P. Toth. *Knapsack Problems.* John Wiley & Sons Ltd, Chichester, 1990. Cited on page 36.

[29] A. Martin. *Packen von Steinerbäumen: Polyedrische Studien und Anwendung.* PhD thesis, Technische Universität Berlin, 1992. URL http://www.zib.de/Publications/abstracts/TR-92-04/. Cited on page 35.

[30] R. Müller. *On the partial order polytope of a digraph.* Math. Programming **73**(1):31–49, 1996. Cited on page 33.

[31] R. Müller & A. S. Schulz. *The interval order polytope of a digraph.* In Balas & Clausen (1995) [5], pp. 50–64. Cited on page 33.

[32] R. Müller & A. S. Schulz. *Transitive packing.* In Cunningham, McCormick & Queyranne (1996) [11], pp. 430–444. Cited on page 33.

[33] G. L. Nemhauser & L. E. Trotter, Jr. *Properties of vertex packing and independence system polyhedra.* Math. Programming **6**:48–61, 1973. Cited on page 33.

[34] G. L. Nemhauser & L. A. Wolsey. *Integer and Combinatorial Optimization.* Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons Ltd, New York, 1988. Cited on pages 32, 37, 38, 39.

[35] G. L. Nemhauser, A. H. G. Rinnooy Kan & M. J. Todd, (Eds.). *Optimization*, vol. 1 of *Handbooks in OR and Management Science.* Elsevier Science B. V., Amsterdam, 1989. Cited on page 45.

[36] M. W. Padberg. *On the facial structure of set packing polyhedra.* Math. Programming **5**:199–215, 1973. Cited on pages 33, 43.

[37] M. W. Padberg. *A note on zero-one programming.* Oper. Res. **23**(4): 833–837, 1975. Cited on page 33.

[38] M. W. Padberg. *On the complexity of set packing polyhedra.* Ann. Discrete Math. **1**:421–434, 1977. Cited on page 33.

[39] M. W. Padberg. *Covering, packing, and knapsack problems.* Ann. Discrete Math. **4**:265–287, 1979. Cited on page 32.

[40] M. W. Padberg & T.-Y. Sung. *An analytical comparison of different formulations of the travelling salesman problem.* Math. Programming **52**(2):315–357, 1991. Cited on page 33.

[41] A. S. Schulz. *Polytopes and Scheduling.* PhD thesis, Technische Universität Berlin, 1996. Cited on page 33.

[42] Y. Sekiguchi. *A note on node packing polytopes on hypergraphs.* Oper. Res. Lett. **2**(5):243–247, 1983. Cited on page 33.

[43] M. Sol. *Column Generation Techniques for Pickup and Delivery Problems.* PhD thesis, Technische Universiteit Eindhoven, 1994. Cited on page 41.

[44] B. Toft. *Colouring, stable sets and perfect graphs.* In Graham, Grötschel & Lovász (1995) [19], chap. 4, pp. 233–288. Cited on page 36.

[45] L. E. Trotter, Jr. *A class of facet producing graphs for vertex packing polyhedra.* Discrete Math. **12**:373–388, 1975. Cited on page 33.

Paper III

Decomposing Matrices into Blocks

R. Borndörfer, C. E. Ferreira & A. Martin.
Decomposing matrices into blocks.
SIAM Journal on Optimization **9**(1):236–269, 1998.

Abstract. In this paper we investigate whether matrices arising from linear or integer programming problems can be decomposed into so-called *bordered block diagonal form*. More precisely, given some matrix A, we try to assign as many rows as possible to some number β of blocks of size κ such that no two rows assigned to different blocks intersect in a common column. Bordered block diagonal form is desirable because it can guide and speed up the solution process for linear and integer programming problems. We show that various matrices from the LP- and MIP-libraries `Netlib` and `Miplib` can indeed be decomposed into this form by computing optimal decompositions or decompositions with proven quality. These computations are done with a branch-and-cut algorithm based on polyhedral investigations of the matrix decomposition problem. In practice, however, one would use heuristics to find a good decomposition. We present several heuristic ideas and test their performance. Finally, we investigate the usefulness of optimal matrix decompositions into bordered block diagonal form for integer programming by using such decompositions to guide the branching process in a branch-and-cut code for general mixed integer programs.

Mathematics Subject Classification (MSC 2000). 90C10, 65F50

Keywords. Block structure of a sparse matrix, matrix decomposition, integer programming, polyhedral combinatorics, cutting planes

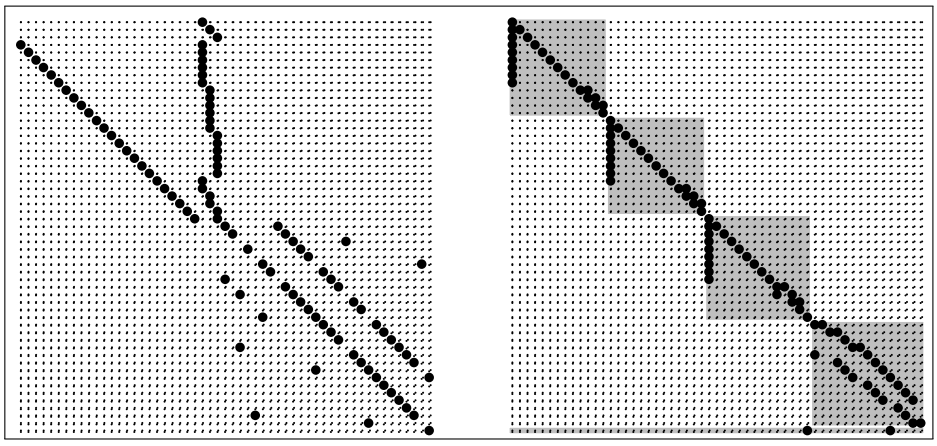


Figure 1: Decomposing a Matrix into Bordered Block Diagonal Form.

1 Introduction

We consider in this paper the following *matrix decomposition problem*. Given some matrix A, some number β of *blocks* (sets of rows), and some *capacity* κ (maximum block-size); try to assign as many rows as possible to the blocks such that (i) each row is assigned to at most one block, (ii) each block contains at most κ rows, and (iii) no two rows in different blocks have a common nonzero entry in a column. The set of rows that are not assigned to any block is called the *border*.

An equivalent statement of the problem in matrix terminology is as follows: Try to decompose the matrix into *bordered block diagonal form* with β blocks of capacity at most κ. The decomposition is considered the better, the smaller the number of rows in the border is; in the best case the border will be empty and the matrix decomposes into block diagonal form. Figure 1 shows on its left side the structure of a 55×55 non-symmetric matrix, namely, an optimal basis matrix of the `Netlib`-problem `recipe`. The right side of Figure 1 shows an optimal decomposition of this matrix into four blocks of capacity $\lceil 55/4 \rceil = 14$. To make the block structure of the decomposition visible, we have not only permuted the rows such that rows assigned to the same block appear consecutively, but also the columns. In this case, the blocks turn out to be almost square of sizes 13×13, 13×13, 14×14, and 14×15, but in general this does not need to be the case. The border consists of only one row that could not be assigned to any block.

The matrix decomposition problem fits into the general context of reordering matrices to *special forms*. Special forms are well studied in the literature because they can be exploited by solution methods for linear equation systems, for example by LU- or Cholesky factorization, or by conjugate gradient methods. The two main points of interest are that special forms allow (i)

to control fill-in (bordered block diagonal form, in particular, restricts fill-in to the blocks and the border) and (ii) independent processing of individual blocks by parallel algorithms.

Methods to obtain special forms, including (bordered) block diagonal form, are widely discussed in the literature of computational linear algebra, see, for instance, Duff, Erisman & Reid (1986) [4], Kumar et al. (1994) [18], or Gallivan et al. (1990) [9]. The matrices studied in this context mainly arise from the discretization of partial differential equations. Some newer publications deal with matrices that appear in interior point algorithms for linear programs, see Gupta (1996) [14] and Rothberg & Hendrickson (1998) [25].

The applications we have in mind are different. We are interested in matrices arising from *(mixed) integer programs* (MIPs). Such matrices often have potential for decomposition into bordered block diagonal form for two reasons. First, such matrices are sparse and generally have a small number of nonzero entries in each column. Second, bordered block diagonal form comes up in a natural way in many real world MIPs. The problems are often composed of small blocks to model decisions in a division of a company, in a technical unit, or in a time period. These individual blocks are linked by a couple of constraints that model possible interactions to yield an appropriate model that covers the whole company, technical process, or time horizon. Examples of this type are production planning problems like the unit commitment problem in power generation, where the blocks correspond to single unit subproblems and the border is given by load balance and reserve constraints, see Sheble & Fahd (1994) [27] for a literature synopsis and Dentcheva et al. (1997) [3] for a recent application, multicommodity flow problems that come up, for example, in vehicle scheduling, see Löbel (1998) [20], classes of combinatorial programs like the Steiner tree packing problem, see Grötschel, Martin & Weismantel (1996) [13], or, recently, scenario decompositions of stochastic mixed integer programs, see Carøe & Schultz (1999) [1].

Bordered block diagonal form helps to *accelerate* the solution process of integer programs in several ways. First, to solve the LP-relaxation of an integer program; here bordered block diagonal form can speed up the required linear algebra (both if a simplex type method or an interior point method is used). Second, it can be used to improve the polyhedral description of the set of feasible points of a MIP. For example, given a block decomposition and taking one constraint from each block plus an according number from the border results in the structure of a generalized assignment or multiple knapsack problem (see Gottlieb & Rao (1990) [11] and Ferreira, Martin & Weismantel (1996) [6]) whose facets are valid inequalities for the MIP under consideration. Third, block decomposition can be used in a branch-and-

bound or -cut algorithm to guide branching decisions: Decomposing the transposed constraint matrix A^T will identify the columns in the border as linking columns that are interesting candidates for branching.

In this paper we develop a branch-and-cut algorithm for solving the matrix decomposition problem. Of course the expected running time of such an algorithm will neither permit its usage within a parallel LU-factorization nor within a branch-and-cut algorithm for general MIPs. Our aim is rather to have a tool at hand that in principle obtains an *optimal* bordered block diagonal form. We can then evaluate whether this special matrix structure indeed helps in solving general integer programs, and we can evaluate the success of decomposition heuristics that try to obtain (bordered) block diagonal form and that could be used, for instance, within a parallel LU-factorization framework.

The paper is organized as follows. In Section 2 we formulate the matrix decomposition problem as a 0/1 linear program and discuss connections to related combinatorial optimization problems, namely, node separation problems in graphs, the set packing, and the set covering problem. Section 3 is devoted to a polyhedral investigation of the matrix decomposition problem and presents (new) valid and facet defining inequalities. In the branch-and-cut Section 4 we present our matrix decomposition algorithm including separation routines, primal heuristics, preprocessing, and other aspects of the implementation. We use this code in Section 5 to decompose optimal basis matrices of linear programs taken from the `Netlib` (available by anonymous `ftp` from `ftp://netlib2.cs.utk.edu`), to decompose matrices arising from general mixed integer programs from the `Miplib` (available from `URL` `http://www.caam.rice.edu:80/$\sim$bixby/miplib/miplib.html`), and to solve some equipartition problems investigated by Nicoloso & Nobili (1992) [21].

2 Integer Programming Formulation and Related Problems

Consider an instance (A, β, κ) of the matrix decomposition problem where $A \in \mathbb{R}^{m \times n}$ is some real matrix, $\beta \in \mathbb{N}$ is the number of blocks and $\kappa \in \mathbb{N}$ is the block capacity. We introduce for each row $i = 1, \ldots, m$ and block $b = 1, \ldots, \beta$ a binary variable x_i^b that has value 1 if row i is assigned to block b and 0 otherwise. Then the matrix decomposition problem (A, β, κ) can be formulated as the 0/1 linear program (IP) that is stated on the next page.

Inequalities (i) guarantee that each row is assigned to at most one block.

Constraints (ii) ensure that the number of rows assigned to a particular block b does not exceed its capacity. Finally, (iii) expresses that two rows i and j must not be assigned to different blocks if both have a nonzero entry in some common column. These three sets of inequalities plus the bounds (iv) and the integrality constraints (v) establish a one-to-one correspondence between feasible solutions of (IP) and block decompositions of the matrix A into β blocks of capacity κ. In the sequel we will also call a vector $x \in \mathbb{R}^{m\times\beta}$ a *block decomposition* if it is feasible for (IP). Note that formulation (IP) as it stands is not polynomial, since the number of variables $m\beta$ is not polynomial in the encoding length of β. However, we may assume without loss of generality $\beta \leq m$, because no more than m rows will be assigned. We also assume that the block capacity is at least one ($\kappa \geq 1$) and that we have at least two blocks ($\beta \geq 2$).

$$
\begin{array}{lll}
\max & \displaystyle\sum_{i=1}^{m}\sum_{b=1}^{\beta} x_i^b & \\
\text{(i)} & \displaystyle\sum_{b=1}^{\beta} x_i^b \leq 1, & \text{for } i = 1,\ldots,m; \\
\text{(ii)} & \displaystyle\sum_{i=1}^{m} x_i^b \leq \kappa, & \text{for } b = 1,\ldots,\beta; \\
\text{(iii)} & x_i^b + x_j^{b'} \leq 1, & \text{for } b, b' = 1,\ldots,\beta,\ b \neq b' \text{ and} \\
 & & \text{for } i, j = 1,\ldots,m,\ i \neq j \text{ such that} \\
 & & a_{ik} \neq 0 \neq a_{jk} \text{ for some } k \in \{1,\ldots,n\}; \\
\text{(iv)} & 0 \leq x_i^b \leq 1, & \text{for } i = 1,\ldots,m,\ b = 1,\ldots,\beta; \\
\text{(v)} & x_i^b \text{ integer}, & \text{for } i = 1,\ldots,m,\ b = 1,\ldots,\beta.
\end{array}
\qquad \text{(IP)}
$$

A first observation about (IP) is that different matrices A can give rise to the same integer program or, in other words, different matrices can be decomposed in exactly the same way. In fact, such matrices form equivalence classes as can be seen by considering the *(column) intersection graph* $G(A)$ of an $m \times n$-matrix A as introduced by Padberg (1973) [23]. $G(A)$ has the set $\{1,\ldots,n\}$ of column indices of A as its node set and there is an edge ij between two columns i and j if they have a common nonzero entry in some row. Applying this concept to the transposed matrix A^T, we obtain the *row intersection graph* $G(A^T)$ of A where two rows i and j are joined by an edge ij if and only if they have nonzero entries in a common column. But then the edges of $G(A^T)$ give rise to the inequalities (IP) (iii) and we have that for fixed β and κ two matrices A and A' have the same row intersection graph if and only if the corresponding integer programs (IP) are equal.

The matrix decomposition problem is related to several other combinatorial optimization problems. First, the problem can be interpreted in terms of the row intersection graph as a *node separator problem*. To see this, let $G(A^T) = (V, E)$ and consider some block decomposition x. The set $S := \{i \in V : \sum_{b=1}^{\beta} x_i^b = 0\}$ of rows in the border is a node separator in $G(A^T)$ such that the graph obtained by deleting all nodes in S and all its adjacent edges decomposes into at most β parts, each of cardinality at most κ. Conversely, each node separator in $G(A^T)$ with these properties gives rise to a block decomposition for (A, β, κ). Various node separator problems have been studied in the literature. Lengauer (1990) [19] gives a survey and discusses applications in VLSI design, Duff, Erisman & Reid (1986) [4] and Gallivan et al. (1990) [9] emphasize heuristic methods for use in computational linear algebra. Lower bounds on the size of a node separator in a general graph are rather rare. The only results we are aware of are due to Pothen, Simon & Liou (1990) [24] and Helmberg et al. (1995) [15], who use Eigenvalue methods to derive non-trivial lower bounds on the size of a node separator for $\beta = 2$, if lower bounds on the size of the blocks are imposed.

A second connection exists to *set packing*, and this relationship is two-fold. On the one hand, matrix decomposition is a generalization of set packing, because feasible solutions (stable sets) of some set packing problem $\max\{\mathbf{1}^T x : Ax \leq \mathbf{1}, x \in \{0,1\}^n\}$, $A \in \{0,1\}^{m \times n}$, correspond to solutions of the matrix decomposition problem $(A^T, m, 1)$ of the same objective value and vice versa. This shows that the matrix decomposition problem is $\mathcal{NP}$-hard. On the other hand, we obtain a *set packing relaxation* of the matrix decomposition problem by deleting the block capacity constraints (ii) from the formulation (IP). All inequalities that are valid for this relaxation are also valid for the matrix decomposition problem and we will use some of them (namely clique- and cycle-inequalities) as cutting planes in our branch-and-cut algorithm. Note, however, that the set packing relaxation allows assignment of all rows to any single block and our computational experiments seem to indicate that these cuts are rather weak.

A close connection exists also to *set covering* via complementing variables. To see this we rewrite (IP), substituting each capacity constraint (ii) by $\binom{m}{\kappa+1}$ inequalities that sum over all subsets of cardinality $\kappa + 1$ of variables $\{x_1^b, \ldots, x_m^b\}$ for some block b and each constraint in (i) by $\binom{\kappa}{2}$ inequalities that sum over all pairs of variables in $\{x_i^1, \ldots, x_i^\beta\}$. Replacing all variables x_i^b

by $1 - y_i^b$, one obtains the following set covering problem:

$$
\begin{array}{lll}
\min & \displaystyle\sum_{i=1}^{m}\sum_{b=1}^{\beta} y_i^b & \\
\text{(i)} & y_i^b + y_i^{b'} \geq 1, & \text{for } b, b' = 1, \ldots, \beta,\ b \neq b' \text{ and} \\
 & & \text{for } i = 1, \ldots, m; \\
\text{(ii)} & \displaystyle\sum_{i \in I} y_i^b \geq 1, & \text{for } b = 1, \ldots, \beta \text{ and} \\
 & & \text{for } I \subseteq \{1, \ldots, m\} \text{ with } |I| = \kappa + 1; \\
\text{(iii)} & y_i^b + y_j^{b'} \geq 1, & \text{for } b, b' = 1, \ldots, \beta,\ b \neq b' \text{ and} \\
 & & \text{for } i, j = 1, \ldots, m,\ i \neq j \text{ such that} \\
 & & a_{ik} \neq 0 \neq a_{jk} \text{ for some } k \in \{1, \ldots, n\}; \\
\text{(iv)} & 0 \leq y_i^b \leq 1, & \text{for } i = 1, \ldots, m,\ b = 1, \ldots, \beta; \\
\text{(v)} & y_i^b \text{ integer}, & \text{for } i = 1, \ldots, m,\ b = 1, \ldots, \beta.
\end{array}
\tag{IP$_c$}
$$

This shows that the matrix decomposition problem is a (special) set covering problem. For the case of two blocks, this formulation has been used by Nicoloso & Nobili (1992) [21] for the solution of the *matrix equipartition problem*. The matrix equipartition problem is the matrix decomposition problem for $\beta = 2$ and $\kappa = \lfloor m/2 \rfloor$, plus the additional equipartition constraint

$$
\sum_{i=1}^{m} x_i^1 = \sum_{i=1}^{m} x_i^2, \quad \text{or, in complemented variables,} \quad \sum_{i=1}^{m} y_i^1 = \sum_{i=1}^{m} y_i^2,
$$

that states that the two blocks of the decomposition must have equal size.

3 Polyhedral Investigations

Associated to the IP-formulation (IP) of the matrix decomposition problem is the polytope

$$
P(A, \beta, \kappa) := \text{conv}\{x \in \mathbb{R}^{m \times \beta} : \ x \text{ satisfies (IP) (i) to (v)}\}, \tag{1}
$$

given by the convex hull of all block decompositions. We study in this section the structure of $P(A, \beta, \kappa)$ to derive classes of valid and facet defining inequalities for later use as cutting planes. We start by determining its dimension.

Propositiondd 3.1 (Dimension). *$P(A, \beta, \kappa)$ is full dimensional.*

Proof. The vector 0 and all unit vectors $e_i^b \in \mathbb{R}^{m\times\beta}$ are feasible, i. e., are in $P(A,\beta,\kappa)$, and affinely independent. □

This means that the facets of $P(A,\beta,\kappa)$ are uniquely determined up to a scalar factor (see Schrijver (1986) [26]). Two further easy observations are gathered in the following remark.

Remarkdd 3.2. *(i) The non-negativity inequalities $x_i^b \geq 0$ are facet defining for all $i = 1,\ldots,m$ and all $b = 1,\ldots,\beta$.*

(ii) All facet defining inequalities $a^T x \leq \alpha$ that are not non-negativity constraints satisfy $a \geq 0$ and $\alpha > 0$.

Remark 3.2 (i) is proven in the same way as Theorem 3.1 and Remark 3.2 (ii) is a consequence of the down monotonicity of $P(A,\beta,\kappa)$.

Facet-defining inequalities have another interesting property. Consider some vector $x \in \mathbb{R}^{m\times\beta}$, some *permutation σ of the blocks* $\{1,\ldots,\beta\}$, and define the vector $\bar{x} \in \mathbb{R}^{m\times\beta}$ by

$$\bar{x}_i^b := x_i^{\sigma(b)},$$

for $i = 1,\ldots,m, b = 1,\ldots,\beta$. We will use in the sequel the symbol $\sigma(x)$ to denote the vector $\bar{x}$ that arises from x by applying the block permutation σ. Then $\sigma(x) = \bar{x}$ is a feasible block decomposition if and only if x is. This simple observation has two consequences. First, it implies that $a^{\mathrm{T}}x \leq b$ is a facet of $P(A,\beta,\kappa)$ if and only if its block-wise permutation $\sigma(a)^{\mathrm{T}}x \leq b$ is. Facets arising from each other via block permutations can thus be viewed as forming a single class that can be represented by a single member. Or, to put it in a more negative way, each facet can and will be "blown up" by block permutations to a whole set of combinatorially essentially identical conditions. Second, the objective function of the matrix decomposition problem is invariant under block permutation and thus the matrix decomposition problem is dual degenerate (has multiple optima). Both dual degeneracy and the large number of permutable facets cause difficulties in our branch-and-cut algorithm and we will have to control the number of cuts generated and to handle stalling of the objective value.

The next two subsections list the results of our polyhedral investigations in the form of valid and facet defining inequalities. We distinguish between inequalities $a^T x \leq b$ that are *invariant under block permutations* or, equivalently, have the same coefficients $a_i^b = a_i^{b'}$ for all blocks $b \neq b'$ and row indices i, and *block-discernible inequalities* that do not have this property and distinguish different blocks. It will turn out that most of the block-discernible inequalities will be inherited from the stable set relaxation of the matrix decomposition problem, while the block-invariant constraints are related to an "aggregated" version of the problem. In both subsections we

want to assume $\kappa \geq 2$, because otherwise the matrix decomposition problem is a (special) set packing problem.

3.1 Block-Discernible Inequalities

We saw in Section 2 that we obtain a set packing relaxation of the matrix decomposition problem by dropping the block capacity constraints (ii) from the integer program (IP). The column intersection graph associated to the matrix $\text{IP}_{\text{(i),(iii)}}$ formed by the left-hand sides of the constraints (IP) (i) and (iii) has the set of possible row assignments $\{1, \ldots, m\} \times \{1, \ldots, \beta\}$ as its node set. A (conflict) edge exists between two assignments (i, b) and (j, b'), if rows i and j cannot be simultaneously assigned to the blocks b and b', i. e., either if $i = j$ and $b \neq b'$ or if $i \neq j$, $b \neq b'$, and rows i and j have a common nonzero entry in some column of A. We want to call this graph the *conflict graph* associated to the matrix decomposition problem (A, β, κ) and denote it by $G_c(A, \beta)$. In formulas: $G_c(A, \beta) = G(\text{IP}_{\text{(i),(iii)}})$. This graph allows us to interpret the inequality classes (i) and (iii) of (IP) as *clique inequalities* of the set packing relaxation corresponding to the matrix decomposition problem as also introduced by Padberg (1973) [23].

Theoremdd 3.3 (Clique). *Let $G_c(A, \beta) = (V, E)$ and $Q \subseteq V$. The inequality*

$$\sum_{(i,b)\in Q} x_i^b \leq 1$$

is valid for $P(A, \beta, \kappa)$ if and only if Q is a clique in $G_c(A, \beta)$. It is facet defining if and only if Q is a maximal clique in $G_c(A, \beta)$.

Proof. The validity part is obvious. It remains to show that it is facet defining if and only if Q is a maximal clique.

Suppose first that Q is not maximal but contained in a larger clique Q'. But then $\sum_{(i,b)\in Q} x_i^b \leq 1$ is the sum of the inequality $\sum_{(i,b)\in Q'} x_i^b \leq 1$ and the non-negativity constraints $x_i^b \geq 0$ for $(i, b) \in Q' \setminus Q$ and cannot be facet defining.

Assume now that Q is maximal. We will construct a set of $m\beta$ affinely independent block decompositions for which the inequality is tight. $|Q|$ such affinely independent vectors are the unit vectors e_i^b with $(i, b) \in Q$. For each other assignment $(j, b') \notin Q$ there exists some assignment (i, b) in Q that is not in conflict with (j, b'), since Q is a maximal clique. Thus, the vector $e_j^{b'} + e_i^b$ is the incidence vector of a feasible block decomposition for which the inequality is tight. (Note that we assumed $\kappa \geq 2$ at the beginning of this section for the case $b = b'$.) The resulting $m\beta - |Q|$ characteristic vectors

obtained in this way plus the $|Q|$ vectors constructed in the beginning are affinely independent. □

In the spirit of Theorem 3.3, (IP) (i) and (iii) are both clique inequalities and do not represent two different types of inequalities. The separation problem for clique inequalities is a maximum-weight clique problem and thus $\mathcal{NP}$-hard, see Garey & Johnson (1979) [10]. But some subclasses can be separated efficiently. One such class that we use in our implementation are the *two-partition inequalities*

$$\sum_{b\in B} x_i^b + \sum_{b'\notin B} x_j^{b'} \leq 1,$$

that are defined for all sets of blocks $B \subseteq \{1,\ldots,\beta\}$ and all pairs of non-disjoint rows i, j. Polynomial separation of this class is by inspection: Given i and j, we examine for each block b the variables x_i^b and x_j^b. If $x_i^b > x_j^b$, we add b to the set B, otherwise to its complement. Note that for the case of two blocks ($\beta = 2$), the two-partition inequalities are exactly the inequalities (IP) (i) and (iii) and, moreover, these are already all clique inequalities. In particular, separation of clique inequalities is polynomial for $\beta = 2$. In general, maximal cliques in $G_c(A,\beta)$ are of the form $\{(i_1,b_1),\ldots,(i_\beta,b_\beta)\}$, where the blocks $b_k, k = 1,\ldots,\beta$ are mutually different and the set of rows $\{i_1,\ldots,i_\beta\}$ forms a clique in $G(A^T)$. Thus all maximal cliques in $G_c(A,\beta)$ are of size β.

Another class inherited from the set packing relaxation are the *cycle inequalities*.

Theoremdd 3.4 (Odd Cycle). *If C is an odd cycle in $G_c(A,\beta)$, then the cycle inequality*

$$\sum_{(i,b)\in C} x_i^b \leq \lfloor |C|/2 \rfloor$$

is valid for $P(A,\beta,\kappa)$.

Analogously to the set packing case, see again Padberg (1973) [23], the odd cycle inequality is facet defining for its support if C is an odd hole (has no chords) and $|C|/2 \leq \kappa$. These conditions are, however, not necessary. Cycle inequalities can be separated in polynomial time using the algorithm of Lemma 9.1.11 in Grötschel, Lovász & Schrijver (1988) [12].

Along the same lines as for the clique and cycle inequalities, the matrix decomposition polytope clearly also inherits all other packing inequalities. But not only set packing, also set covering inequalities for (IP_c) can be applied (note that complementing variables preserves validity and dimension of the induced face), see Nobili & Sassano (1989) [22]. We do, however, not use any of them for our computations.

We close this section investigating the *block capacity constraints* (IP) (ii) which are not inherited from the set packing polytope or the set covering polytope.

Theoremdd 3.5 (Block Capacity). *The block capacity constraint*

$$\sum_{i=1}^{m} x_i^b \leq \kappa$$

is facet defining for $P(A, \beta, \kappa)$ if and only if $|\gamma(i)| \leq m - \kappa$ holds for every row i (where $\gamma(i)$ denotes all nodes adjacent to i in $G(A^T)$).

Proof. We first show that the inequality is facet defining if the above mentioned condition holds. To this purpose, let $a^T x \leq \alpha$ be a valid inequality that induces a facet such that $\{x \in P(A, \beta, \kappa) | \sum_{i=1}^{m} x_i^b = \kappa\} \subseteq \{x \in P(A, \beta, \kappa) | a^T x = \alpha\}$. We will show that the two inequalities are the same up to a positive scalar multiplicative factor.

Define x by

$$x_i^{b'} = \begin{cases} 1, & \text{if } 1 \leq i \leq \kappa,\ b' = b; \\ 0, & \text{else.} \end{cases}$$

x is a feasible block decomposition that assigns the first κ rows to block b. x satisfies the block capacity constraint with equality and thus $a^T x = \alpha$. Now observe that, for all $1 \leq i \leq \kappa < j \leq m$, the vector $x - e_i^b + e_j^b$ is also a feasible assignment that is tight for the block capacity inequality. It follows that $a_i^b = a_j^b$ for all $1 \leq i, j \leq m$.

Now consider assigning some row j to a block $b' \neq b$. By the assumption $|\gamma(j)| \leq m - \kappa$, there is a set $R(j)$ of κ rows not adjacent to j. But then $\sum_{i \in R(j)} e_i^b$ and $\sum_{i \in R(j)} e_i^b + e_j^{b'}$ are both feasible decompositions that satisfy the block capacity constraint with equality and thus $a_j^{b'} = 0$, completing the first part of the proof.

It remains to prove the converse direction. If there is some row j with $|\gamma(j)| > m - \kappa$, the inequality $\sum_{i=1}^{m} x_i^b + \sum_{b' \neq b} x_j^{b'} \leq \kappa$ is valid. But then the block capacity constraint can be obtained by summing up this inequality with $\sum_{b' \neq b} x_j^{b'} \geq 0$, and therefore it cannot be facet defining. □

3.2 Block-Invariant Inequalities

We investigate in this section inequalities for the matrix decomposition polytope that are invariant under block permutation. Consider for each block

decomposition x the "aggregated" vector

$$z(x) := \left(\sum_{b=1}^{\beta} x_1^b, \ldots, \sum_{b=1}^{\beta} x_m^b \right) \in \mathbb{R}^m.$$

$z(x)$ only records whether the matrix rows are assigned to some block or not, but no longer to which block. From a polyhedral point of view, the aggregated block decompositions give rise to an "aggregated" version of the block decomposition polytope

$$P_z(A, \beta, \kappa) := \text{conv}\{z \in \mathbb{R}^m : \text{ there is } x \in P(A, \beta, \kappa) \text{ with } z = z(x)\}.$$

The aggregated polytope is interesting because any valid inequality $\sum_{i=1}^{m} a_i z_i \leq \alpha$ for $P_z(A, \beta, \kappa)$ can be "expanded" into an inequality $\sum_{i=1}^{m} a_i \sum_{b=1}^{\beta} x_i^b \leq \alpha$ that is valid for $P(A, \beta, \kappa)$. All inequalities in this subsection are of this type. Obviously, the expansion process yields inequalities that are invariant under block permutations, hence the name.

From a computational point of view, block-invariant inequalities are promising cutting planes, because the objective of the matrix decomposition problem can be written in terms of aggregated z-variables as $\mathbf{1}^T x = \mathbf{1}^T z(x)$. Thus, a complete description of $P_z(A, \beta, \kappa)$ would already allow us to determine the correct objective function value of the matrix decomposition problem and z-cuts will help to raise the lower bound of an LP-relaxation.

The aggregated polytope $P_z(A, \beta, \kappa)$ provides a model of the matrix decomposition problem that rules out degeneracy due to block permutations. While this is a very desirable property of the aggregated z-formulation, its drawback is that it is already $\mathcal{NP}$-complete to decide whether a given vector $z \in \{0, 1\}^m$ is an aggregated block decomposition or not. (It can be shown that this is a bin-packing problem.) Our choice to use z-cuts within the x-model tries to circumvent this difficulty and combines the strengths of both formulations. We remark that degeneracy problems of this type arise also in block-indexed formulations of grouping problems in cellular manufacturing, where the difficulty can be resolved by means of alternative formulations, see Crama & Oosten (1996) [2].

We already know one example of an expanded aggregated constraint: Expanding the inequality $z_i \leq 1$ for the aggregated block decomposition polytope yields the block assignment constraint (IP) (i) $\sum_{b=1}^{\beta} x_i^b \leq 1$ that we have analyzed in the previous subsection. More inequalities are derived from the observation that adjacent rows (with respect to $G(A^T)$) can only be assigned to the same block. A first example of this sort of inequalities are the *z-cover inequalities*.

Theoremdd 3.6 (z-Cover). *Let $G(A^T) = (V, E)$ and let $W \subseteq V$ be a set of rows of cardinality $\kappa + 1$. Then, the z-cover inequality*

$$\sum_{i \in W} \sum_{b=1}^{\beta} x_i^b \leq \kappa$$

is valid for $P(A, \beta, \kappa)$ if and only if $(W, E(W))$ is connected (where $E(W)$ denotes all edges with both endpoints in W). It is facet defining for $P(A, \beta, \kappa)$ if and only if for each row $i \notin W$ the graph $(W \cup \{i\}, E(W \cup \{i\}))$ has an articulation point different from i.

Proof. The validity part is easy. Since $|W| = \kappa + 1$, not all rows can be assigned to the same block. If some rows of W are assigned to different blocks, there must be at least one row in W that is not assigned because $(W, E(W))$ is connected. Conversely, if W is not connected one easily finds a partition of W into two subsets that can be assigned to different blocks.

The proof that this inequality is facet defining if and only if for each row $i \notin W$ the graph $(W \cup \{i\}, E(W \cup \{i\}))$ has an articulation point different from i is analogous to the proof of Theorem 3.5. The condition guarantees that if row i is assigned to some block, the assignment can be extended in such a way that κ rows from W can be assigned to at least two blocks. On the other hand, if the condition is not satisfied for some $j \notin W$, the inequality $\sum_{i \in W \cup \{j\}} \sum_{b=1}^{\beta} x_i^b \leq \kappa$ is valid, and thus the z-cover inequality cannot be facet defining. □

In the set covering model, z-cover inequalities correspond to constraints of the form $\sum_{i \in W} \sum_{b=1}^{\beta} y_i^b \geq 1$ that have been used by Nicoloso & Nobili (1992) [21] for their computations. The separation problem is to find a tree of size $\kappa + 1$ of maximum node weight. This problem has been studied by Ehrgott (1992) [5] and was shown to be $\mathcal{NP}$-hard using a reduction to the node-weighted Steiner tree problem.

The z-cover inequalities are induced by trees, but it is possible to generalize them for subgraphs of higher connectivity.

Theoremdd 3.7 (Generalized z-Cover). *Let $G(A^T) = (V, E)$ and let $W \subseteq V$ be a set of rows of cardinality $\kappa + k$ with $k \geq 1$. Then, the (generalized) z-cover inequality*

$$\sum_{i \in W} \sum_{b=1}^{\beta} x_i^b \leq \kappa$$

is valid for $P(A, \beta, \kappa)$ if and only if $(W, E(W))$ is k-node connected. It is facet defining for $P(A, \beta, \kappa)$ if and only if for each row $i \notin W$ there exists some node cut N in $(W \cup \{i\}, E(W \cup \{i\}))$ of cardinality k with $i \notin N$.

The proof of this generalization follows exactly the lines of the proof of Theorem 3.6. In our branch-and-cut algorithm we restrict attention to the cases $k = 1$ and $k = 2$.

Closely related to the z-cover inequality is the *z-clique inequality*. Here, we consider some node set W that is not only k-node connected for some fixed k, but induces a complete subgraph. In this case the condition for being facet defining slightly changes.

Theoremdd 3.8 (z-Clique). *If Q is a clique in $G(A^T)$, then the z-clique inequality*

$$\sum_{i \in Q} \sum_{b=1}^{\beta} x_i^b \leq \kappa$$

is valid for $P(A, \beta, \kappa)$. It is facet-defining if and only if $|Q| \geq \kappa + 1$ and for each row $i \notin Q$ there exists a set of rows $R(i) \subseteq Q$, $|R(i)| = \kappa$, such that i is not adjacent in $G(A^T)$ to any node in $R(i)$.

Proof. The inequality is clearly valid. To show that it is facet defining given the mentioned conditions, let $a^T x \leq \alpha$ define a facet such that

$$\{x \in P(A, \beta, \kappa) | \sum_{i \in Q} \sum_{b=1}^{\beta} x_i^b = \kappa\} \subseteq \{x \in P(A, \beta, \kappa) | a^T x = \alpha\}.$$

We will show that the two inequalities are the same up to a positive scalar multiplicative factor. To this purpose, consider any κ rows of Q. The block decomposition obtained by assigning these rows to some block b is feasible and tight for the z-clique inequality. Since $|Q| \geq \kappa + 1$, we can use these solutions to show that $a_i^b = a_j^{b'}$ for all $i, j \in Q$ and for all blocks $b, b' \in \{1, \ldots, \beta\}$. Assuming that for each row $i \notin Q$ there exists a set of nodes $R(i) \subseteq Q$, $|R(i)| = \kappa$, that are not adjacent to i, we observe that for all $b' \neq b$, the vectors $\sum_{j \in R(i)} e_j^b$ and $\sum_{j \in R(i)} e_j^b + e_i^{b'}$ are valid block decompositions that satisfy the z-clique inequality with equality. It follows that $a_i^{b'} = 0$ for all $i \notin Q$, for all $b' \neq b$, and even for all blocks b', since b was arbitrary. This completes the first part of the proof.

If, on the other hand, Q has size less than or equal to κ, we obtain from (IP) (i) that the left hand side of the inequality is at most $|Q|$. Thus, the inequality is redundant and cannot define a facet. Suppose now the second condition is not satisfied, i. e., there is some $j \notin Q$ such that j is incident to at least $|Q| - \kappa + 1$ nodes in Q. This implies that $Q \cup \{j\}$ is at least $(|Q| - \kappa + 1)$-node connected. Theorem 3.7 states that $\sum_{i \in Q \cup \{j\}} \sum_{b=1}^{\beta} x_i^b \leq \kappa$ is valid and this implies that the z-clique inequality is redundant. □

The z-clique separation problem is again a max-clique problem and thus $\mathcal{NP}$-hard. In our implementation we check easily detectable special cases

like the following so-called *big-edge inequalities*

$$\sum_{i \in \text{supp}(A_{\cdot j})} \sum_{b=1}^{\beta} x_i^b \leq \kappa,$$

for all blocks b, where $A_{\cdot j}$ denotes the j-th column of A and $\text{supp}(A_{\cdot j})$ its nonzero row indices. These inequalities can be separated by inspection.

Another way to generalize the z-cover inequalities is by looking at node induced subgraphs that consist of several components. This idea, that gives rise to the class of *bin-packing inequalities*, came up in our computational experiments. Starting point is again a set of rows W that induces a subgraph of $G(A^T) = (V, E)$. Suppose $(W, E(W))$ consists of l connected components of sizes (in terms of nodes) $a_1, \ldots, a_l$. We can then associate a *bin-packing problem* with $(W, E(W))$, β, and κ in the following way: There are l items of sizes $a_1, \ldots, a_l$, and β bins of capacity κ each. The problem is to put all the items into the bins such that no bin holds items of a total size that exceeds the capacity κ. If this is not possible, we can derive a valid inequality for $P(A, \beta, \kappa)$.

Theoremdd 3.9. *Let $G(A^T) = (V, E)$ and $W \subseteq V$ be some subset of rows. If the bin packing problem associated to $(W, E(W))$, β, and κ has no solution, the bin-packing inequality*

$$\sum_{i \in W} \sum_{b=1}^{\beta} x_i^b \leq |W| - 1$$

is valid for $P(A, \beta, \kappa)$.

Proof. Consider some block decomposition x. If at least one row in W is not assigned to some block, the inequality is obviously satisfied. Otherwise all rows that belong to the same (connected) component of $(W, E(W))$ must be assigned to the same block. This yields a solution to the bin packing problem associated to $(W, E(W))$, β, and κ, a contradiction. □

We do not know any reasonable conditions that characterize when the bin packing inequalities are facet defining. Bin-packing separation is $\mathcal{NP}$-hard, see Garey & Johnson (1979) [10].

Next we give another class of *z-cycle inequalities* that generalize the cycle inequalities of the set packing polytope.

Theoremdd 3.10 (z-Cycle). *Let $G(A^T) = (V, E)$ and $C \subseteq V$ be a cycle in $G(A^T)$ of cardinality at least $\kappa + 1$. Then the z-cycle inequality*

$$\sum_{i \in C} \sum_{b=1}^{\beta} x_i^b \leq |C| - \left\lceil \frac{|C|}{\kappa + 1} \right\rceil$$

is valid for $P(A, \beta, \kappa)$.

The z-cycle inequality is valid because at least every $(\kappa+1)$-st node cannot be assigned to a block. One can also show that the inequality is facet defining for its support under certain rather restrictive conditions, for example, if C is an odd hole, $|C| \neq 0 \mod (\kappa+1)$, and the right-hand side is less than $\beta\kappa$. z-Cycle separation can be reduced to the TSP and is thus $\mathcal{NP}$-hard.

Our next class of inequalities comes up in several instances in our test set.

Theoremdd 3.11 (Composition of Cliques (COQ)). *Let* $G(A^T) = (V, E)$ *and consider* p *mutually disjoint cliques* $Q_1, \ldots, Q_p \subseteq V$ *of size* q *and* q *mutually disjoint cliques* $P_1, \ldots, P_q \subseteq V$ *of size* p *such that* $|P_i \cap Q_j| = 1$ *for all* i, j. *Let* $W = \cup_{i=1}^{p} Q_i$. *Then, the following inequality is valid for* $P(A, \beta, \kappa)$*:*

$$\sum_{i \in W} \sum_{b=1}^{\beta} x_i^b \leq \max_{\substack{\{r \in \mathbb{N}^\beta, s \in \mathbb{N}^\beta: \\ \sum r_b = p, \sum s_b = q\}}} \sum_{b=1}^{\beta} \min\{\kappa, r_b s_b\} =: \alpha(p, q, \beta, \kappa). \tag{2}$$

Proof. Consider a block decomposition x and let

$$\begin{aligned} \overline{r}_b &:= |\{j : \textstyle\sum_{i \in Q_j} x_i^b \geq 1, j \in \{1, \ldots, p\}\}|, \\ \overline{s}_b &:= |\{j : \textstyle\sum_{i \in P_j} x_i^b \geq 1, j \in \{1, \ldots, q\}\}|, \end{aligned}$$

for $b = 1, \ldots, \kappa$. Because Q_j and P_j are all cliques, we have that $\sum_{b=1}^{\beta} \overline{r}_b \leq p$ and $\sum_{b=1}^{\beta} \overline{s}_b \leq q$. Since $|P_i \cap Q_j| = 1$ for all i, j it follows that $\sum_{i \in W} x_i^b \leq \overline{r}_b \overline{s}_b$. Thus,

$$\begin{aligned} \sum_{i \in W} \sum_{b=1}^{\beta} x_i^b &= \sum_{b=1}^{\beta} \sum_{i \in W} x_i^b \\ &\leq \sum_{b=1}^{\beta} \min\{\kappa, \overline{r}_b \overline{s}_b\} \\ &\leq \max_{\substack{\{r \in \mathbb{N}^\beta, s \in \mathbb{N}^\beta: \\ \sum r_b \leq p, \sum s_b \leq q\}}} \sum_{b=1}^{\beta} \min\{\kappa, r_b s_b\} \\ &= \max_{\substack{\{r \in \mathbb{N}^\beta, s \in \mathbb{N}^\beta: \\ \sum r_b = p, \sum s_b = q\}}} \sum_{b=1}^{\beta} \min\{\kappa, r_b s_b\}, \end{aligned}$$

showing the statement. □

The right hand side of (2) is quite complicated, and we do not even know whether it can be computed in polynomial time. For $\beta = 2$ the right hand side looks more tractable:

$$\sum_{i \in W} \sum_{b=1}^{\beta} x_i^b \leq \max_{\substack{r=0,\dots,p \\ s=0,\dots,q}} \left(\min\{\kappa, rs\} + \min\{\kappa, (p-r)(q-s)\}\right). \tag{3}$$

But we do not know a closed formula in this case either. An interesting special case is $p = 2$. Here the graph $(W, E(W))$ consists of two disjoint cliques that are joint by a perfect matching. Suppose further $q < \kappa < 2q$. Then the right hand side of (3) reads

$$\begin{aligned} &\max\{0, \max_{s=0,\dots,q} (\min\{\kappa, s\} + \min\{\kappa, q-s\}), \min\{\kappa, 2q\}\} \\ =&\max\{0, q, \kappa\} = \kappa. \end{aligned}$$

In this case (2) turns out to be even facet defining if we require in addition that each node $i \notin W$ has at most $2q - \kappa$ neighbors in W, i, e., $|\gamma(i) \cap W| \leq 2q - \kappa$.

The development of our heuristic separation routine for COQ inequalities resulted in a slight generalization of this class. The support graphs of the left-hand sides of these *extended composition of clique inequalities* are COQs where some nodes have been deleted, the right-hand sides are left unchanged.

Theorem 3.12 (Extended Composition of Cliques (xCOQ)). *Let $G(A^T) = (V, E)$ and consider p mutually disjoint non-empty cliques $Q_1, \dots, Q_p \subseteq V$ of size at most q and q mutually disjoint non-empty cliques $P_1, \dots, P_q \subseteq V$ of size at most p such that*

(i) $|P_i \cap Q_j| \leq 1$ *for all* i, j *and*

(ii) $$\sum_{i=1}^{q} \sum_{j=1}^{p} |P_i \cap Q_j| = \sum_{i=1}^{q} |P_i| = \sum_{j=1}^{p} |Q_j|,$$

i.e., every element in one of the sets P_i appears in exactly one of the sets Q_j and vice versa. Let $W = \cup_{i=1}^{p} Q_i$. Then, the following inequality is valid for $P(A, \beta, \kappa)$:

$$\sum_{i \in W} \sum_{b=1}^{\beta} x_i^b \leq \alpha(p, q, \beta, \kappa). \tag{4}$$

Proof. The proof works by turning $P_1, \dots, P_q$ and $Q_1, \dots, Q_p$ into a proper COQ by adding some nodes that correspond to "artificial rows" and projecting the resulting inequality down to the original space of variables.

Let

$$\delta := \sum_{i=1}^{q} \sum_{j=1}^{p} (1 - |P_i \cap Q_j|)$$

be the number of nodes that "miss" to turn $P_1, \ldots, P_q$ and $Q_1, \ldots, Q_p$ into a COQ and add a row $\mathbf{1}^T$ of all ones to A for each of them to obtain a matrix $\overline{A}$ such that

$$\overline{A}_{i\cdot} = A_{i\cdot}, \quad i = 1, \ldots, m \qquad \text{and} \qquad \overline{A}_{i\cdot} = \mathbf{1}^T, \quad i = m+1, \ldots, m+\delta.$$

Consider the matrix decomposition problem $(\overline{A}, \beta, \kappa)$. Its row intersection graph $G(\overline{A}^T)$ contains $G(A^T)$ as a subgraph and the additional artificial nodes in $G(\overline{A}^T)$ are incident to every node of $G(\overline{A}^T)$ that corresponds to a row that is not all zero (except itself).

The sets $P_1, \ldots, P_q$ and $Q_1, \ldots, Q_p$ are again cliques in $G(\overline{A}^T)$. Associating each of the artificial nodes $i = m+1, \ldots, m+\delta$ to a different index pair ij such that $|P_i \cap Q_j| = 0$ and adding this node to both P_i and Q_j, we can extend $P_1, \ldots, P_q$ and $Q_1, \ldots, Q_p$ to a COQ $\overline{P}_1, \ldots, \overline{P}_q$ and $\overline{Q}_1, \ldots, \overline{Q}_p$ in $G(\overline{A}^T)$ with $\overline{W} := \cup_{j=1}^{p} \overline{Q}_j = W \cup \{m+1, \ldots, m+\delta\}$. Then, the COQ inequality

$$\sum_{i \in \overline{W}} \sum_{b=1}^{\beta} x_i^b \leq \alpha(p, q, \beta, \kappa) \tag{5}$$

is valid for $P(\overline{A}, \beta, \kappa)$ and, of course, also for

$$P(\overline{A}, \beta, \kappa) \cap \{x_i^b = 0 : \ i = m+1, \ldots, m+\delta, \ b = 1, \ldots, \beta\}.$$

Since the artificial variables in this polytope attain only values of zero, this remains true if one sets their coefficients in (5) also to zero. But as this results in the desired extended COQ inequality (4) and

$$P(\overline{A}, \beta, \kappa) \cap \{x_i^b = 0 : \ i = m+1, \ldots, m+\delta, \ b = 1, \ldots, \beta\} = P(A, \beta, \kappa) \times \{0\}^{\delta \times \beta},$$

the theorem follows by a projection on the space of the original variables. □

The last *star inequality* that we present in this section is special in the sense that it is the only one with non-0/1 coefficients. It was designed to deal with rows with many neighbors.

Theoremdd 3.13 (Star). *Let $G(A^T) = (V, E)$ and consider some row $i \in V$ with $|\gamma(i)| > \kappa$. Then the star inequality*

$$(|\gamma(i)| - \kappa + 1) \sum_{b=1}^{\beta} x_i^b + \sum_{j \in \gamma(i)} \sum_{b=1}^{\beta} x_j^b \leq |\gamma(i)|$$

is valid for $P(A, \beta, \kappa)$.

Proof. If i is assigned to some block b, then all rows in $\gamma(i)$ can only be assigned to b, but at most $\kappa - 1$ of them. The case where i is not assigned is trivial. □

The star inequality can be viewed as a lifting of the (redundant) inequality

$$\sum_{j \in \gamma(i)} \sum_{b=1}^{\beta} x_j^\beta \leq |\gamma(i)|$$

and we want to close this section with another simple lifting theorem for block-invariant inequalities with 0/1 coefficients.

Theoremdd 3.14 (Strengthening). *Let $G(A^T) = (V, E)$, W be a subset of V, and $\sum_{i \in W} \sum_{b=1}^{\beta} x_i^b \leq \alpha$ be a valid inequality for $P(A, \beta, \kappa)$. If for some row $j \notin W$ the condition*

$$|W \setminus \gamma(j)| + \kappa \leq \alpha$$

holds, then $\sum_{i \in W \cup \{j\}} \sum_{b=1}^{\beta} x_i^b \leq \alpha$ is also valid for $P(A, \beta, \kappa)$.

Proof. If j is assigned to some block b, the rows in $\{j\} \cup \gamma(j)$ can only be assigned to b, but at most κ of them. □

4 A Branch-And-Cut Algorithm

The polyhedral investigations of the last section form the basis for the implementation of a branch-and-cut algorithm for the solution of the matrix decomposition problem. This section describes the four main ingredients of this code: Separation and LP-management, heuristics, problem reduction, and searchtree management.

4.1 Separation and LP-Management

We use all of the inequalities described in Section 3 as cutting planes in our algorithm. It will turn out that some of them appear in large numbers. We thus opted for the following separation strategy: We try to identify many inequalities using fast heuristics, but add only selected ones to the LP. More expensive separation methods are used depending on their success.

We classify our separation routines according to their basic algorithmic principles: Inspection (enumeration), greedy heuristics, other heuristics, exact polynomial methods, and "hybrid" methods (combinations of exact and heuristic methods).

Input: $z(x) \in \mathbb{R}^m$, $G(A^T)$
Output: Node set T of a one- or two-connected subgraph of $G(A^T)$

function z-Cover Separation
  sort $z(x)$ such that $z(x)_{i_1} \geq z(x)_{i_2} \geq \cdots \geq z(x)_{i_m}$;
  for $k \leftarrow 1$ to m **do**
    $T \leftarrow \{i_k\}$;
    connectivity $\leftarrow 2$;
    for $j \leftarrow 1$ to m **do**
      if $j = k$ or $\gamma(i_j) \cap T = \emptyset$ **then** continue **end if**
      if $|T| = \kappa + 1$ and $|\gamma(i_j) \cap T| = 1$ **then** continue **end if**
      if $|T| \geq 2$ and $|\gamma(i_j) \cap T| = 1$ **then** connectivity $\leftarrow 1$ **end if**
      $T \leftarrow T \cup \{i_j\}$;
      if $|T| \geq \kappa +$ connectivity **then** break **end if**
    end for
  end for
  return T and connectivity;
end function

Algorithm 1: Separating z-Cover Inequalities With a Greedy Heuristic.

The most simple methods are used for *big-edge*, *two-partition*, and *star inequalities*: These classes can be separated by simple *inspection*, the details for two-partition inequalities were already described in Section 3.

Clique, *z-clique*, and *z-cover inequalities* are separated using *greedy heuristics*. In the last two cases, these methods start by sorting the rows of A with respect to increasing z-value, i.e., such that $z(x)_{i_1} \geq z(x)_{i_2} \geq \cdots \geq z(x)_{i_m}$. Then the greedy heuristic is called m times, once for each row i_j. In each call, i_j is used to initialize a tree/clique with respect to $G(A^T)$, that is iteratively extended greedily in the order of the z-sorting of the rows until z_{i_j} becomes zero and the growing procedure stops. There is also a second variant for z-cliques that is an adaptation of a similar routine by Hoffman & Padberg (1993) [16]. Here we call the greedy heuristic once for each column of A and initialize the clique with the support of this column. Having detected a violated clique inequality in one of these ways, we lift randomly determined additional rows with zero z-value sequentially into the inequality. This is done by applying the strengthening procedure of Theorem 3.14 which in this case amounts to a further growth of the clique by randomly determined rows of zero z-value. We tried to strengthen cover inequalities, but the computational effort was not justified by the one or two coefficients that were usually lifted. But, as was already mentioned in Section 3, we (heuristically) keep track of the connectivity of the growing graph. If the connectivity is 2 after the graph reached size $\kappa + 1$, we add another two-connected node if possible. Figure 1 gives more detailed pseudocode for

z-cover separation. Separation of clique inequalities is done using exactly the same routine as for z-cliques, but applied to $G_c(A, \beta)$ with node weights given by the x-variables.

z-Cycle inequalities are separated in the following heuristic way. We look at some path P with end-nodes u and v, where initially u and v coincide. In each iteration we extend the path at one of its end-nodes by a neighbor w with maximal $z(x)_w$-value. Let j_w be a column of A that connects w to the path P. Since j_w forms a clique in $G(A^T)$ there are additional nodes that can be potentially added to the path if the support of j_w is greater than two, i. e., $|\operatorname{supp}(A_{\cdot j_w})| > 2$. We store these additional nodes in a buffer which will be exploited later in the heuristic. Now we test whether the new path P extended by w can be closed to a cycle C that satisfies $|C| > \kappa$ and $|C| \neq 0 \bmod (\kappa + 1)$. This is done by looking for a column j of A that contains both end-nodes of P (one of them w). $\operatorname{supp}(A_{\cdot j})$ again forms a clique, and the additional nodes in this clique together with the nodes in the buffer give the flexibility to add further nodes to the cycle. This freedom is exploited in our routine. We try the procedure for several starting nodes $u = v$, whose number depends on the success of the heuristic.

Separation of the *composition of clique* inequalities is not easy: We do not even know a way to compute the right-hand side $\alpha(p, q, \beta, \kappa)$ in polynomial time! But there are problems in our test set, e.g., `pipex` (see Section 5.2), where compositions of cliques occur and there seems to be no way to solve this (small!) problem without them. Our heuristic was developed to capture these cases. It lead to the development of the more general class of extended COQ inequalities, which are easier to find. The idea is as follows.

Let us start with a composition of cliques $Q_1, \ldots, Q_p$ and $P_1, \ldots, P_q$ as stated in Theorem 3.11. Suppose that these cliques are contained in the columns $1, \ldots, p, p+1, \ldots, p+q$ of the matrix A, i.e., $\operatorname{supp}(A_{\cdot i}) \supseteq Q_i$, $i = 1, \ldots, p$, and $\operatorname{supp}(A_{\cdot i}) \supseteq P_i$, $i = p+1, \ldots, p+q$. Consider a *column/column-incidence matrix* S of A defined by

$$s_{ij} = \begin{cases} k, & \text{for } k \in \{l : \ a_{li} \neq 0 \neq a_{lj}\} \text{ arbitrary, but fixed;} \\ 0, & \text{if } A_{\cdot i}^T A_{\cdot j} = 0, \end{cases}$$

i.e., $s_{ij} = k \neq 0$ if and only if columns i and j intersect in some row k and in case there is no unique k we pick an arbitrary, but fixed one. Suppose for the moment that all entries in the submatrix $S_{\{1,\ldots,p\}\times\{p+1,\ldots,p+q\}}$ of S are mutually different, that is, there is no row index k that appears more than once. Then the composition of cliques corresponds to the rectangle submatrix $S_{\{1,\ldots,p\}\times\{p+1,\ldots,p+q\}}$ of S that is completely filled with nonzeros: The rows that appear on the left-hand side of the COQ inequality (2) are exactly those appearing in the matrix $S_{\{1,\ldots,p\}\times\{p+1,\ldots,p+q\}}$. In other words, the node

set W in (2) is $W = \{s_{ij} : \ i = 1, \dots, p,\ j = p+1, \dots, p+q\}$. Thus, given some vector $x \in \mathbb{R}^{m \times \beta}$, the left-hand side of (2) is $\sum_{i=1}^{p} \sum_{j=p+1}^{p+q} \sum_{b=1}^{\beta} x^b_{s_{ij}}$ and a final calculation of the right-hand side allows to check for a possible violation of the inequality.

Our heuristic tries to go the reverse direction: It identifies large filled rectangles in S and derives COQ and xCOQ inequalities from them. There are three difficulties. First, a clique in a composition can not only be a subset of a column of A, but any clique in $G(A^T)$. However, we have not incorporated this generality in our heuristic, because we do not know how to select a promising set of cliques in $G(A^T)$. Second, columns in A that form a composition of cliques may not appear in the right order: The rectangle identifies itself only after a suitable permutation of S. In this case, we have to reorder the columns and rows of S. We obtain a filled rectangle submatrix $S_{I \times J}$ of S by starting with each of column j of S once, extend this $1 \times |\operatorname{supp}(A_{\cdot j})|$ rectangle submatrix by columns that fit best in a greedy way, sort its rows lexicographically, and consider all maximal filled submatrices in the upper left corner as potential COQ-rectangles. A third serious problem arises when two columns of A intersect in more than one row. In this case the entries of the matrix S are no longer uniquely determined and it can happen that the entries of the rectangular submatrix $S_{I \times J}$ under consideration are no longer mutually different. Then $S_{I \times J}$ corresponds no longer to a composition of cliques and the inequality $\sum_{ij \in I \times J} \sum_{b=1}^{\beta} x^b_{s_{ij}} \leq \alpha\big(|I|, |J|, \beta, \kappa\big)$ is in general not valid. But one can set duplicate entries in S to zero until, for every row k, there is only one representative $s_{ij} = k$ left; denote the resulting matrix by S'. Then the sets

$$\overline{Q}_i := \{s'_{ij} : \ s'_{ij} \neq 0,\ j \in J\},\ i \in I \text{ and } \overline{P}_j := \{s'_{ij} : \ s'_{ij} \neq 0,\ i \in I\},\ j \in J$$

of non-zero entries in the rows and columns of S' form an extended composition of cliques and the corresponding xCOQ inequality

$$\sum_{k \in \operatorname{im} S_{I \times J}} \sum_{b=1}^{\beta} x^b_k \leq \alpha(|I|, |J|, \beta, \kappa)$$

is valid for $P(A, \beta, \kappa)$, where $\operatorname{im} S_{I \times J} = \{s_{ij} : \ ij \in I \times J\}$ denotes the set of row indices that appear in the submatrix $S_{I \times J}$. The interesting feature of separating extended COQ inequalities instead of COQs is that the generalization gives us the algorithmic freedom to handle multiple occurrences of rows in filled rectangles of S and this is the key to a successful heuristic separation of an otherwise rigid structure. The price for this, of course, is a reduced support in the left-hand side. To pay this price only when necessary, we heuristically determine a column/column-intersection matrix S with a large variety of rows in $\operatorname{im} S$. The right-hand side itself is computed

in amortized (pseudo-polynomial) time of $O(\beta\kappa n^2)$ steps by a dynamic program (for our tests $\beta \leq 4$ and $\kappa = O(n)$, and thus this effectively amounts to $O(n^3)$).

The reader might have noticed that several degrees of freedom in this separation routine can be used to search for rectangles with large z-value and this is what we would like to find. However, the running time of the method is too large to apply it after each LP and when we did, we did not find additional cuts. We thus call the routine only once, determine some promising COQs by combinatorial criteria, store them in memory, and separate them by inspection.

To separate *clique inequalities* (for $\beta > 2$), we use an *exact branch-and-bound algorithm* for the maximum weight clique problem. Although in principle exponential, this algorithm works fast for the separation problems coming up in our matrix decomposition instances because the maximum clique size is bounded by β. We have also tried to separate z-cliques exactly, but we never observed that additional cuts were found: In the small examples, the greedy heuristic is good enough, while in the larger ones with high capacities cliques of size κ don't seem to exist. Another exact, but this time polynomial, algorithm is used to separate *cycle inequalities*: We apply the odd-cycle algorithm described in Lemma 9.1.11 in Grötschel, Lovász & Schrijver (1988) [12].

Finally, a mixture of exact and heuristic ideas is used in a hybrid algorithm to separate the *bin-packing inequalities*. We start by determining a node set W that can result in a violated inequality. A necessary condition for this is

$$\sum_{i \in W} z(x)_i > |W| - 1 \iff 1 > \sum_{i \in W} (1 - z(x)_i)$$

and it is reasonable to construct W by iteratively adding rows that have a z-value close to one. We thus sort the nodes with respect to increasing z-value and add them to W in this order as long as the condition stated above is satisfied. This node set W induces a subgraph $(W, E(W))$ of $G(A^T)$ and we determine the components of this subgraph. The resulting bin-packing problem (see page 63) is solved using an exact dynamic programming algorithm (with a time bound).

In addition to these classical types of cutting planes we also use a number of *"tie-breaking" inequalities* to cut off decompositions that are identical up to block permutations or give rise to multiple optima for other reasons as a means to counter dual degeneracy and stalling. These inequalities are in general not valid for $P(A, \beta, \kappa)$, but for at least one optimal solution. The

most simple kind of these cuts are the *permutation inequalities*

$$\sum_{i=1}^{m} x_i^b \leq \sum_{i=1}^{m} x_i^{b+1}, \quad b = 1, \ldots, \beta - 1,$$

stating that blocks with higher indices are of larger size. To break further ties, we supplement them with inequalities stipulating that in case of equal sized blocks the row with the smallest index will be assigned to the block with smaller index. These *strengthened permutation inequalities* read

$$x_k^{b+1} + \sum_{i=1}^{m} x_i^b - \sum_{i=1}^{m} x_i^{b+1} \leq \sum_{i=0}^{k-1} x_i^b, \quad b = 1, \ldots, \beta - 1,\ k = 2, \ldots, m - 1.$$

If $\sum_{i=1}^{m} x_i^b - \sum_{i=1}^{m} x_i^{b+1} < 0$, the inequality is redundant, but in case of equality, the row with the smallest index in blocks b and $b+1$ must be in block b. The case $k = m$ is left out because it yields a redundant inequality. Both permutation and strengthened permutation inequalities can be separated by *inspection*.

Another idea that we use to eliminate multiple optima is based on the concept of *row preference*. We say that row i is *preferred* to row j or, in symbols, $i \prec j$ if

$$\gamma(i) \subseteq \gamma(j)$$

with respect to the row intersection graph $G(A^T)$. We may in this situation not know whether or not row i or j can be assigned to a block in some optimal solution, but we can say that for any decomposition x with $z(x)_j = 1$, say $x_j^b = 1$, either $z(x)_i = 1$ or we can get a feasible decomposition $x' = x - e_j^b + e_i^b$ with the same number of rows assigned. In this sense, row i is more attractive than row j. If we break ties on row preference by indices (i.e., $i \prec j \iff \gamma(i) \subsetneqq \gamma(j) \vee (\gamma(i) = \gamma(j) \wedge i < j)$), row preferences induce a partial order that we represent in a transitive and acyclic digraph

$$D(A) := (V, \{(i, j) : i \prec j\}).$$

Since the number of row preferences tends to be quadratic in the number of rows, we thin out this digraph by removing all transitive (or implied) preferences. The remaining row preferences are forced in our code by adding the *row preference inequalities*

$$\sum_{b=1}^{\beta} x_i^b \geq \sum_{b=1}^{\beta} x_j^b \quad \text{for } (i, j) \text{ with } i \prec j.$$

These can sometimes be strengthened to

$$x_i^b \geq x_j^b \quad \text{for all } b = 1, \ldots, \beta,$$

if we can be sure that rows i and j can not be assigned to different blocks in any decomposition. This will be the case, for example, if i and j are adjacent in $G(A^T) = (V, E)$ or if both i and j are adjacent to some third row k preferable to both of them (i.e., $i \prec j$, $k \prec i$, $k \prec j$, $ik \in E$ and $jk \in E$). Once $D(A)$ is set up, row preference inequalities can be separated by *inspection*.

Our last separation routine uses a *cut pool* that stores all inequalities found by the hitherto explained algorithms: The *pool separation* routine just checks all inequalities in the pool for possible violation.

The separation algorithms described in the previous paragraphs turned out to be very successful: Not only block-discernible (permutable) inequalities like two-partitions are found in large numbers, also block-invariant cuts like z-covers occur in abundance. Controlling the growth of the LP-relaxation is thus the main goal of our *separation and LP-maintenance strategy*. We start with a minimal LP-relaxation containing (besides the bounds) only the block assignment and block capacity constraints plus the $\beta - 1$ permutation inequalities. The cuts that are separated by inspection, i.e., big-edge inequalities, star inequalities, tie-breaking inequalities, and composition of clique inequalities are placed in a *cut pool*; they will be found by pool separation. The separation algorithms are called dynamically throughout the course of the branch-and-cut algorithm. After an LP is solved, we call the pool separation routine, followed by two-partition inequality separation and a couple of heuristics: The z-cover heuristic is called as it is, but application of the more expensive z-clique and z-cycle algorithms is controlled by a simple time- and success-evaluation. This control mechanism is motivated by the observation that our test set fell into two groups of examples, where one of these routines was either indispensable or essentially did not find a single cut. We empirically try to adapt to these situations by calling the separation routines only if their past success is proportional to the running time, or more precisely, if after the first call

$$\frac{\#\text{ of successful calls} + 1}{\#\text{ of calls}} > \frac{\text{time spent in routine}}{\text{total time spent in separation}}.$$

A call is counted as successful if a violated cut is found. If $\beta > 2$, there can be clique inequalities that are not two-partition constraints and in this case we next call the exact clique separation routine, that returns at most one cut. The branch-and-bound algorithm used there turned out to be fast enough to be called without any further considerations. Finally, we separate bin-packing inequalities. To avoid excessive running times due to the dynamic program, the routine is called with a time limit: The dynamic program will be stopped if the time spent in bin-packing separation exceeds the cumulated separation time of all other separation routines.

All violated cuts determined in this separation process are not *added* directly to the LP-relaxation, but stored in a *cut buffer* first. This buffer is saved to the pool, and then a couple of promising cuts are selected to strengthen the LP-relaxation. Our criteria here have an eye on the amount of violation and on the variety of the cuts. Since inequalities of one particular type tend to have similar support, we restrict the number of cuts per type and prefer to add inequalities of other types, even if they are not among the most violated. To accomplish this we add the

$$\frac{\#\text{ of cuts in cut buffer}}{\#\text{ number of types of cuts}}$$

most violated cuts of each type to the LP-relaxation. We also *delete* cuts from the LP-relaxation if they become non-binding by a slack of at least 10^{-3}, but keep them in the cut pool for a possible later pool separation.

Another feature of our code that aims for small LPs is to *locally setup* the LP-relaxation prior to computation at any node of the searchtree. This means that when branching on some node v we store at each of its sons a description of the last LP solved at v and of the optimal basis obtained. When we start to process v's sons, we set up this LP from scratch and load the associated (dual feasible) basis. In this way, we continue the computation exactly at the point where it stopped and the LP will be the result of a contiguous process independent of the node selection strategy. We have compared this approach to one where the start-LP at each newly selected node is just the last LP in memory and this leads to larger LPs and larger running times.

While these strategies were sufficient to keep the size of the LP-relaxation under control, explosive growth of the cut pool was a serious problem in our computations until we implemented the following *cut pool management*. We distinguish between disposable and indisposable cuts in the pool. *Indisposable cuts* are inequalities that are needed to set up the LP-relaxation at some node in the searchtree yet to be processed and all big-edge, star, and tie-breaking inequalities. All other cuts are *disposable* and can potentially be deleted from the cut pool, possibly having to be recomputed later. In order to control the pool size we restrict the number of disposable cuts in the pool by eliminating cuts that have not been in any LP for a certain number of iterations. This number depends on the size of the pool and the ratio of disposable to indisposable cuts.

The LPs themselves are solved with the `CPLEX 4.0` dual simplex algorithm using steepest edge pricing, see the CPLEX [17] documentation.

4.2 Heuristics

Raising the lower bound using cutting planes is one important aspect in a branch-and-cut algorithm, finding good feasible solutions early to enable fathoming of branches of the searchtree is another and we have implemented several primal heuristics for our matrix decomposition code. Since different nodes in a branch-and-bound tree correspond to different fixings of variables to zero or one, the heuristics should respect these fixings to increase the probability of finding different solutions. Applied at the root node where (at least initially) no variables are fixed, our methods can be seen as LP-based or pure combinatorial heuristics for the matrix decomposition problem.

Our heuristics fall into three groups: "Primal" methods that iteratively fix block assignments, "dual" methods that iteratively exclude assignments until decisions become mandatory due to lack of alternatives, and an improvement method that is applied as an "afterburner" to enhance the quality of the two groups of opening heuristics.

The *primal methods* consist of a *greedy algorithm* and a *bin-packing heuristic*, both are LP-based. The greedy algorithm starts by ordering the x_i^b variables; with probability $\frac{1}{2}$ a random ordering is chosen, otherwise a sorting according to increasing x-value is used. The rows are assigned greedily to the blocks in this order. This heuristic is similar in spirit to the popular "LP-plunging" method, i.e., the iterative rounding of some fractional LP-value to an integer followed by an LP-reoptimization, but much faster. We have also tried LP-plunging, but for the matrix decomposition problem the results were not better than with the simple greedy method, while the running time was much larger. The bin-packing heuristic starts by determining a set of nodes W that will be assigned to the blocks and used to set up a corresponding bin-packing problem. In order to find a better decomposition than the currently best known with, say, $z^\star$ rows assigned, W should be of cardinality at least $z^\star + 1$ and therefore we take the $z^\star + 1$ rows with the largest $z(x)$-values to be the members of W. The corresponding bin-packing problem is set up and solved with the same dynamic program that we used for the separation of the bin-packing inequalities; it is also called with a time limit, namely 10 times as much as all other primal heuristics (that are very fast) together. Clearly, we also watch out for better solutions that might be detected in bin-packing separation.

The *dual methods* also respect variable fixings, but are not LP-based. The idea behind them is not to assign rows to blocks, but to iteratively eliminate assignments of "bad" rows. Suppose that a decision was made to assign certain rows (assigned rows) to certain blocks, to exclude other rows from assignment (unassigned rows), while for the remaining rows a decision has yet to be made (free rows). Removing the unassigned nodes from the row

intersection graph $G(A^T)$ leaves us with a number of connected components, some of them larger than the maximum block capacity κ, some smaller. Both variants of the dual method will break up the components that are larger than the block capacity κ by unassigning free rows until no more such components exist. At this point, a simple first-fit decreasing heuristic is called to solve the corresponding bin-packing problem. The two variants differ in the choice of the next bad row to remove. Variant I chooses the free row in some component of size larger than κ with the largest degree with respect to the row intersection graph $G(A^T)$, variant II excludes assignment of a free row of some component with size larger than κ with the largest indegree with respect to $D(A)$, or, in other words, the least preferable row. We have also tried to use a dynamic program to solve the bin-packing problems, but it did not provide better results in our tests.

Our *improvement heuristic* is a variation of a local search technique presented by Fiduccia & Mattheyses (1982) [8]. Given some block decomposition, it performs a sequence of local exchange steps each of the following type. Some assigned row is chosen to be made unassigned opening up possibilities to assign its unassigned neighbors. These assignments are checked and feasible assignments are executed. The details are as follows. The heuristic performs a number of passes (10 in our implementation). At the beginning of each pass, all rows are eligible for unassignment in the basic exchange step. Each row may be selected only once for unassignment in each pass and will then be "locked". Candidates for becoming unassigned are all currently assigned and unlocked rows. These candidates are rated according to the number of possible new assignments (computed heuristically) and we choose the one that is best with respect to this rating. As a special annealing-like feature, the algorithm will also perform the exchange step if it leads to a change of the current solution to the worse. If no exchange step is possible because all assigned rows are already locked, the pass ends and the next pass is started.

The *strategy to call the heuristics* is as follows. The primal methods are called after each individual LP, whereas the dual heuristics are called only once at each node in the branch-and-bound tree, because they behave in a different way only due to changes in the variable fixings.

4.3 Problem Reduction

We use a couple of problem reduction techniques to eliminate redundant data. First we apply an *initial preprocessing* to the matrix A before the branch-and-cut algorithm is initiated. The purpose of this preprocessing step is to eliminate columns from the matrix A without changing the row intersection graph. We first perform a couple of straightforward tests to iden-

tify columns that are contained in other columns and can thus be deleted: We remove empty columns, then unit columns, duplicate columns, and finally by enumeration columns that are contained in others.

These simple initial preprocessing steps are amazingly effective as we will see in the section on computational results. In principle, the number of rows can be reduced also. For example, empty rows could be eliminated and later used to fill excess capacity in any block, duplicate rows or rows with just one nonzero entry could be eliminated by increasing the capacity requirements of one of its adjacent rows. These reductions, however, lead to changes in the IP model and affect all separation routines discussed so far so that we refrained from implementing them.

In addition to this initial preprocessing we do *local fixings* at the individual nodes of the branch-and-bound searchtree after each call to the LP-solver. Apart from *reduced cost fixing* and *fixing by logical implication* (i.e., if x_i^b is fixed to one, $x_i^{b'}$ will be fixed to zero for all blocks $b' \neq b$), we try to identify rows that cannot be assigned to any block given the current state of fixings. To this purpose we look at all rows W that are currently fixed for assignment to some block. We then check for each unassigned row i whether the subgraph $(W \cup \{i\}, E(W \cup \{i\}))$ of $G(A^T) = (V, E)$ contains a component with more than κ rows. If so, row i can be fixed to be unassigned.

4.4 Searchtree Management

Despite our efforts to understand the polyhedral combinatorics of the matrix decomposition problem, we do not have a strong grip on the corresponding polytope and after an initial phase of rapid growth of the lower bound, stalling occurs in the presence of significant duality gaps. We believe that — up to a certain point— it is favorable in this situation to resort to branching early, even if there is still slow progress in the cutting plane loop. In fact, we apply a rather "aggressive" *branching strategy*, splitting the currently processed node if the duality gap could not be reduced by at least 10% in any 4 consecutive LPs. On the other hand, we *pause* a node (put it back into the list of nodes yet to be processed) if the local lower bound exceeds the global lower bound by at least 10%.

Branching itself is guided by the fractional LP-values. We first look for a most fractional $z(x)$-value. If, e.g., $z(x)_i$ is closest to 0.5 (breaking ties arbitrarily), we create $\beta + 1$ new nodes corresponding to the variable fixings

$$x_i^1 = 1, x_i^2 = 1, \ldots, x_i^\beta = 1, \quad \text{and} \quad \sum_{b=1}^{\beta} x_i^b = 0.$$

In other words, we branch on the block assignment constraint corresponding to row i. The advantage of this scheme is that it leads to only $\beta + 1$ new nodes instead of 2^β-nodes in an equivalent binary searchtree. If all $z(x)$-values are integral, we identify a row with a most fractional x-variable and perform the same branching step. We have also tried other branching rules by taking, for instance, the degree of a row in $G(A^T)$ into account, but the performance was inferior to the current scheme.

5 Computational Results

In this section we report on computational experiences with our branch-and-cut algorithm for the solution of matrix decomposition problems arising in linear and integer programming problems. Our aim is to find answers to two complexes of *questions*. First, we would like to evaluate our branch-and-cut approach: What are the limits in terms of the size of the matrices that we can solve with our algorithm? What is the quality of the cuts, do they provide a reasonable solution guarantee? Second, we want to discuss our concept of decomposition into bordered block diagonal form: Do the test instances have this structure or are most integer programming matrices not decomposable in this way? Does knowledge of the decomposition help solving them? And do our heuristics provide reasonable decompositions that could be used within a parallel LU-factorization framework or an integer programming code?

Our *test set* consists of matrices from real-world linear programs from the `Netlib` and integer programming matrices from the `Miplib`. In addition, we consider two sets of problems with "extreme" characteristics as benchmark problems: Some small matrices arising from Steiner tree packing problems, see Grötschel, Martin & Weismantel (1996) [13], and equipartition problems introduced in Nicoloso & Nobili (1992) [21]. The Steiner tree problems are known to be in bordered block diagonal form and we wanted to see whether our code is able to discover this structure. The equipartition problems, on the other hand, are randomly generated. Our complete test data is available via anonymous `ftp` from `ftp://ftp.zib.de` at `/pub/Packages/mp-testdata/madlib` and via webbrowser at URL `ftp://ftp.zib.de/pub/Packages/mp-testdata/madlib/index.html`.

In the following subsections, we report the *results of our computations* on the different sets of problems. Our algorithm is implemented in `C` and consists of about 36,000 lines of code. The test runs were performed on a Sun Ultra Sparc 1 Model 170E and we used a time limit of $1,800$ CPU seconds. The format of the upcoming tables is as follows: Column 1 provides the name of the problem, Columns 2 to 4 contain the number of rows, columns and

nonzeros of the matrix to be decomposed. The two succeeding columns give the number of columns and nonzeros after presolve. Comparing Column 3 with 5 and 4 with 6 shows the performance of our preprocessing. The succeeding 5 columns give statistics about the number of cuts generated by our code. There are, from left to right, the number of initial cuts (*Init*) including block assignment, block capacity, big-edge, star, and tie-breaking inequalities (but not composition of clique inequalities, although they are also separated from the pool), the number of z-cover (*Cov*), the number of two-partition (*2part*), the sum of the number of bin-packing, cycle, z-cycle, clique, z-clique, and composition of clique inequalities (*BCC*), and finally the number of violated inequalities separated from the pool (*pool*). The following two columns (Columns 12 and 13) show the number of branch-and-bound nodes (*Nod*) and the number of LPs (*Iter*) solved by the algorithm. The second part of the tables starts again with the name of the problem. The next eight columns give solution values. We do not report the number of assigned rows, but the number of rows in the border, because it is easier to see whether the matrix could be decomposed into block diagonal form (in this case the value is zero) or close to this form (then the value is a small positive integer). *Lb* gives the global lower bound provided by the algorithm. It coincides with the value of the upper bound *Ub* (next column) when the problem is solved to *proven optimality*. Skipping two columns for a moment, the next four columns refer to the heuristics. *G*, *D1*, *D2* and *B* stand for the *greedy*, the *dual (variant I and II),* and the *bin-packing* heuristic. The corresponding columns show the best solutions obtained by these heuristics throughout the computations at the root node. If this value coincides with the number of rows of the matrix, all rows are in the border and the heuristic failed. The two (skipped) columns right after *Ub* show which heuristic *He* found the best solution after *No* many branch-and-bound nodes (1 means it was found in the root node, 0 means that preprocessing solved the problem). The additional letter *I* indicates that the value was obtained by a succeeding call to the *improvement heuristic*. *BS* means that the bin-packing separation routine found the best solution, an asterisk $*$ shows that the LP solution provided an optimal block decomposition. The remaining five columns show timings. The last of these columns *Tot* gives the total running time measured in CPU seconds. The first four columns show the percentage of the total time spent in cut-management (*Cm*), i.e., local setup, LP-, cut-buffer, and pool-management, the time to solve the linear programs (*LP*), the time of the separation algorithms (*Sep*), and the time for the heuristics (*Heu*).

5.1 The Netlib Problems

The first test set that we are going to consider consists of matrices that arise from *linear programming problems* taken from the Netlib[1]. We investigated whether *basis matrices* corresponding to optimal solutions of these linear programs can be decomposed into (bordered) block diagonal form. These bases were taken from the dual simplex algorithm of CPLEX [17]. Analyzing the decomposibility of such matrices gives insight into the potential usefulness of parallel LU-factorization methods within a simplex-type solver. In this context β reflects the number of processors that are available. We have chosen $\beta = 4$, since this is somehow the first interesting case where parallelization might pay. As a heuristic means for good load balancing we aim at equal-sized blocks and have set the capacity to $\kappa := \frac{\#rows}{4}$ rounded up. We tested all instances with up to 1,000 rows.

Table 1 shows the *results* of these experiments. The problems up to 100 rows are easy. The range of 100–200 rows is where the limits of our code become visible and this is the most interesting "hot area" of our table: The problems here are already difficult, but because of the combinatorial structure and not because of sheer size. The results for the problems with more than 200 rows are of limited significance, because these matrix decomposition problems are large-scale and the algorithm solves too few LPs within the given time limit. We can only solve a couple of readily decomposable large instances, but it is worth noticing that a significant number of such instances exists: The difficulty of matrix decomposition problems depends as much on the structure of the matrix as on the number of rows, columns, or nonzeros.

Let us first investigate the "*dual side*" of the results. We observe that we solve very few problems at the root node (only 9 out of 77), and that the number of cuts is very large, in particular in the hot area of the table. The reason for this is basically the symmetry of the problem, as can be seen from the pool separation column (*Pool*) that counts, in particular, all violated tie-breaking cuts. Unfortunately, we don't see a way to get around this phenomenon, but we believe that the symmetry mainly prevents us from solving difficult instances of larger size. The quality of the cuts is in our opinion reasonable, as can be seen from the size of the branch-and-bound tree and the number of LPs solved. It is true, however, that the lower bound improves fast at first while stalling occurs in later stages of the computation although still large numbers of cuts are found and the problem is finished by branch-and-bound. The same behaviour has been reported for similar problems like the node capacitated graph partitioning problem discussed in Ferreira et al. (1998) [7].

[1] Available by anonymous `ftp` from `ftp://netlib2.cs.utk.edu/lp/data`.

Table 1: Decomposing Linear Programming Basis Matrices.

	Original			Presolved		Cuts					B&B		Best Solutions				Heuristics at Root				Time				
Name	rows	col	nz	col	nz	Init	Cov	2part	BCC	Pool	Nod	Iter	Lb	Ub	He	No	G	D1	D2	B	Cm	LP	Sep	Heu	Tot
seba	2	2	2	0	0	0	0	0	0	0	1	0	0	**0**	-	0	-	-	-	-	0%	0%	0%	0%	0.0
afiro	20	20	34	10	24	106	12	6	5	2	1	2	2	**2**	B	1	3	3	3	2	0%	40%	10%	0%	0.1
fit1d	24	24	178	9	126	298	97	38	265	105	6	18	16	**16**	B	1	16	17	17	16	6%	50%	25%	6%	0.7
fit2d	25	25	264	11	147	379	549	458	1504	892	41	99	18	**18**	D1	1	20	18	18	25	12%	54%	22%	5%	5.5
sc50b	28	28	84	27	82	124	2304	1291	152	5452	56	182	11	**11**	IG	1	11	11	11	28	6%	71%	12%	4%	12.7
sc50a	29	29	88	24	77	137	1062	654	53	1487	26	79	8	**8**	ID1	1	8	8	8	29	3%	78%	11%	2%	6.8
kb2	39	39	213	16	164	328	38	21	2	8	1	3	14	**14**	IG	1	14	14	14	39	4%	69%	14%	2%	0.5
vtpbase	51	51	198	27	149	415	9130	4796	407	32630	561	1025	14	**14**	IG	1	14	17	17	15	7%	57%	20%	8%	94.6
bore3d	52	52	311	29	262	579	2241	1025	2333	2706	51	116	23	**23**	D1	9	24	24	24	52	7%	69%	17%	2%	24.0
adlittle	53	53	203	35	167	454	818	244	20	515	11	31	10	**10**	D1	1	10	10	10	53	6%	72%	15%	2%	5.0
blend	54	54	313	28	274	547	3357	1553	2757	6143	61	166	20	**20**	IG	2	21	25	25	21	6%	66%	20%	3%	35.9
recipe	55	55	100	24	69	317	0	14	3	2	1	3	1	**1**	IB	1	3	4	4	1	0%	51%	18%	10%	0.4
scagr7	58	58	242	34	210	439	770	331	718	468	16	30	21	**21**	B	4	25	25	25	58	6%	46%	32%	10%	5.2
sc105	59	59	220	48	203	318	6200	3415	151	19447	66	207	16	**16**	B	20	17	17	17	59	4%	82%	9%	2%	85.1
stocfor1	62	62	180	25	143	497	2395	801	55	2977	21	65	10	**10**	B	7	12	12	12	62	6%	68%	17%	4%	15.2
scsd1	77	77	215	70	207	500	4264	1570	60	6331	21	74	8	**8**	IB	5	10	10	10	77	3%	81%	10%	3%	53.1
beaconfd	90	90	618	48	576	1626	4618	1403	3231	6296	121	263	26	**26**	D1	21	28	28	28	90	4%	67%	21%	4%	115.7
share2b	93	93	482	37	340	825	7335	4636	212	14713	626	860	9	**9**	ID1	5	10	10	10	93	4%	77%	8%	6%	282.5
share1b	102	102	485	56	364	721	3769	2135	294	8204	806	1142	7	**7**	B	4	8	8	8	102	5%	65%	13%	9%	203.6
forplan	104	104	575	70	535	815	43011	16754	4783	145556	316	681	31	36	ID1	18	39	44	44	104	2%	85%	8%	3%	1801.1
scorpion	105	105	383	46	300	812	48144	12649	1178	113261	1206	2485	11	**11**	B	7	14	14	14	105	4%	67%	14%	9%	940.7
sc205	113	113	691	104	676	561	24727	15842	272	116813	136	299	28	39	IG	9	45	45	45	113	1%	92%	4%	1%	1806.6
brandy	113	113	874	83	774	1270	33012	13041	5037	107955	271	664	34	**34**	*	214	51	51	51	113	2%	82%	11%	2%	1200.7
lotfi	122	122	349	60	267	1026	85763	17243	1110	203255	1076	2217	19	**19**	B	279	23	23	23	122	4%	87%	12%	4%	1643.3
boeing2	122	122	435	70	383	1038	18823	5794	200	41101	71	237	19	**19**	B	33	31	32	32	122	2%	84%	7%	3%	411.9
tuff	137	137	820	84	748	1795	17387	4823	200	32293	116	306	26	**26**	B	43	28	35	35	137	2%	85%	7%	3%	635.3
grow7	140	140	1660	51	243	948	33529	12866	320	182151	206	424	10	18	IG	9	22	22	22	140	3%	80%	9%	6%	1801.2
scsd6	147	147	383	138	373	804	41055	12440	297	116468	111	331	10	22	IG	4	23	23	23	147	1%	87%	5%	4%	1805.1
e226	148	148	954	88	807	1469	33011	14063	272	112223	206	355	22	30	IG	15	44	44	44	148	1%	84%	7%	5%	1800.5
israel	163	163	1317	37	1014	2217	537	289	2115	237	11	25	98	**98**	IG	2	101	119	118	163	4%	32%	51%	10%	42.1
agg	164	164	669	52	551	1492	37378	12257	325	174665	341	539	19	24	IG	56	43	43	43	164	3%	81%	9%	5%	1802.3
capri	166	166	826	102	718	1625	9291	4015	396	9008	71	94	23	53	D1	7	59	64	64	166	0%	90%	4%	3%	1801.8
wood1p	171	171	2393	55	1340	1650	6253	2457	56	7758	46	94	28	**28**	B	29	32	32	32	171	2%	73%	15%	7%	277.6
bandm	180	180	1064	90	815	1690	11854	6042	405	31838	66	96	19	43	IG	11	58	58	58	180	1%	93%	4%	1%	1835.0
scrs8	181	181	887	110	675	1400	11399	5375	72	30637	56	85	17	43	IG	9	56	56	56	181	0%	92%	3%	2%	1802.1
ship04s	213	213	573	176	536	1530	1084	72	7	190	1	7	4	**4**	IB	1	8	8	8	4	5%	55%	31%	2%	10.8
scagr25	221	221	1627	91	1470	1831	14509	8239	6884	23224	121	261	62	69	B	53	81	81	81	221	1%	52%	21%	23%	1814.7
scfxm1	242	242	1064	154	922	2182	16607	3887	68	38232	31	80	10	30	IG	7	50	50	50	242	1%	87%	5%	5%	1803.7
stair	246	246	3402	216	3154	1552	7192	3058	33	1005	51	44	58	122	IG	3	123	133	133	246	0%	83%	10%	5%	1914.6
shell	252	252	493	235	476	1495	14086	1311	85	5878	46	101	4	**4**	B	17	8	8	8	252	3%	58%	16%	18%	187.8
standata	258	258	513	109	364	1733	0	0	0	0	1	1	1	**1**	D1	1	17	1	1	258	0%	52%	35%	3%	5.3
sctap1	269	269	640	77	373	1957	25028	4187	110	37924	61	143	9	19	D1	13	20	20	20	269	1%	86%	6%	5%	1801.2
agg2	280	280	1468	109	1275	2270	9856	4124	34	13961	31	45	13	78	IG	5	110	112	112	280	0%	89%	5%	3%	1810.8
agg3	282	282	1444	99	1162	2259	9572	4674	32	13575	26	43	12	79	IG	6	92	116	116	282	0%	91%	4%	2%	1808.6
ship08s	284	284	699	201	616	1909	7352	480	33	3403	11	35	4	**4**	B	3	10	10	10	284	3%	76%	14%	4%	144.0
boeing1	284	284	1384	174	1271	2670	13291	4198	48	22032	31	56	12	55	IG	2	56	56	56	284	1%	91%	3%	4%	1812.6
grow15	300	300	3680	102	489	2044	17134	5225	72	28665	21	77	7	20	ID1	1	20	20	20	300	1%	84%	5%	8%	1817.1
fffff800	306	306	1382	182	1237	3045	9590	3320	30	15509	36	51	12	52	IG	6	79	104	104	306	0%	89%	5%	3%	1820.2
etamacro	307	307	1005	215	907	1824	8431	3985	33	10512	16	36	6	83	IG	3	91	91	91	307	0%	93%	2%	2%	1832.6
ship04l	313	313	868	274	829	2321	4387	520	125	1918	356	491	5	**5**	B	14	8	8	8	313	1%	66%	3%	25%	1727.2
gfrdpnc	322	322	623	276	576	1569	13308	2328	53	4899	31	59	4	**4**	BS	7	8	8	8	322	2%	65%	9%	21%	398.4
ship12s	344	344	858	247	761	2313	1676	108	7	62	6	8	3	**3**	IB	2	10	10	10	344	4%	61%	24%	6%	35.4
finnis	350	350	831	178	653	2371	13570	3020	55	17997	26	61	7	27	B	4	31	31	31	350	1%	90%	4%	3%	1842.8
pilot4	352	352	3157	265	2988	3014	8563	4104	25	13824	16	29	6	106	IG	1	106	109	109	352	0%	87%	6%	4%	1881.6
standmps	360	360	836	217	691	2785	32633	4731	183	37695	71	210	7	10	D1	4	11	11	11	360	2%	79%	9%	7%	1802.0
degen2	382	382	2440	262	2230	2717	6922	1959	16	2396	26	25	12	114	ID1	1	114	114	114	382	0%	92%	3%	2%	1922.1
scsd8	397	397	1113	394	1109	1755	9128	4799	24	14671	6	25	4	63	IG	2	85	98	89	397	1%	90%	2%	5%	1912.1
grow22	440	440	5272	161	732	2984	10976	1880	26	10062	11	28	4	24	ID1	3	25	31	25	440	1%	83%	4%	11%	1808.9
bnl1	448	448	1656	307	1481	3788	6984	1623	16	6521	11	19	6	68	D1	1	68	68	68	448	0%	93%	2%	3%	2046.8
czprob	475	475	939	464	928	3640	1227	25	4	9	1	4	3	3	D1	1	3	3	3	475	4%	53%	35%	1%	42.0
scfxm2	485	485	2179	313	1906	4317	7582	1712	17	10240	6	18	5	65	IG	2	113	120	115	485	1%	80%	4%	13%	1849.6
perold	500	500	3277	424	3077	3448	5983	3669	13	6810	6	14	5	166	D1	1	166	166	166	500	0%	90%	3%	4%	2079.5
ship08l	520	520	1404	436	1320	3767	11855	703	76	4080	71	129	4	5	D1	9	12	12	12	520	1%	48%	7%	40%	1807.1
maros	545	545	2637	301	2323	4384	7750	2718	15	2539	11	18	8	103	D1	1	103	103	103	545	1%	85%	4%	8%	1832.8
ganges	576	576	3002	466	2892	4098	19490	2429	40	10251	16	42	6	15	IB	3	36	39	39	576	2%	65%	10%	21%	1805.4
pilotwe	613	613	2982	561	2833	3804	5489	4184	10	4238	6	11	3	181	IG	2	182	186	182	613	0%	83%	3%	11%	1911.8
nesm	622	622	1925	419	1461	3720	5929	3407	11	4785	6	12	3	151	D1	2	152	152	152	622	1%	86%	4%	7%	1803.8
fit1p	627	627	4992	1	627	5013	0	0	0	0	1	1	470	**470**	IG	1	470	470	470	627	0%	92%	5%	0%	726.4
25fv47	677	677	3750	442	3278	5699	4644	1315	7	482	6	9	4	190	IG	1	190	195	195	677	0%	90%	3%	5%	2102.4
ship12l	686	686	1883	589	1786	5026	3212	127	6	74	1	6	3	**3**	B	1	4	18	18	3	3%	57%	27%	8%	169.8
woodw	711	711	3044	528	2849	4517	3736	2395	6	399	11	9	5	184	IG	2	186	187	187	711	0%	89%	4%	5%	2069.4
scfxm3	728	728	3285	469	2876	6411	4969	1203	8	3541	6	9	3	104	D1	2	105	105	105	728	0%	90%	3%	4%	1924.6
pilotja	745	745	4738	548	4121	5708	4462	2129	7	670	6	8	3	226	ID1	1	226	226	226	745	0%	88%	4%	6%	1989.3
pilotnov	783	783	4428	498	3606	5479	3859	2466	6	1159	6	7	3	250	D1	1	250	250	250	783	0%	85%	3%	9%	1855.4
bnl2	940	940	3284	489	2509	6981	4930	880	7	6334	6	8	3	199	D1	1	199	199	199	940	0%	81%	4%	11%	2051.2
sctap2	977	977	1491	161	642	5506	1256	0	0	156	6	6	1	**1**	D1	2	2	4	4	2	0%	8%	16%	73%	834.5
truss	1000	1000	3564	986	3540	4792	4999	4137	6	1810	6	7	3	286	ID1	2	288	288	288	1000	0%	70%	4%	23%	2347.0
$\sum$	22911	22911	108546	14614	82679	169450	867384	285672	37498	1909629	8022	15550	1455	4423		1026	4841	4987	4962	20893	1%	82%	6%	8%	85517.1

Investigating the "*primal side*", we see that the greedy heuristic seems to be most reliable. The two dual methods perform exactly the same and yield solutions of the same quality as the greedy. Bin-packing is either very good (a rather rare event) or a catastrophe, but complements the other heuristics. If we look at the quality of the solutions found at the root node as a measure of the method as a stand-alone decomposition heuristic, the table shows pretty good results for the small problems. For the larger instances the situation is a bit different. We detect larger gaps, see, for instance, `scfxm1` or `ship12s`. In fact, we have often observed in longer runs on larger examples that the best solution could steadily be improved and the optimal solution was found late. A reason might be that the heuristics are closely linked to the LP-fixings and essentially always find the same solutions until the branching process forces them strongly into another direction. We believe (and for some we know) that many of the larger problems can be decomposed much better than the *Ub*-column indicates and that there might be potential to further improve stand-alone primal heuristics.

The answer to the final question whether LP basis matrices are decomposable into four blocks is 'Yes' and 'No'. Some like `recipe` or `standata` are (only one row is in the border), others are not, for example `israel`: 98 out of 163 are in the border. The results leave, however, the possibility that larger LP-matrices, that are generally sparser than small ones, can be decomposed better so that we can not give a final answer to this question.

5.2 The `Miplib` Problems

In this test series we examine whether matrices arising from *integer programs* can be decomposed into (bordered) block diagonal form. There are two applications here.

First, decomposing the original constraint matrix A of some general integer program can be useful to *tighten* its LP-relaxations within a branch-and-cut algorithm. The structure of the decomposed matrix is that of a multiple knapsack or general assignment problem, and inequalities known for the associated polytopes (see Gottlieb & Rao (1990) [11], Ferreira, Martin & Weismantel (1996) [6]) are valid for the MIP under consideration. The first interesting case in this context are two blocks and we set $\beta := 2$. We used $\kappa := \frac{(\#rows)\cdot 1.05}{2}$ rounded up as the block capacity, which allows a deviation of 10% of the actual block sizes in the decomposition.

Table 2 shows the *results* that we obtained for matrices of mixed integer programs taken from the `Miplib`[2] and preprocessed with the presolver of the general purpose MIP-solver `SIP` that is currently under development at

[2] Available at `URL` http://www.caam.rice.edu:80/$\sim$bixby/miplib/miplib.html.

Table 2: Decomposing Matrices of Mixed Integer Programs.

	Original			Presolved		Cuts					B&B		Best Solutions				Heuristics at Root				Time				
Name	rows	col	nz	col	nz	Init	Cov	2part	BCC	Pool	Nod	Iter	Lb	Ub	He	No	G	D1	D2	B	Cm	LP	Sep	Heu	Tot
mod008	6	319	1243	1	6	23	0	0	0	0	1	1	3	**3**	IG	1	3	3	3	6	0%	6%	6%	0%	0.1
stein9	13	9	45	9	45	50	115	64	1	23	16	34	7	**7**	IG	1	7	7	7	13	6%	34%	24%	6%	0.3
p0040	13	40	70	30	60	16	0	0	0	0	1	1	3	**3**	*	1	13	13	13	13	0%	0%	40%	0%	0.1
p0033	15	32	97	11	41	41	23	3	0	1	1	3	4	**4**	D1	1	4	4	4	15	0%	50%	16%	0%	0.1
gt1	15	46	92	46	92	19	262	87	0	61	13	33	6	**6**	D1	1	6	6	6	15	4%	47%	26%	4%	0.4
flugpl	16	16	40	6	21	46	16	0	1	0	1	2	1	**1**	ID2	1	3	3	1	16	0%	25%	0%	0%	0.0
bm23	20	27	478	1	20	79	0	0	0	0	1	1	10	**10**	IG	1	10	10	10	20	0%	22%	0%	11%	0.1
enigma	21	100	289	99	287	40	922	216	0	315	28	83	10	**10**	IG	1	10	10	10	21	7%	41%	28%	14%	2.0
air01	23	771	4215	18	125	88	0	0	0	0	1	1	3	**3**	IG	1	3	3	3	23	0%	7%	0%	0%	0.4
rgn	24	180	460	33	102	64	724	163	1	324	13	47	5	**5**	IG	1	5	5	5	24	12%	48%	18%	10%	1.2
pipex	25	48	192	48	192	46	421	180	0	201	13	33	9	**9**	IG	1	9	9	9	25	12%	42%	23%	7%	0.8
lseu	28	88	308	50	185	68	98	43	0	19	1	5	7	**7**	D1	1	7	7	7	28	8%	30%	43%	4%	0.2
gt2	28	173	346	173	346	39	1333	340	0	290	28	86	11	**11**	IG	1	11	11	11	28	8%	46%	28%	8%	2.8
sentoy	30	60	1800	1	30	119	0	0	0	0	1	1	15	**15**	IG	1	15	15	15	30	0%	18%	12%	0%	0.2
stein15	36	15	120	15	120	142	17654	13145	0	26682	5458	7595	18	**18**	IG	1	18	18	18	36	18%	27%	22%	16%	90.1
misc02	43	55	405	46	368	153	2455	1065	0	1612	34	110	15	**15**	IG	9	17	21	21	43	6%	59%	18%	11%	9.2
sample2	45	64	140	55	131	81	976	171	1	331	7	29	4	**4**	D1	1	4	4	4	45	6%	61%	21%	6%	2.0
air02	50	6774	61555	76	897	270	4	0	34	0	1	3	22	**22**	IG	1	22	24	24	50	0%	6%	18%	0%	6.0
misc01	54	79	729	66	678	218	4556	2762	0	4861	367	581	24	**24**	IG	17	25	26	26	54	7%	42%	29%	14%	31.4
mod013	62	96	192	48	144	254	1810	108	1	722	13	43	6	**6**	D1	1	6	6	6	62	6%	68%	17%	3%	5.1
mod014	74	86	172	43	129	246	148	0	1	9	1	3	2	**2**	D1	1	2	2	2	74	5%	32%	35%	8%	0.3
lp4l	85	1086	4677	791	3462	277	23233	16941	0	45877	1129	1779	35	**35**	IG	487	39	41	41	85	4%	13%	27%	52%	481.2
bell5	87	101	257	73	215	171	2418	473	1	800	13	38	4	**4**	ID1	4	6	7	7	87	5%	67%	15%	9%	10.6
p0291	92	103	373	63	279	365	2943	366	0	757	19	54	7	**7**	D1	1	7	7	7	92	7%	52%	21%	15%	10.7
misc03	96	154	2023	133	1934	360	50431	32949	1	85357	12745	15609	43	44	IG	9	46	46	46	96	5%	11%	39%	38%	1800.0
l152lav	97	1989	9922	695	3712	308	32271	23459	0	71535	1525	2276	36	**36**	IG	1102	43	43	43	97	3%	16%	24%	53%	754.5
khb05250	100	1299	2598	1275	2574	196	12634	1917	1	3421	73	227	25	**25**	IG	1	25	25	25	100	3%	25%	42%	25%	89.0
harp2	100	1373	2598	1225	2450	116	250	158	0	3	1	4	17	**17**	D1	1	17	17	17	100	0%	2%	94%	0%	20.4
bell4	101	114	293	84	248	191	7162	1041	1	2157	49	145	5	**5**	IB	4	11	11	11	101	6%	55%	17%	17%	31.7
bell3a	107	121	311	89	263	203	2718	492	1	802	10	32	4	**4**	B	6	13	13	13	107	5%	63%	16%	12%	14.5
bell3b	107	121	311	89	263	203	2718	492	1	802	10	32	4	**4**	B	6	13	13	13	107	5%	63%	15%	13%	14.6
p0201	113	195	1677	177	1527	200	9230	2390	1	5757	46	128	21	**21**	IG	24	30	30	30	113	3%	64%	12%	19%	125.4
stein27	118	27	378	27	378	353	280101	82635	3	382512	1753	3874	32	56	IG	1	56	56	56	118	12%	44%	34%	6%	1800.1
air03	124	10757	91028	656	5878	830	21108	9064	0	24479	130	377	49	**49**	IG	13	58	59	58	124	1%	57%	27%	12%	1146.7
p0808a	136	240	480	120	304	392	14298	1463	1	4711	49	175	8	**8**	D1	1	8	8	8	136	6%	62%	16%	13%	83.2
mod010	146	2655	11203	1973	8404	453	53084	33036	1	118368	727	1170	47	59	IG	28	61	66	66	146	2%	24%	25%	46%	1802.5
blend2	169	319	1279	88	1039	515	0	0	0	0	4	4	10	**10**	IG	1	10	10	10	169	2%	57%	22%	1%	1.7
noswot	182	127	732	50	455	745	84957	34285	2	533391	1006	1646	10	15	IG	10	37	42	42	182	10%	47%	33%	6%	1800.4
10teams	210	1600	9600	1600	9600	210	67719	25710	1	127827	130	418	43	90	D1	1	90	90	90	210	3%	58%	30%	6%	1804.0
misc07	224	254	8589	229	8474	894	10476	8516	0	13110	118	119	57	93	IG	11	101	107	107	224	0%	23%	22%	52%	1836.5
vpm1	234	378	749	203	574	906	18959	446	1	4060	34	117	7	**7**	IG	1	7	14	14	234	3%	80%	9%	5%	322.7
vpm2	234	378	917	203	574	906	18959	446	1	4060	34	117	7	**7**	IG	1	7	14	14	234	3%	79%	9%	6%	317.4
p0808aCUTS	246	240	839	230	828	612	15096	1731	1	3058	22	84	8	**8**	D1	1	8	8	8	246	4%	65%	12%	17%	231.7
p0548	257	477	1522	250	1009	523	94934	20509	1	94842	175	462	25	49	IG	35	68	68	68	257	5%	69%	12%	12%	1800.1
misc05	266	131	2873	129	2869	581	16094	4604	0	5545	43	80	28	116	IG	13	120	126	126	266	1%	89%	5%	3%	1835.0
modglob	289	420	966	356	902	865	75843	1685	1	29153	148	422	8	**8**	IG	18	12	12	12	289	3%	74%	11%	9%	1512.6
gams	291	556	2431	540	2400	1622	41190	11	0	6897	169	295	15	21	B	12	36	36	36	291	1%	84%	9%	3%	1801.1
fiber	297	1232	2644	418	1254	593	32222	7410	1	19253	31	123	9	22	ID1	3	38	38	38	297	2%	86%	4%	6%	1807.5
p0282	305	202	1428	168	841	609	56425	17488	1	52114	91	252	23	36	D1	1	36	36	36	305	3%	71%	9%	14%	1803.2
stein45	331	45	1034	45	1034	992	25305	11011	0	22779	28	88	11	154	IG	2	157	157	157	331	1%	89%	5%	2%	1820.3
qnet1_o	369	1454	4040	837	2474	699	34759	9401	1	19450	37	111	12	28	D1	1	28	28	28	369	2%	74%	6%	15%	1812.7
qnet1	407	1454	4405	924	2843	653	24614	8345	1	16223	22	69	8	35	D1	1	35	35	35	407	2%	77%	5%	14%	1801.5
air05	408	7195	50762	3104	23458	600	5677	1286	0	207	13	19	19	189	IG	1	189	194	194	408	0%	67%	20%	10%	1868.9
set1ch	477	697	1382	445	1130	1437	36069	2756	1	9871	25	89	7	12	D1	1	12	12	12	477	3%	72%	7%	16%	1804.4
fixnet3	478	878	1756	498	1374	1990	23068	145	1	959	34	83	12	13	D1	1	13	13	13	478	1%	88%	4%	4%	1809.4
fixnet4	478	878	1756	498	1374	1990	26165	192	1	1860	34	97	13	**13**	D1	1	13	13	13	478	1%	85%	5%	6%	1789.9
fast0507	484	63001	406865	4927	31419	659	9549	4335	0	2135	10	24	5	183	IG	3	217	217	217	484	1%	51%	15%	26%	1800.1
set1al	492	712	1412	460	1160	1452	27188	2041	1	5790	22	65	6	12	D1	1	12	12	12	492	2%	78%	5%	13%	1816.9
set1cl	492	712	1412	460	1160	1452	26700	2001	1	5711	22	64	6	12	D1	1	12	12	12	492	2%	77%	5%	14%	1805.3
gen	622	797	2064	360	1303	2076	15337	304	1	3914	13	30	5	19	ID1	1	19	19	19	622	1%	68%	5%	24%	1809.8
mod015	622	797	2064	360	1303	2076	14722	299	1	3398	13	28	5	19	ID1	1	19	19	19	622	1%	68%	4%	24%	1807.0
danoint	664	521	3232	521	3232	1016	13895	5307	1	4119	7	24	4	188	ID1	2	189	189	189	664	1%	77%	6%	14%	1861.4
misc06	696	1572	5126	949	3106	1298	15987	3428	1	4911	13	28	4	70	ID1	1	70	70	70	696	2%	62%	7%	27%	1835.3
air06	763	8572	67571	4092	34800	1145	7622	3531	1	732	4	12	3	354	IG	2	356	363	362	763	1%	43%	35%	18%	1934.3
air04	782	8904	70189	4154	35000	1052	6242	3243	1	434	4	10	3	355	IG	1	355	355	355	782	0%	34%	26%	37%	1913.6
adrud	795	998	15876	495	8479	6342	3955	287	0	209	25	32	10	32	D1	1	32	32	32	795	0%	77%	10%	10%	1835.6
$\sum$	14814	134914	876632	35938	221378	43230	1395844	405976	75	1778801	26610	39627	905	2729		1863	2931	2990	2986	14814	2%	63%	15%	17%	56337.8

the Konrad-Zuse-Zentrum. We again considered all instances with up to 1,000 rows.

The picture here is a bit different from the one for the linear programming problems. Since the number of blocks is $\beta = 2$ instead of $\beta = 4$, the IP-formulation is much smaller: The number of variables is only one half, the number of conflicting assignments for two adjacent rows is only 4 instead of 12. In addition, there is much less symmetry in the problem. Consequently, the number of cuts does not increase to the same extent, the LPs are smaller and easier to solve (the percentage of LP-time in column *LP* decreases). We can solve instances up to 200 rows and many of the small ones within seconds. Note that $\beta = 2$, on the other hand, leads to doubled block capacities. This means that it becomes much more difficult to separate inequalities that have this number as their right-hand side and have a large or combinatorially restrictive support like z-clique, bin-packing, or composition of clique inequalities, see column *BCC*.

On the *primal side* the results are similar to the `Netlib` instances, but the heuristics seem to perform a little better for two blocks than for four.

How decomposable are the MIP-matrices? We see that not all, but many of the larger problems can be brought into bordered block diagonal form (the small ones can not). Of course, there are also exceptions like the `air`-problems which were expected to be not decomposable. Anyway, there seems to be potential for the multiple-knapsack approach and further research in this direction, especially because there are only very few classes of cutting planes known for general MIPs.

The second application of matrix decomposition to integer programming is a *new branching rule*. Decomposing the transposed constraint matrix will identify the variables in the border as linking variables that are interesting candidates for branching. Since most MIP-codes create a binary searchtree, we try to decompose these matrices into $\beta := 2$ blocks. As block capacity we use $\kappa := \frac{(\#rows)\cdot 1.05}{2}$ rounded up to obtain two subproblems of roughly the same size. The test set consists of all problems with up to 1,000 rows (1,000 columns in the original problem).

Table 3 shows the *results* of our computations. Surprisingly, the performance of our algorithm is not only similar to the "primal" case, in fact it is even better! We can solve almost all problems with up to 400 rows. One reason for this is that MIPs tend to have sparse columns, but not necessarily sparse rows. Dense columns in the transposed matrices (dense rows in the original ones) leave less freedom for row assignments, there are fewer possibilities for good decompositions, and the LP-relaxations are tighter than in the primal case.

For the reason just mentioned we expected that the transposed problems

Table 3: Decomposing Transposed Matrices of Mixed Integer Programs.

	Original			Presolved		Cuts					B&B		Best Solutions				Heuristics at Root				Time				
Name	rows	col	nz	col	nz	Init	Cov	2part	BCC	Pool	Nod	Iter	Lb	Ub	He	No	G	D1	D2	B	Cm	LP	Sep	Heu	Tot
stein9	9	13	45	1	9	35	0	0	0	0	1	1	5	**5**	IG	1	5	5	5	9	0%	9%	0%	0%	0.1
stein15	15	36	120	1	15	59	0	0	0	0	1	1	8	**8**	IG	1	8	8	8	15	0%	18%	0%	0%	0.2
flugpl	16	16	40	11	30	36	16	2	1	0	1	2	1	**1**	IG	1	1	1	1	16	0%	8%	8%	0%	0.1
bm23	27	20	478	3	78	109	0	0	0	0	1	1	13	**13**	IG	1	13	13	13	27	0%	25%	5%	5%	0.2
stein27	27	118	378	1	27	107	0	0	0	0	1	1	13	**13**	IG	1	13	13	13	27	0%	23%	4%	0%	0.2
p0033	32	15	97	9	59	120	205	51	0	31	7	16	6	**6**	IG	1	6	6	6	32	9%	46%	16%	5%	0.5
p0040	40	13	70	13	70	100	47609	7530	10	27421	1522	3361	13	**13**	IG	1	13	14	14	40	18%	38%	27%	8%	90.5
stein45	45	331	1034	1	45	179	0	0	0	0	1	1	22	**22**	IG	1	22	22	22	45	2%	27%	8%	0%	0.4
gt1	46	15	92	15	92	46	119471	29195	15	71563	4924	8786	16	**16**	IG	20	18	22	22	46	20%	30%	33%	8%	283.8
pipex	48	25	192	19	96	48	142	76	16	0	4	7	20	**20**	IG	2	21	22	22	48	2%	22%	33%	8%	0.4
misc02	55	43	405	18	229	279	547	517	0	562	25	60	27	**27**	D1	1	27	27	27	55	7%	43%	27%	10%	3.8
sentoy	60	30	1800	1	60	239	0	0	0	0	1	1	29	**29**	IG	1	29	29	29	60	1%	33%	9%	1%	0.5
sample2	64	45	140	45	140	82	2223	638	1	1308	13	45	4	**4**	*	9	6	6	6	64	7%	63%	18%	6%	6.6
misc01	79	54	729	23	359	435	2138	1121	0	1229	28	79	38	**38**	IG	1	38	38	38	79	7%	44%	32%	10%	10.3
mod014	86	74	172	74	172	172	6137	697	1	1927	28	109	8	**8**	ID2	1	8	9	8	86	9%	52%	23%	11%	16.7
lseu	88	28	308	20	198	332	10336	3532	0	8335	76	216	26	**26**	*	71	36	42	39	88	7%	64%	15%	10%	65.4
mod013	96	62	192	62	192	192	448495	45097	11	171613	4711	8604	14	20	B	32	23	23	23	96	14%	25%	47%	9%	1800.2
enigma	100	21	289	12	199	375	3695	1802	0	1575	40	110	44	**44**	IG	12	46	48	48	100	6%	40%	37%	11%	19.4
bell5	101	87	257	87	257	185	5598	969	1	1645	40	108	5	**5**	IG	4	9	9	9	101	7%	56%	21%	12%	21.5
p0291	103	92	373	65	319	261	7980	2306	0	3581	61	170	21	**21**	IG	5	24	29	29	103	4%	70%	11%	12%	78.6
bell4	114	101	293	101	293	204	11829	2005	1	3979	109	277	6	**6**	IG	6	9	9	9	114	6%	58%	17%	15%	64.5
bell3a	121	107	311	107	311	217	4322	942	1	1294	10	46	5	**5**	ID1	1	5	5	5	121	5%	63%	17%	11%	23.8
bell3b	121	107	311	107	311	217	4322	942	1	1294	10	46	5	**5**	ID1	1	5	5	5	121	5%	64%	17%	10%	23.7
noswot	127	182	732	103	531	427	55038	4869	1	12790	778	1605	15	**15**	IG	2	19	19	19	127	6%	40%	21%	28%	337.5
misc05	131	266	2873	79	535	289	255523	30000	0	124752	2236	3462	35	49	IG	64	53	58	58	131	8%	38%	30%	19%	1800.0
misc03	154	96	2023	39	769	840	3666	4828	0	8008	328	565	73	**73**	IG	3	74	74	74	154	3%	28%	38%	19%	159.2
gt2	173	28	346	28	346	173	173371	81355	37	280323	1807	2468	57	67	IG	158	76	83	83	173	7%	56%	23%	10%	1800.2
rgn	180	24	460	24	460	730	38963	6126	1	22421	112	347	20	**20**	IG	1	20	78	42	180	5%	67%	14%	11%	474.6
p0201	195	113	1677	77	699	579	181201	53287	74	223647	625	1295	42	76	IG	192	86	93	93	195	8%	54%	26%	8%	1800.0
p0282	202	305	1428	189	1196	407	4311	1449	0	479	13	36	20	**20**	ID1	1	20	20	20	202	2%	74%	9%	13%	122.7
p0808a	240	136	480	136	480	480	106598	14308	1	70632	151	507	11	18	IG	12	32	36	32	240	5%	76%	12%	5%	1802.1
p0808aCUTS	240	246	839	191	729	674	107094	7435	1	58191	202	525	12	17	ID2	28	36	36	36	240	5%	69%	12%	11%	1800.7
misc07	254	224	8589	53	1367	1456	11803	14984	0	28234	757	1117	119	**119**	IG	1	119	121	121	254	2%	18%	40%	29%	939.8
blend2	319	169	1279	160	478	955	103558	8951	1	21986	226	680	38	**38**	D1	1	38	38	38	319	8%	33%	32%	22%	753.6
vpm1	378	234	749	234	749	784	37116	4150	1	14141	37	136	7	**7**	IG	8	51	61	61	378	3%	80%	7%	7%	1402.3
vpm2	378	234	917	66	581	1072	25484	2483	1	6355	19	87	7	**7**	IG	3	21	69	69	378	3%	76%	8%	11%	982.0
modglob	420	289	966	289	966	1014	42137	6770	1	19025	40	116	9	29	IG	12	39	56	39	420	3%	79%	7%	8%	1807.3
p0548	477	257	1522	153	823	1525	36798	3048	1	8883	46	96	29	49	IG	10	79	116	116	477	3%	48%	9%	39%	1803.7
danoint	521	664	3232	544	2336	649	21785	10004	1	12677	13	47	6	210	IG	4	219	238	219	521	2%	79%	8%	9%	1808.8
gams	556	291	2431	91	1096	4003	14962	2869	0	4200	19	35	47	170	IG	1	170	170	170	556	2%	65%	12%	19%	1800.8
set1ch	697	477	1382	477	1382	1657	20852	1581	1	2680	10	34	4	33	IG	4	42	91	91	697	2%	68%	6%	22%	1843.7
set1al	712	492	1412	492	1412	1672	16354	1713	1	2886	7	26	3	40	IG	2	64	101	101	712	2%	76%	5%	15%	1813.6
set1cl	712	492	1412	492	1412	1672	16354	1713	1	2886	7	26	3	40	IG	2	64	101	101	712	2%	76%	5%	15%	1800.2
air01	771	23	4215	22	4185	7186	0	0	0	0	4	2	184	367	IG	1	367	367	367	771	0%	95%	1%	1%	1893.0
gen	797	622	2064	298	1416	2113	16984	1847	1	2813	10	26	4	70	ID2	1	70	84	70	797	2%	70%	7%	19%	1821.7
mod015	797	622	2064	298	1416	2113	16984	1847	1	2813	10	26	4	70	ID2	1	70	84	70	797	2%	70%	7%	19%	1853.2
fixnet3	878	478	1756	478	1756	1638	13145	3715	1	1757	7	18	3	190	IG	1	190	221	209	878	2%	72%	6%	17%	1850.5
fixnet4	878	478	1756	478	1756	1638	13130	3611	1	2272	7	18	3	197	IG	2	209	221	209	878	2%	77%	6%	13%	1873.8
fixnet6	878	478	1756	478	1756	1638	10505	2989	1	901	7	15	3	186	IG	2	209	221	209	878	1%	80%	5%	12%	2087.3
adrud	998	795	15876	1	998	3991	0	0	0	0	1	1	475	**475**	IG	1	475	475	475	998	0%	75%	17%	0%	750.9
$\sum$	14556	10168	72362	6766	35191	45404	2018781	373354	188	1233109	19094	35364	1582	3007		693	3297	3737	3593	14556	4%	66%	13%	14%	41494.7

	SIP without MAD			SIP with MAD			CPLEX		
Name	gap	nodes	time	gap	nodes	time	gap	nodes	time
bell3a	0.0%	33350	92.1	0.0%	55763	221.4	0.0%	19100	108.9
bell3b	0.0%	25909	308.4	0.0%	25706	301.4	0.0%	6537	70.8
bell4	1.5%	91240	1800.0	2.2%	96628	1800.0	0.0%	47001	710.5
bell5	0.1%	100000	806.4	0.1%	100000	824.0	0.1%	100000	586.9
noswot	27.9%	63474	1800.0	20.9%	90743	1800.0	7.5%	43038	1800.1

(a) Strong Branching.

	SIP without MAD			SIP with MAD			CPLEX		
Name	gap	nodes	time	gap	nodes	time	gap	nodes	time
bell3a	96.6%	100000	236.0	67.9%	100000	288.2	0.0%	42446	54.9
bell3b	-	100000	142.1	-	100000	120.3	3.8%	100000	120.6
bell4	-	100000	137.4	2.1%	100000	164.5	3.2%	100000	119.0
bell5	6.3%	100000	184.0	2.7%	100000	153.6	6.1%	100000	113.6
noswot	-	100000	219.8	-	100000	195.2	10.3%	100000	228.1

(b) Maximum Infeasibility Branching.

Table 4: Comparing Integer Programming Branching Rules.

would be less decomposable than the originals and it came as a surprise to us that the "dual" problems decompose nearly as well as the primal ones. The transposed matrices have on average about 60 rows in the border in comparison to only 40 for the originals. But the percentage of rows in the border (i.e., the sum of the *Ub* column divided by the sum of the *rows* column) is 18.4% (12.8%) in the primal and 20.6% (25.3%) in the dual case (the values in parentheses count only instances that were solved to optimality). Note, however, that the dual values are biased heavily by the (not decomposable) `adrud` instance. An explanation may be that many of these problems contain a few important global variables that migrate into the border.

We used optimal decompositions of transposed MIP matrices to test our idea to branch on variables in the border first. The computations were performed using the above mentioned MIP-solver `SIP`. As our *test set* we selected all problems that are not extremely easy (less than 10 CPU seconds for `SIP`) and that decompose into bordered block diagonal form with a "relatively small" border: `mod014, bell3a, bell3b, bell4, bell5, noswot, blend2, vpm1, vpm2, set1ch, set1al` and `set1cl`. Unfortunately, and contrary to what we had expected, it turned out that `mod014, blend2, vpm1, vpm2, set1ch, set1al` and `set1cl` have only *continuos* variables in the border that do not qualify for branching variables! All border variables in `noswot` are integer, in the `bell`-examples 1 (2) out of 5 (6). We tried to extend the test set by decomposing only the integer part of all these

matrices but it failed, because the integer parts turned out to have block diagonal form, i. e., no variables in the border.

For the remaining four `bell*` and the `noswot` example, we performed the following tests. The MIPs were solved with `SIP` using four *different branching strategies*. Maximum infeasibility branching (i.e., branching on a variable that is closest to 0.5) and *strong branching* (cf. Robert E. Bixby, personal communication) are two standard branching strategies for solving mixed integer programs. The other two branching rules result from extending these two methods by our idea of branching on a variable that belongs to the border first. The limit of the computation was 1,800 CPU seconds or 100,000 branch-and-bound nodes, whatever came first.

Table 4 summarizes our results. The version without MAD (MAD stands for the MAtrix Decomposition) uses the original `SIP`-code and the original branching rules, the MAD-version branches on a variable from the border first. For comparison, we also list the results that can be obtained using `CPLEX`. The column labeled *gap* reports the duality gap on termination (- means that no feasible solution was found).

The *results* are mixed. When *strong branching* is used `CPLEX` is overall best and `SIP` with MAD is basically always worst (note that for example *noswot* the value of the LP-relaxation already provides the value of the optimal integer solution and so the main difficulty here is to find a good feasible solution). If we branch on a most infeasible variable the situation changes a bit in favour of `SIP` with MAD. In fact, there are two examples where this strategy performs best in terms of the *gap*. Our limited tests can, of course, not provide a definite evaluation of our idea to branch on border variables. But we think that the results show that this idea might have the potential to become a good branching strategy for certain mixed integer programming problems.

5.3 The Steiner-Tree Packing Problems

We also tested some problem instances for which we know in advance that they have bordered block diagonal from. The problems are integer programming formulations for Steiner tree packing problems, a problem where in some given graph edge sets (so-called Steiner trees), each spanning some given subset of the node set, have to be simultaneously packed in the graph under capacity restrictions on the edges, see Grötschel, Martin & Weismantel (1996) [13]. Unfortunately, our branch-and-cut algorithm performs very bad on these examples, although we extended the time limit to 10,800 CPU seconds, see Table 5. One reason for that might be that the rows that are supposed to be in the border have less nonzero entries than those that are

	Original			Presolved		Cuts					B&B	
Name	rows	col	nz	col	nz	Init	Cov	2part	BCC	Pool	Nod	Iter
g353	81	48	276	48	276	336	19768	6311	240	32467	121	385
g444	114	39	253	37	251	756	11324	4251	156	22828	56	195
d677	324	183	984	174	975	3533	13005	7370	56	54768	25	59
d688	383	230	1350	223	1341	4506	10719	6871	37	32015	19	40
$\sum$	902	500	2863	482	2843	9131	54816	24803	489	142078	221	679

	Best Solutions				Heuristics at Root				Time				
Name	Lb	Ub	He	No	G	D1	D2	B	Cm	LP	Sep	Heu	Tot
g353	16	**16**	IG	4	23	23	23	81	5%	77%	11%	3%	180.7
g444	13	**13**	B	36	22	22	22	114	3%	80%	11%	3%	186.9
d677	7	79	IG	2	98	98	98	324	0%	97%	1%	0%	11168.4
d688	7	108	IG	3	134	140	134	383	0%	97%	1%	0%	10901.3
$\sum$	43	216		45	277	283	277	902	0%	97%	1%	0%	22437.2

Table 5: Decomposing Steiner-Tree Packing Problems.

expected to be in the blocks. The heuristics and the LP solutions, however, tend to put the rows into the border that have the most nonzeros entries. The "natural decompositions" result in borders of sizes 22, 24, 71, and 82 for problems `g353`, `g444`, `d677`, and `d688`, respectively.

5.4 The Equipartition Problems

Our last test set consists of equipartition problems introduced by Nicoloso & Nobili (1992) [21]. These have been generated randomly prescribing a certain matrix-density. We have modified our code to handle the additional equipartition constraint. The results are given in Table 6: We can solve all problems within 10 CPU seconds and, as was already known from Nicoloso & Nobili (1992) [21], random matrices of this type do not decompose well.

Summary and Conclusions

We have shown in this paper that it is possible to decompose typical linear and integer programming matrices up to 200 and more rows to proven optimality using a cutting plane approach based on polyhedral investigations of the matrix decomposition problem. It turned out that a substantial number, but not all, LPs decompose well into four blocks, while even many MIPs, as well as some of their transposes, can be brought into bordered block diagonal form with two blocks. We think that these results show a significant potential for methods that can exploit this structure to solve general MIPs. Our decomposition heuristics work well for small instances, but there is room for improvement for problems of large scale, in particular, if more than two blocks are considered.

	Original			Presolved		Cuts					B&B	
Name	rows	col	nz	col	nz	Init	Cov	2part	BCC	Pool	Nod	Iter
m22	9	6	21	6	21	30	37	19	0	12	16	31
m25	9	6	23	5	22	44	6	20	0	18	13	24
m33	14	9	40	8	37	38	210	90	0	68	13	33
m34	14	9	41	9	41	53	181	111	0	69	22	53
m41	18	9	43	9	43	41	654	241	1	253	28	76
m44	18	9	55	8	51	89	333	122	0	133	25	56
m51	21	14	61	12	58	52	977	349	1	441	52	143
m54	21	14	90	12	83	93	424	211	0	278	34	77
m61	21	17	69	15	65	52	742	204	0	245	28	73
m64	21	17	108	15	100	106	417	285	0	376	40	89
m71	28	15	71	14	69	64	2353	713	0	1171	76	196
m74	28	15	128	14	122	135	823	575	0	714	88	171
m81	28	21	86	20	85	74	3169	979	0	1717	268	476
m84	28	21	177	19	167	172	475	392	617	347	49	116
$\sum$	278	182	1013	166	964	1043	10801	4311	619	5842	752	1614

	Best Solutions				Heuristics at Root				Time				
Name	Lb	Ub	He	No	G	D1	D2	B	Cm	LP	Sep	Heu	Tot
m22	5	**5**	G	1	5	9	9	5	6%	46%	20%	6%	0.1
m25	7	**7**	G	2	9	9	9	9	16%	33%	8%	8%	0.1
m33	6	**6**	G	4	8	10	10	8	8%	40%	17%	20%	0.3
m34	8	**8**	G	1	8	10	10	8	23%	28%	21%	4%	0.5
m41	8	**8**	G	1	8	14	14	10	10%	57%	12%	6%	0.9
m44	10	**10**	G	1	10	10	10	10	11%	35%	35%	5%	0.6
m51	11	**11**	G	11	15	15	15	15	9%	53%	18%	8%	2.1
m54	13	**13**	G	6	15	15	15	15	8%	40%	23%	17%	1.3
m61	9	**9**	G	4	11	11	11	11	14%	49%	22%	4%	1.3
m64	15	**15**	G	5	17	17	17	17	10%	33%	23%	22%	1.6
m71	12	**12**	G	5	16	16	16	16	13%	54%	17%	7%	5.0
m74	20	**20**	G	1	20	22	22	20	8%	33%	14%	35%	4.2
m81	14	**14**	IG	3	16	16	16	16	12%	47%	18%	10%	7.7
m84	22	**22**	*	18	28	28	28	28	7%	28%	23%	35%	3.4
$\sum$	160	160		63	186	202	202	188	11%	43%	19%	16%	29.2

Table 6: Equipartitioning Matrices.

Acknowledgements

We are grateful to the Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) and the German Academic Exchange Service (DAAD) for supporting this work. We enjoyed the cooperation a lot and intend to continue. We want to thank Robert E. Bixby and CPLEX Optimization, Inc., who made the newest β-versions of `CPLEX` and the `CPLEX` Callable Library available to us, and Paolo Nobili for sending a collection of his equipartition problems.

References

[1] C. C. Carøe & R. Schultz. *Dual decomposition in stochastic integer programming.* Oper. Res. Lett. **24**(1–2):37–45, 1999. ZIB preprint SC 96-46 available at http://opus.kobv.de/zib/volltexte/1996/257/. Cited on page 51.

[2] Y. Crama & M. Oosten. *Models for machine-part grouping in cellular manufacturing.* Internat. J. Prod. Res. **34**(6):1693–1713, 1996. Cited on page 60.

[3] D. Dentcheva, R. Gollmer, A. Möller, W. Römisch & R. Schultz. *Solving the unit commitment problem in power generation by primal and dual methods*. In M. Bendsøe & M. Sørensen, (Eds.), *Proc. 9th Conf. Europ. Consortium for Math. in Industry (ECMI)*, pp. 332–339. Teubner Verlag, Stuttgart, 1997. Cited on page 51.

[4] I. Duff, A. Erisman & J. Reid. *Direct Methods for Sparse Matrices*. Oxford Univ. Press, Oxford, 1986. Cited on pages 51, 54.

[5] M. Ehrgott. Optimierungsprobleme in Graphen unter Kardinalitätsrestriktionen. Master's thesis, Universität Kaiserslautern, 1992. Cited on page 61.

[6] C. E. Ferreira, A. Martin & R. Weismantel. *Solving multiple knapsack problems by cutting planes*. SIAM J. Optim. **6**:858–877, 1996. Cited on pages 51, 82.

[7] C. E. Ferreira, A. Martin, C. C. de Souza, R. Weismantel & L. A. Wolsey. *The node capacitated graph partitioning problem: a computational study*. Math. Programming **81**:229–256, 1998. ZIB preprint SC 94-17 available at http://opus.kobv.de/zib/volltexte/1994/146/. Cited on page 80.

[8] C. Fiduccia & R. Mattheyses. *A linear-time heuristic for improving network partitions*. In *DAC '82: Proc. 19th Design Automation Conference*, pp. 175–181. IEEE Press, Piscataway, New Jersey, 1982. ISBN 0-89791-020-6. Cited on page 76.

[9] K. Gallivan, M. T. Heath, E. Ng, J. M. Ortega, B. W. Peyton, R. Plemmons, C. H. Romine, A. Sameh & R. G. Voigt. *Parallel Algorithms for Matrix Computations*. SIAM, 1990. Cited on pages 51, 54.

[10] M. Garey & D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979. Cited on pages 58, 63.

[11] E. Gottlieb & M. Rao. *The generalized assignment problem: Valid inequalities and facets*. Math. Programming **46**:31–52, 1990. Cited on pages 51, 82.

[12] M. Grötschel, L. Lovász & A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, vol. 2 of *Algorithms and Combinatorics*. Springer Verlag, Berlin, 1988. ISBN 3-540-13624-X, 0-387-13624-X (U.S.). Cited on pages 58, 71.

[13] M. Grötschel, A. Martin & R. Weismantel. *Packing Steiner trees: A cutting plane algorithm and computational results*. Math. Programming **72**:125–145, 1996. ZIB preprint SC 92-09 available at http://opus.kobv.de/zib/volltexte/1992/78/. Cited on pages 51, 78, 87.

[14] A. Gupta. *Fast and effective algorithms for graph partitioning and*

sparse matrix ordering. IBM J. Res. Develop. **41**:171–183, 1996. Cited on page 51.

[15] C. Helmberg, B. Mohar, S. Poljak & F. Rendl. *A spectral approach to bandwidth and separator problems in graphs*. Linear and Multilinear Algebra **39**:73–90, 1995. Cited on page 54.

[16] K. L. Hoffman & M. W. Padberg. *Solving airline crew-scheduling problems by branch-and-cut*. Management Sci. **39**:657–682, 1993. Cited on page 68.

[17] CPLEX. *Using the CPLEX Callable Library*. ILOG CPLEX Division, 889 Alder Avenue, Suite 200, Incline Village, NV 89451, USA, 1997. Information available at http://www.cplex.com. Cited on pages 74, 80.

[18] V. Kumar, A. Grama, A. Gupta & G. Karypis. *Introduction to Parallel Computing*. The Benjamin/Cummings Publishing Company, San Francisco, California, 1994. Cited on page 51.

[19] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. Teubner Verlag, Stuttgart, 1990. Cited on page 54.

[20] A. Löbel. *Optimal Vehicle Scheduling in Public Transit*. Berichte aus der Mathematik. Shaker Verlag, Aachen, 1998. ISBN 978-3-8265-3504-8. URL http://www.shaker.de/Online-Gesamtkatalog/Details.asp?ISBN=978-3-8265-3504-8. PdD thesis, Technische Universität Berlin, 1998. Available at http://www.zib.de/bib/books/Loebel.disser.ps. Cited on page 51.

[21] S. Nicoloso & P. Nobili. *A set covering formulation of the matrix equipartition problem*. In P. Kall, (Ed.), *System Modelling and Optimization, Proc. of the 15th IFIP Conf., Zürich, Sept. 1991*, pp. 189–198. Springer Verlag, Berlin, 1992. Cited on pages 52, 55, 61, 78, 88.

[22] P. Nobili & A. Sassano. *Facets and lifting procedures for the set covering polytope*. Math. Programming **45**:111–137, 1989. Cited on page 58.

[23] M. W. Padberg. *On the facial structure of set packing polyhedra*. Math. Programming **5**:199–215, 1973. Cited on pages 53, 57, 58.

[24] A. Pothen, H. D. Simon & K.-P. Liou. *Partitioning sparse matrices with eigenvectors of graphs*. SIAM J. Matrix Anal. Appl. **11**(3):430–452, 1990. Cited on page 54.

[25] E. Rothberg & B. Hendrickson. *Sparse matrix ordering methods for interior point linear programming*. INFORMS J. Comput. **10**(1): 107–113, 1998. Cited on page 51.

[26] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons Ltd, Chichester, 1986. Cited on page 56.

[27] G. Sheble & G. Fahd. *Unit commitment literature synopsis*. IEEE Trans. Power Systems **9**:128–135, 1994. Cited on page 51.

Paper IV

A Bundle Method for Integrated Multi-Depot Vehicle and Duty Scheduling in Public Transit

---

R. Borndörfer, A. Löbel & S. Weider.
A bundle method for integrated multi-depot vehicle and duty scheduling in public transit.
In M. Hickman, P. Mirchandani & S. Voss, (Eds.), *Computer-aided Systems in Public Transport*, vol. 600 of *Lecture Notes in Economics and Mathematical Systems*, pp. 3–24. Springer-Verlag, 2008.

Abstract. This article proposes a Lagrangean relaxation approach to solve integrated duty and vehicle scheduling problems arising in public trantransportsport. The approach is based on the proximal bundle method for the solution of concave decomposable functions, which is adapted for the approximate evaluation of the vehicle and duty scheduling components. The primal and dual information generated by the bundle method is used to guide a branch-and-bound type algorithm.

Computational results for large-scale real-world integrated vehicle and duty scheduling problems with up to 1,500 timetabled trips are reported. Compared with the results of a classical sequential approach and with reference solutions, integrated scheduling offers remarkable potentials in savings and drivers' satisfaction.

Mathematics Subject Classification (MSC 2000). 90C06, 90B06

Keywords. Public transit, integrated vehicle and duty scheduling, proximal bundle method

1 Introduction

The process of *operational planning* in public transit is traditionally organized in successive steps of timetabling, vehicle scheduling, duty scheduling, duty rostering, and crew assignment. These tasks are well investigated in the optimization and operations research literature. An enormous progress has been made in both the theoretical analysis of these problems and in the computational ability to solve them. For an overview see the proceedings of the last five CASPT conferences (Voß & Daduna (2001) [28], Wilson (1997) [29], Daduna, Branco & Paixão (1995) [5], Desrochers & Rousseau (1992) [7], and Daduna & Wren (1988) [4]).

It is well known that the *integrated treatment* of planning steps discloses additional degrees of freedom that can lead to further efficiency gains. The first and probably best known approach in this direction is the so-called *sensitivity analysis*, a method on the interface between timetabling and vehicle scheduling that uses slight shiftings of trips in the timetable to improve the vehicle schedule. The method has been used with remarkable success in HOT and HASTUS, see Daduna & Völker (1997) [3] and Hanisch (1990) [17].

Vehicle and duty scheduling, the topic of this article, is another area where integration is important. The need is largest in *regional scenarios*, which often have few relief points for drivers, such that long vehicle rotations can either not be covered with legal duties at all or only at very high cost. In such scenarios the powerful optimization tools of sequential scheduling are useless. Rather, the vehicle and the duty scheduling steps must be synchronized to produce acceptable results, i.e., an *integrated vehicle and duty scheduling* method is indispensable. Urban scenarios do, of course, offer efficiency potentials as well.

The current *planning systems* provide only limited support for integrated vehicle and duty scheduling. There are frameworks for manual integrated scheduling that allow to work on vehicles and duties simultaneously, rule out infeasibilities, make suggestions for concatenations, etc. Without integrated optimization tools, however, the planner must still build vehicle schedules by hand, anticipating the effects on duty scheduling by skill and experience.

The *literature* on integrated vehicle and duty scheduling is also comparably scant. The first article on the *integrated vehicle and duty scheduling problem* (ISP) that we are aware of was published by Ball, Bodin & Dial (1983) [1]. They describe an ISP at the Baltimore Metropolitan Transit Authority and develop a mathematical model for it. However, they propose to solve this model by decomposing it into its vehicle and duty scheduling parts, i.e., the model is integrated, but the solution method is sequential.

For the next two decades, the predominant approach to the ISP was to include duty scheduling considerations into a vehicle scheduling method or vice versa. The first approach is, e.g., presented by Scott (1985) [26] and Darby-Dowman, J. K. Jachnik & Mitra (1988) [6], who propose two-step methods that first includes some duty scheduling constraints in a vehicle scheduling procedure and afterwards solve the duty scheduling problem in a second step. Examples of the opposite approach are the articles Tosini & Vercellis (1988) [27], Falkner & Ryan (1992) [9], and Patrikalakis & Xerocostas (1992) [25]. They concentrate on duty scheduling and take the vehicle scheduling constraints and costs heuristically into account. A survey of integrated approaches until 1997 can be found in Gaffi & Nonato (1997) [15].

The complete integration of vehicle and crew scheduling was first investigated in a series of publications by Freling and coauthors (Freling (1997) [10]; Freling, Huisman & Wagelmans (2001) [11]; Freling, Wagelmans & Paixao (2001) [12]; Freling, Huisman & Wagelmans (2003) [13]). They propose a combined vehicle and duty scheduling model and attack it by integer programming methods. Computational results on several problems from the Rotterdam public transit company RET with up to 300 timetabled trips, and from Connexxion, the largest bus company in the Netherlands, with up to 653 timetabled trips are reported. A branch-and-price approach to ISP instances involving a single type of vehicles was also described by Friberg & Haase (1997) [14] and tested on artificial data. Another approach to the single-depot ISP is presented in Haase, Desaulniers & Desrosiers (2001) [16]. There a set partitioning model for the duty scheduling problem is used that ensures that also a vehicle schedule can be build. Additional constraints are introduced to count the number of vehicles. This model was tested on artificial data with up to 350 timetabled trips and up to 700 tasks on timetabled trips. It was solved by a branch and price approach using CPLEX as LP-solver.

We propose in this article an integrated vehicle and duty scheduling method similar to that of Freling et al. Our main contribution is the use of bundle techniques for the solution of the Lagrangean relaxations that come up there. The advantages of the bundle method are its high quality bounds and automatically generated primal information that can both be used to guide a branch-and-bound type algorithm. We apply this method to real-world instances from several German carriers with up to 1,500 timetabled trips. As far as we know, these are the largest and most complex instances that have been tackled in the literature using an integrated scheduling approach. Our optimization module IS-OPT has been developed in a joint research project with IVU Traffic Technologies AG (IVU), Mentz Datenverarbeitung GmbH (mdv), and the Regensburger Verkehrsbetriebe (RVB). It is incorporated in IVU's commercial scheduling system MICROBUS 2 (now ivu.plan).

The article is organized as follows. Section 3 gives a formal description of the ISP and states an integer programming model that provides the basis of our approach. Section 4 describes our scheduling method. We discuss the Lagrangean relaxation that arises from a relaxation of the coupling constraints for the vehicle and the duty scheduling parts of the model, the solution of this relaxation by the proximal bundle method, in particular, the treatment of inexact evaluations of the vehicle and duty scheduling component functions, and the use of primal and dual information generated by the bundle method to guide a branch-and-bound algorithm. Section 5 reports computational results for large-scale real-world data. In particular, we apply our integrated scheduling method to mostly urban instances for the German city of Regensburg with up to 1,500 timetabled trips.

2 Notation

We use the following notation to state linear and integer programs: Let A be a set with a finite number of elements and X be an arbitrary set. Let $x \in X^A$ be a vector with $|A|$ elements, every element is in X. Let $b : A \mapsto \{1, 2, \dots |A|\}$ be a bijection. Then we denote by $x_a, a \in A$ the $b(a)$-th entry of x. A sum over a subset of entries of x is denoted by $x(B) := \sum_{a \in B} x_a$, for all $B \subset A$. A vector consisting only of a subset of entries $B \subset A$ is denoted by x_B.

3 Integrated Vehicle and Duty Scheduling

The *integrated vehicle and duty scheduling problem* (ISP) contains a vehicle and a duty scheduling part. We describe these individual parts first and conclude with the integrated scheduling problem. The exposition assumes that the reader is familiar with the terminology of vehicle and duty scheduling; suitable references are Löbel (1998) [24] for vehicle scheduling and Borndörfer, Grötschel & Löbel (2003) [2] for duty scheduling.

The vehicle scheduling part of the ISP is based on an acyclic directed multigraph $G = (\mathcal{T} \cup \{s, t\}, \mathcal{D})$. The nodes of G are the set $\mathcal{T}$ of *timetabled trips* plus two additional artificial nodes s and t, which represent the beginning and the end of a vehicle rotation, respectively; s is the source of G and t the sink. The arcs $\mathcal{D}$ of G are called *deadheads*, the special deadheads that emanate from the source s are the *pull-out trips*, those entering the sink t are the *pull-in trips*. Associated with each deadhead a is a *depot* $g_a \in \mathcal{G}$ from some set $\mathcal{G}$ of *depots* (i.e., vehicle types), that indicates a valid vehicle type, and a *cost* $d_a \in \mathbb{Q}$. There may be parallel arcs in G with different de-

pots and costs. We denote by $\mathcal{D}_g := \{a \in \mathcal{D} : g_a = g\}$ the set of deadheads that can be covered by a vehicle of type $g \in \mathcal{G}$, by $\delta_g^+(v) := \delta^+(v) \cap \mathcal{D}_g$ the outcut of node v, restricted to arcs in $\mathcal{D}_g$, and by $\delta_g^-(v) := \delta^-(v) \cap \mathcal{D}_g$ the incut of node v, restricted to arcs in $\mathcal{D}_g$.

A *vehicle rotation* or *block* of type $g \in \mathcal{G}$ is an st-path in G that uses only deadheads of type g, i.e., an st-path p such that $p \subseteq \mathcal{D}_g$ for some depot $g \in \mathcal{G}$. A vehicle schedule is a set of blocks such that each timetabled trip is contained in one and only one block. The *vehicle scheduling problem* (VSP) is to find a vehicle schedule of minimal cost. It can be stated as the following integer program:

$$
\begin{array}{lrll}
\text{(VSP)} & \min\ d^{\mathrm{T}}y & & \\
\text{(i)} & y(\delta_g^+(v)) - y(\delta_g^-(v)) & = 0 & \forall v \in \mathcal{T}, g \in \mathcal{G} \\
\text{(ii)} & y(\delta^+(v)) & = 1 & \forall v \in \mathcal{T} \\
\text{(iii)} & y(\delta^-(v)) & = 1 & \forall v \in \mathcal{T} \\
\text{(iv)} & y & \in \{0,1\}^{\mathcal{D}} &
\end{array}
$$

The duty scheduling part of the ISP also involves an acyclic digraph $D = (\mathcal{R} \cup \{s,t\}, \mathcal{L})$. The nodes of D consist of a set of *tasks* $\mathcal{R}$ plus two artificial nodes s and t, which mark the beginning and the end of a *part of work* of a duty; again s is the source of D and t the sink. A task r can correspond either to a timetabled trip $v_r \in \mathcal{T}$ or to a deadhead trip $a_r \in \mathcal{D}$; there may also be additional tasks independent of the vehicle schedule that model sign-on and sign-off times and similar activities of drivers.

Let $\mathcal{R}_{\mathcal{T}}$ and $\mathcal{R}_{\mathcal{D}}$ be the sets of tasks that correspond to a timetabled trip and a deadhead trip, respectively. We assume that there is at least one task associated with every timetabled trip and every deadhead trip; these tasks correspond to units of driving work on such a trip. Several tasks for one trip indicate that this trip is subdivided by relief opportunities to exchange a driver into several units of driving work. The arcs $\mathcal{L}$ of D are called *links*; they correspond to feasible concatenations of tasks in a potential duty. A *part of work* of a duty is an st-path p in D that corresponds to the certain legality rules and has a certain cost c_p, again determined by certain rules. A *duty* is a concatenation of one or more (usually one or two) compatible parts of work.

Denote by $\mathcal{S}$ the set of all such duties, and by c_p, $p \in \mathcal{S}$, their costs. Let further $\mathcal{S}_r := \{p \in \mathcal{S} : r \in p\}$ be the set of all duties that contain some task $r \in \mathcal{R}$ and let $\mathcal{D}_r \subset \mathcal{D}$ be the set of deadheads that imply the use of task r. Given a vehicle schedule y, a *compatible duty schedule* is a collection of duties such that each task that corresponds to either a timetabled trip or a deadhead trip from the vehicle schedule is contained in exactly one duty, while the tasks corresponding to deadhead trips that are not contained in the vehicle schedule are not contained in any duty. The *duty scheduling*

problem associated with a vehicle schedule y is to find a compatible duty schedule of minimum cost. The DSP can be stated as the following integer program:

$$
\begin{array}{lrll}
(\mathrm{DSP_y}) & \min\ c^{\mathrm{T}}x & & \\
\text{(i)} & x(\mathcal{S}_r) & = 1 & \forall r \in \mathcal{R}_{\mathcal{T}} \\
\text{(ii)} & x(\mathcal{S}_r) & = y_a & \forall (r,a) \in \mathcal{R} \times \mathcal{D} \text{ with } a \in \mathcal{D}_r \\
\text{(iii)} & x & \in \{0,1\}^{\mathcal{S}} &
\end{array}
$$

Such kind of models have in general to many variables to be solved directly. Therefore column generation approaches are in use. The first publication, to our knowledge, in which such an approach to the duty scheduling problem is described is Desrochers & Soumis (1989) [8].

The *integrated vehicle and duty scheduling problem* is to simultaneously construct a vehicle schedule and a compatible duty schedule of minimum overall cost. Introducing suitable constraint matrices and vectors, the ISP reads:

$$
\begin{array}{lrcrl}
(\mathrm{ISP}) & \min\ d^{\mathrm{T}}y & + & c^{\mathrm{T}}x & \\
\text{(i)} & Ny & & & = b \\
\text{(ii)} & & & Ax & = \mathbf{1} \\
\text{(iii)} & My & - & Bx & = 0 \\
\text{(iv)} & y & & & \in \{0,1\}^{\mathcal{D}} \\
\text{(v)} & & & x & \in \{0,1\}^{\mathcal{S}}
\end{array}
$$

In this model, the *multiflow constraints* (ISP) (i) correspond to the vehicle scheduling constraints (VSP) (i)–(iii); they generate a feasible vehicle schedule. The *(timetabled) trip partitioning constraints* (ISP) (ii) are exactly the duty scheduling constraints (DSP_y) (i); they make sure that each timetabled trip is covered by exactly one duty. Finally, the *coupling constraints* (ISP) (iii) correspond to the duty scheduling constraints (DSP_y) (ii)–(iii); they guarantee that the vehicle and duty schedules x and y are synchronized on the deadhead trips, i.e., a deadhead trip is either assigned to both a vehicle and a duty or to none. The structure of the coupling constraints can later be used to reduce the number of duties, deadheads, and also coupling constraints if variables are fixed in a branch-and-bound like algorithm.

We remark that a practical version includes several types of additional constraints such as depot capacities, and duty scheduling base constraints (e.g. duty type capacities, average paid/working times), which we omit in this article. The inclusion of such constraints in our scheduling method is, however, straightforward.

The integrated scheduling model (ISP) consists of a multicommodity flow model for vehicle scheduling and a set partitioning model for duty scheduling on timetabled trips. These two models are joined by a set of coupling

constraints for the deadhead trips, one for each task on a deadhead trip. The model (ISP) is the same as that used by Freling and coauthors, see Freling (1997) [10].

4 A Bundle Method

Our general solution strategy for the ISP is based on a Lagrangean relaxation of the coupling constraints (ISP) (iii). This decomposes the problem into a vehicle scheduling subproblem, a duty scheduling subproblem, and a Lagrangean master problem. All three of these problems are large scale, but of quite different nature. Efficient methods are available to solve vehicle scheduling problems of the sizes that come up in an integrated approach with a very good quality or even to optimality. We use the method of Löbel (1998) [24]. Duty scheduling is, in fact, the hardest part. We are not aware of methods that can produce high quality lower bounds for large-scale real-world instances. However, duty scheduling problems can be tackled in a practically satisfactory way using column generation algorithms, see Borndörfer, Grötschel & Löbel (2003) [2] for the algorithm we used to "solve" our duty scheduling subproblems. In the Lagrangean master, multipliers for several tens of thousands of coupling constraints have to be determined. Here, the complexity of the vehicle and the duty scheduling subproblems demands a method that converges quickly and that can be adapted to inexact evaluation of the subproblems. The *proximal bundle method* by Kiwiel (1995) [22] has these properties; it further produces primal information that can be used in a superordinate branch-and-bound algorithm to guide the branching decisions. Moreover, the large dimension of the Lagrangean multiplier space, a potential computational obstacle, can be collapsed by a simple dualization.

This section discusses our Lagrangean relaxation/column generation approach to the ISP using the proximal bundle method. In a first phase, the procedure aims at the computation of an "estimation" of a global lower bound for the ISP *and* at the computation of a set of duties that is likely to contain the major parts of a good duty schedule. This procedure constitutes the core of our integrated vehicle and duty scheduling method. In a second phase, the bundle core is called repeatedly in a branch-and-bound type procedure to produce integer solutions.

4.1 Lagrangean Relaxation

Lagrangean relaxation is a tool to find lower bounds on a minimization problem. A good introduction to it and an overview of applications and its variants can be found in Lemaréchal (2001) [23].

We consider in this subsection a restriction (ISP_I) of the ISP to some subset of duties $I \subseteq \mathcal{S}$ that have been generated explicitly (in some way). This set I may change (grow and shrink) from one iteration to another in our algorithm, however for the next two sections we keep it constant, in Section 4.3 we will describe the dynamic case:

$$\begin{array}{lrrrrl}
(\mathrm{ISP_I}) & \min & d^{\mathrm{T}}y & + & c_I^{\mathrm{T}}x^I & \\
\text{(i)} & & Ny & & & = d \\
\text{(ii)} & & & & A_I x^I & = \mathbf{1} \\
\text{(iii)} & & My & - & B_I x^I & = 0 \\
\text{(iv)} & & y & & & \in \{0,1\}^{\mathcal{D}} \\
\text{(v)} & & & & x^I & \in \{0,1\}^I
\end{array}$$

A Lagrangean relaxation with respect to the coupling constraints (ISP_I) (iii) and a relaxation of the integrality constraints (iv) and (v) results in the Lagrangean dual

$$(\mathrm{L}_I) \quad \max_{\lambda} \left[\min_{\substack{Ny=d,\\ y\in[0,1]^{\mathcal{D}}}} (d^{\mathrm{T}} - \lambda^{\mathrm{T}}M)y + \min_{\substack{A_I x^I=\mathbf{1},\\ x^I\in[0,1]^I}} (c_I^{\mathrm{T}} + \lambda^{\mathrm{T}}B_I)x^I \right].$$

Define functions and associated arguments by

$$\begin{array}{lll}
f_V & : \mathbb{R}^{\mathcal{R}_{\mathcal{D}}} \to \mathbb{R}, \quad \lambda \mapsto \min(d^{\mathrm{T}} - \lambda^{\mathrm{T}}M)y; & Ny = d; \quad y \in [0,1]^{\mathcal{D}} \\
f_D^I & : \mathbb{R}^{\mathcal{R}_{\mathcal{D}}} \to \mathbb{R}, \quad \lambda \mapsto \min(c^{\mathrm{T}} + \lambda^{\mathrm{T}}B_I)x^I; & A_I x^I = \mathbf{1}; \quad x^I \in [0,1]^I \\
f^I & := f_V + f_D^I, &
\end{array}$$

and

$$\begin{array}{ll}
y(\lambda) & := \operatorname{argmin}_{y\in[0,1]^{\mathcal{D}}} f_V(\lambda); \quad Ny = d \\
x^I(\lambda) & := \operatorname{argmin}_{x^I\in[0,1]^I} f_D^I(\lambda); \quad A_I x^I = \mathbf{1}
\end{array}$$

breaking ties arbitrarily. With this notation, (L_I) becomes

$$(\mathrm{L}_I) \quad \max_{\lambda} f^I(\lambda) = \max_{\lambda} \left[f_V(\lambda) + f_D^I(\lambda) \right].$$

The functions f_V and f_D^I are concave and piecewise linear. Their sum f^I is therefore a decomposable, concave, and piecewise linear function; f^I is, in particular, nonsmooth. This is precisely the setting for the proximal bundle method.

4.2 The Proximal Bundle Method

The *proximal bundle method* (PBM) is a subgradient-type procedure for concave functions. It can be adapted to handle decomposable, nonsmooth functions in a particularly efficient way.

We recall the method in this section as far as we need for our exposition. An in-depth treatment can be found in the articles Kiwiel (1990) [21]; Kiwiel (1995) [22].

When applied to (L_I), the PBM produces two sequences of iterates $\lambda_i, \mu_i \in \mathbb{R}^{\mathcal{R}_\mathcal{D}}$, $i = 0, 1, \ldots$. The points μ_i are called *stability centers*; they converge to a solution of (L_I). The points λ_i are trial points; calculations at the trial points result either in a shift of the stability center, or in some improved approximation of f^I.

More precisely, the PBM computes at each iterate λ_i linear approximations

$$\begin{aligned}
\bar{f}_V(\lambda;\lambda_i) &:= f_V(\lambda_i) + g_V(\lambda_i)^{\mathrm{T}}(\lambda - \lambda_i)\\
\bar{f}^I_D(\lambda;\lambda_i) &:= f^I_D(\lambda_i) + g^I_D(\lambda_i)^{\mathrm{T}}(\lambda - \lambda_i)\\
\bar{f}^I(\lambda;\lambda_i) &:= \bar{f}_V(\lambda;\lambda_i) + \bar{f}^I_D(\lambda;\lambda_i)
\end{aligned}$$

of the functions f_V, f^I_D, and f^I by determining the function values $f_V(\lambda_i)$, $f^I_D(\lambda_i)$ and the subgradients $g_V(\lambda_i)$ and $g^I_D(\lambda)$; by definition, these approximations overestimate the functions f_V and f^I_D, i.e., $\bar{f}_V(\lambda;\lambda_i) \geq f_V(\lambda)$ and $\bar{f}^I_D(\lambda;\lambda_i) \geq f^I_D(\lambda)$ for all λ. Note that $\bar{f}_V$ and $\bar{f}^I_D$ are polyhedral, such the subgradients can be derived from the arguments $y(\lambda_i)$ and $x^I(\lambda_i)$ associated with the multiplier λ_i as

$$\begin{aligned}
g_V(\lambda_i) &:= - \quad My(\lambda_i)\\
g^I_D(\lambda_i) &:= \quad\; B_I x^I(\lambda_i)\\
g^I(\lambda_i) &:= - \quad My(\lambda_i) + B_I x^I.
\end{aligned}$$

We call the sets of linearizations collected until iteration i *bundles* and denote them by $J_{V,i}$ and $J_{D,i}$. For implementational issues an affine function $\bar{f}$ can be stored as a tuple of its function value at the origin and its gradient: $(\bar{f}(0), \nabla\bar{f})$. The PBM uses the bundles to build piecewise linear approximations

$$\begin{aligned}
\hat{f}_{V,i}(\lambda) &:= \min_{\bar{f}_V \in J_{V,i}} \bar{f}_V(\lambda)\\
\hat{f}_{D,i}(\lambda) &:= \min_{\bar{f}_D \in J_{D,i}} \bar{f}_D(\lambda)\\
\hat{f}_i &:= \hat{f}_{V,i} + \hat{f}_{D,i}
\end{aligned}$$

Require: Starting point $\lambda_0 \in \mathbb{R}^n$, weights $u_0, m > 0$, optimality tolerance $\epsilon \geq 0$.

1: Initialization: $i \leftarrow 0$, $J_{V,i} \leftarrow \{\lambda_i\}$, $J_{D,i} \leftarrow \{\lambda_i\}$, and $\mu_i = \lambda_i$.
2: Direction Finding: Compute λ_{i+1}, $\tilde{g}_{V,i}$, $\tilde{g}_{D,i}$ by solving problem (QP_i).
3: Function evaluation: Compute $f_V(\lambda_{i+1})$, $g_V(\lambda_{i+1})$, $f_D^I(\lambda_{i+1})$, $g_D^I(\lambda_{i+1})$.
4: Stopping Criterion: If $\hat{f}_i(\lambda_{i+1}) - f^I(\mu_i) < \epsilon(1 + \left|f^I(\mu_i)\right|)$ output μ_i, terminate.
5: Bundle Update:
6: Select $J_{V,i+1} \subseteq J_{V,i} \cup \{\bar{f}_V(\cdot, \lambda_{i+1}), \tilde{f}_{V,i}\}$,
7: select $J_{D,i+1} \subseteq J_{D,i} \cup \{\bar{f}_D^I(\cdot, \lambda_{i+1}), \tilde{f}_{D,i}\}$.
8: Ascent Test: $\mu_{i+1} \leftarrow f^I(\lambda_{i+1}) - f^I(\mu_i) > m(\hat{f}_i(\lambda_{i+1}) - f^I(\mu_i))\,?\,\lambda_{i+1} : \mu_i$.
9: Weight Update: Set u_{i+1}.
10: $i \leftarrow i + 1$, goto step 2.

Algorithm 1: Generic Proximal Bundle Method (PBM).

of f_V, f_D^I, and f^I. Adding a quadratic term to this model that penalizes large deviations from the current stability center μ_i, the next trial point λ_{i+1} is calculated by solving the quadratic programming problem

$$(\mathrm{QP}_i) \qquad \lambda_{i+1} := \operatorname{argmax}_{\lambda} \hat{f}_i(\lambda) - \tfrac{u}{2} \|\mu_i - \lambda\|^2 .$$

Here, u is a positive weight that can be adjusted to increase accuracy or convergence speed. If the approximated function value $\hat{f}_i(\lambda_{i+1})$ at the new iterate λ_{i+1} is sufficiently close to the function value $f^I(\mu_i)$, the PBM stops; μ_i is the approximate solution. Otherwise a test is performed whether the predicted increase $\hat{f}_i(\lambda_{i+1}) - f^I(\mu_i)$ leads to sufficient real increase $f^I(\lambda_{i+1}) - f^I(\mu_i)$; in this case, the model is judged accurate and the stability center is moved to $\mu_{i+1} := \lambda_{i+1}$. The bundles are updated by adding the information computed in the current iteration, and, possibly, by dropping some old information. Then the next iteration starts, see Algorithm 1 for a listing (the affine functions $\tilde{f}_{V,i}$ and $\tilde{f}_{D,i}$ will be defined and explained in a second).

Besides function and subgradient calculations, the main work in the PBM is the solution of the quadratic problem QP_i. This problem can also be stated as

$$\begin{array}{llllll}
(\mathrm{QP_i}) & \max & v_V + & v_D & -\frac{u}{2}\|\mu_i - \lambda\|^2 & \\
\text{(i)} & & v_V & & -\bar{f}_V(\lambda) \leq 0 & \forall \bar{f}_V \in J_{V,i} \\
\text{(ii)} & & & v_D & -\bar{f}_D(\lambda) \leq 0 & \forall \bar{f}_D \in J_{D,i}.
\end{array}$$

A dualization and some transforming using the optimality criterion $0 \in$

$\partial \hat{f}_i(\lambda) + u(\mu_i - \lambda)$ of (QP_i) results in the equivalent formulation

$$
\begin{aligned}
(\mathrm{DQP_i}) \qquad \max \quad & \sum_{\bar{f}_V \in J_{V,i}} \alpha_{V,\bar{f}_V} \bar{f}_V(\mu_i) + \sum_{\bar{f}_D \in J_{D,i}} \alpha_{D,\bar{f}_D} \bar{f}_D(\mu_i) \\
& - \tfrac{1}{2u} \left\| \sum_{\bar{f}_V \in J_{V,i}} \alpha_{V,\bar{f}_V} \nabla \bar{f}_V + \sum_{\bar{f}_D \in J_{D,i}} \alpha_{D,\bar{f}_D} \nabla \bar{f}_D \right\|^2 , \\
& \sum_{\bar{f}_V \in J_{V,i}} \alpha_{V,\bar{f}_V} = 1, \\
& \sum_{\bar{f}_D \in J_{D,i}} \alpha_{D,\bar{f}_D} = 1, \\
& \alpha_V, \alpha_D \geq 0.
\end{aligned}
$$

Here, $\alpha_V \in [0,1]^{J_{V,i}}$ and $\alpha_D \in [0,1]^{J_{D,i}}$ are the dual variables associated with the constraints (QP_i) (i) and (ii), respectively. Note that (DQP_i) is again a quadratic program, the dimension of which is equal to the size of the bundles, while its codimension is only two. In our integrated scheduling method, we solve (DQP_i) using a specialized version of the *spectral bundle method* of Helmberg (2000) [18], a variant of the proximal bundle method that can take advantage of this special structure. Given a solution (α_V, α_D) of DQP_i, the vectors

$$
\begin{aligned}
\tilde{g}_{V,i} &:= \textstyle\sum_{\bar{f}_V \in J_{V,i}} \alpha_{\bar{f}_V} \nabla \bar{f}_V \\
\tilde{g}_{D,i} &:= \textstyle\sum_{\bar{f}_D \in J_{D,i}} \alpha_{\bar{f}_D} \nabla \bar{f}_D \\
\tilde{g}_i &:= \tilde{g}_{V,i} + \tilde{g}_{D,i}
\end{aligned}
$$

are convex combinations of subgradients; they are called *aggregated subgradients* of the functions f_V, f_D^I, and f^I, respectively. It can be shown that they are, actually, subgradients of the respective linear models of the functions at the point λ_{i+1} and, moreover, that this point can be calculated by means of the formula

$$
\lambda_{i+1} = \mu_i + \frac{1}{u} \left[\sum_{\bar{f}_V \in J_{V,i}} \alpha_{V,\bar{f}_V} \nabla \bar{f}_V + \sum_{\bar{f}_D \in J_{D,i}} \alpha_{D,\bar{f}_D} \nabla \bar{f}_D \right].
$$

The aggregated subgradients can be used to define linearizations of $\hat{f}_{V,i}$, $\hat{f}_{D,i}$, and $\hat{f}_i$, at λ_{i+1}:

$$
\begin{aligned}
\tilde{f}_{V,i}(\lambda) &:= \hat{f}_{V,i}(\lambda_{i+1}) + \tilde{g}_{V,i}^{\mathrm{T}}(\lambda - \lambda_{i+1}) \\
\tilde{f}_{D,i}(\lambda) &:= \hat{f}_{D,i}(\lambda_{i+1}) + (\tilde{g}_{D,i})^{\mathrm{T}}(\lambda - \lambda_{i+1}) \\
\tilde{f}_i(\lambda) &:= \hat{f}_i(\lambda_{i+1}) + \tilde{g}_i^{\mathrm{T}}(\lambda - \lambda_{i+1})
\end{aligned}
$$

To calculate primal approximations we use aggregated arguments:

$$\begin{aligned}\tilde{x}_i &:= \textstyle\sum_{\bar{f}_D \in J_{D,i}} \alpha_{\bar{f}_D} x(\bar{f}_D) \\ \tilde{y}_i &:= \textstyle\sum_{\bar{f}_V \in J_{V,i}} \alpha_{\bar{f}_V} y(\bar{f}_V)\end{aligned}$$

With $x(\bar{f}_D)$ or $y(\bar{f}_V)$ is the argument associated with the affine function $\bar{f}_D$ or $\bar{f}_V$, respectively. The PBM (without stopping) is known to have the following properties:

- The series (μ_i) converges to an optimal solution of L_I, i.e., an optimal dual solution of the LP-relaxation of (ISP_I).
- The series $(\tilde{y}_i, \tilde{x}_i)$ converges to an optimal primal solution of the LP-relaxation of (ISP_I).
- Convergence is preserved if, at every iteration i the bundles contain at least two affine functions, namely, the last linearizations $\bar{f}^I_V(\cdot;\lambda_i)$, $\bar{f}^I_D(\cdot;\lambda_i)$ and the linearization of the cutting plane model $\tilde{f}_{D,i}, \tilde{f}_{V,i}$, see step 5 of Algorithm 1.

The bundle size controls the convergence speed of the PBM. If large bundles are used, less iterations are needed, however, problem (QP^I_i) becomes more difficult. We limit the bundle size for both bundles $J_{V,i}$ and J_{D_i} to 500. That means for our instances practically no limit, since we usually perform less than 500 iterations of the bundle method. We use such large bundles because computation time to solve problem (DQP_i) is in comparison to the time needed for the column generation very short even for this size of bundles.

4.3 Adaptations of the Bundle Method

Two obstacles prevent the straightforward application of the proximal bundle method to the ISP. First, the component problem for duty scheduling is $\mathcal{NP}$-hard, even in its LP-relaxation; the vehicle scheduling LP is computationally at least not easy. We can therefore not expect that we can compute the function values $f_V(\lambda_i)$ and $f^I_D(\lambda_i)$ and the associated subgradients $g_V(\lambda_i)$ and $g^I_D(\lambda_i)$ exactly. The algorithms Löbel (1998) [24] and Borndörfer, Grötschel & Löbel (2003) [2] that we use provide in general only approximate solutions. Second, the column generation algorithm process that is carried out for the duty scheduling problem must be synchronized with the bundle method. That is the set I changes throughout the bundle algorithm.

The literature gives two versions of approximate versions of the bundle method that can deal with inexact evaluations of the component functions. Kiwiel (1995) [22] stated a version of the PBM that asymptotically produces

a solution, given that ϵ-linearizations of the function f to be minimized can be found at every trial point $\mu \in \mathbb{R}^m$ for all $\epsilon > 0$, i.e., one can find an affine function $\bar{f}_\epsilon(\lambda;\mu) := f_\epsilon(\mu) + g_\epsilon(\mu)^{\mathrm{T}}(\lambda - \mu)$ such that $f_\epsilon(\mu) \geq f(\mu) - \epsilon$ and $f(\lambda) \geq \bar{f}_\epsilon(\lambda;\mu)$ for all $\lambda \in \mathbb{R}^m$.

Hintermüller (2001) [19] gave another version which replaces exact subgradients of f by ϵ-subgradients. In his method it is not necessary to know or control the actual value of ϵ; his method produces solutions that are as good as the supplied ϵ-subgradients. They converge, in particular, to the optimum if the linear approximation converges to the original function.

We could use these approaches in principle in our setting, but at a high computational cost and with only limited benefit. In fact, our vehicle scheduling algorithm produces not only a primal solution, but also a lower bound and an adequate subgradient from a certain single-depot relaxation of the vehicle scheduling problem. However, the information that can be derived from the subgradients associated with this single-depot relaxation was not very helpful in our computational experiments. Concerning the duty scheduling part, we are also able to compute a lower bound and adequate subgradients for the duty scheduling component function f_D^I for any fixed column set using exact LP-techniques. However, this is a lot of effort for a bound that is not globally valid. We remark that one can, at least in principle, also compute a lower bound for the entire duty scheduling function f_D, see Borndörfer, Grötschel & Löbel (2003) [2]. Such procedures are, however, extremely time consuming and do not yield high quality bounds for large-scale problems. Therefore we use a different, much faster approach to approximate the component functions themselves by piecewise linear functions. We show below how this can be done rigorously for the vehicle scheduling part; in the duty scheduling part, the procedure is heuristic, and we simply update our approximation whenever we notice an error.

Vehicle Scheduling Function f_V. Denote by $f_V^L : \mathbb{R}^{\mathcal{D}} \mapsto \mathbb{R}$ the approximation to the value of the vehicle scheduling component function $f_V(\lambda)$ as given by some vehicle scheduling algorithm, and by $y^L(\lambda) \in [0,1]^{\mathcal{D}}$ the associated argument. We have $f_V^L(\lambda) := (d^{\mathrm{T}} - \lambda^{\mathrm{T}}M)y^L(\lambda) \geq f_V(\lambda)$, but f_V^L is in general not concave. However, we can use f_V^L to create a concave approximation $\hat{f}_{V,i}^L \geq f_V$ using a linearization at the current trial point λ_{i+1} and the linearizations stored in the bundle, namely, by setting

$$\begin{aligned} g_{V,i+1}^L &:= -My^L(\lambda_{i+1}) \\ \bar{f}_V^L(\lambda;\lambda_{i+1}) &:= f_V^L(\lambda_{i+1}) + {g_{V,i+1}^L}^{\mathrm{T}}(\lambda - \lambda_{i+1}) \\ \hat{f}_{V,i+1}^L(\lambda) &:= \min_{\bar{f}_V \in J_{V,i} \cup \{\bar{f}_V^L(\cdot;\lambda_{i+1})\}} \bar{f}_V(\lambda). \end{aligned}$$

We use this approximation in the PBM Algorithm 1 by replacing f_V by $\hat{f}^L_{V,i}$. The bundle update (step 5) is implemented as

$$J_{V,i+1} \subset \begin{cases} J_{V,i} \cup \{\bar{f}^L_V(\cdot;\lambda_{i+1}), \tilde{f}_{V,i}\}, & \text{if } f^L_V(\lambda_{i+1}) < \hat{f}^L_{V,i+1}(\lambda_{i+1}), \\ J_{V,i}, & \text{otherwise.} \end{cases} \tag{1}$$

Since the function $\hat{f}^L_{V,i+1}$ depends on $J_{V,i}$, we must also recalculate its value $\hat{f}^L_{V,i+1}(\mu_i)$ at the stability center in the stopping criterion and the ascent test (steps 4 and 6) of the PBM at each iteration.

Duty Scheduling Function f^I_D. The idea is similar as in the vehicle scheduling case. Denote by I_i the duty set that is used in iteration i, by $f^{L,I_i}_D : \mathbb{R}^{\mathcal{D}} \mapsto \mathbb{R}$ a lower bound of the duty scheduling component function $f^{I_i}_D(\lambda)$ and by $x^{L,I_i}(\lambda_i)$ the argument of f^{L,I_i}_D computed again by the bundle algorithm. Here we have $f^{L,I_i}_D(\lambda) \leq f^{I_i}_D(\lambda)$, and f^{L,I_i}_D is in general not concave. Further we know $f^{I_i}_D(\lambda) \geq f_D(\lambda)$. Thus, $f^{L,I_i}_D(\lambda)$ can be smaller or larger than $f_D(\lambda)$, what we actually want to maximize.

Similar, but this time heuristically, we use f^{L,I_i}_D and the current bundle to create a concave approximation $\hat{f}^L_{D,i}$ of f_D, namely,

$$\begin{aligned} g^L_{D,i+1} &:= B_{I_i} x^{L,I_i}(\lambda_{i+1}) \\ \bar{f}^{L,I_i}_D(\lambda;\lambda_{i+1}) &:= f^{L,I_i}_D(\lambda_i) + {g^L_{D,i}}^{\mathrm{T}}(\lambda - \lambda_{i+1}) \\ \hat{f}^L_{D,i+1}(\lambda) &:= \min_{\bar{f}_D \in J_{D,i} \cup \{\bar{f}^{L,I_i}_D(\lambda;\lambda_{i+1})\}} \bar{f}_D(\lambda). \end{aligned}$$

Since each linearization is computed with respect to a subset of duties I_j, it is in general not true that $\bar{f}^{L,I_j}_D \geq f^{I_i}_D$ if $I_i \neq I_j$. It can (and does) therefore happen that we notice that the current iterate is cut off by some previously computed linearization, i.e.,

$$f^{L,I_i}_D(\lambda_{i+1}) > \bar{f}^{L,I_j}_D(\lambda_{i+1};\lambda_j)$$

for some $j \leq i$. In this case, we have detected an error made in a previous iteration and simply remove the faulty elements from the bundle and also from the approximation. The duty scheduling bundle update in step 5 of Algorithm 1 is implemented as

$$J_{D,i+i} \subset \begin{cases} \{\bar{f}_D \in J_{D,i} : f^{L,I_i}_D(\lambda_{i+1}) \leq \bar{f}_D(\lambda_{i+1})\} \\ \cup \{\bar{f}^{L,I_i}_D(\cdot;\lambda_{i+1}), \tilde{f}_{V,i}\}, & \text{if } f^{L,I_i}_D(\lambda_{i+1}) < \hat{f}^L_{D,i}(\lambda_{i+1}), \\ J_{D,i}, & \text{otherwise.} \end{cases} \tag{2}$$

This approximation must also be recomputed at the stability center in every iteration.

Require: Starting point $\lambda_0 \in \mathbb{R}^n$, duty set I_0, weights $u_0, m > 0$, optimality tolerance $\epsilon \geq 0$.

1: Initialization: $i \leftarrow 0$, $J_{V,i} \leftarrow \{\lambda_i\}$, $J_{D,i} \leftarrow \{\lambda_i\}$, and $\mu_i = \lambda_i$.
2: Direction Finding: Compute λ_{i+1}, $\tilde{g}^L_{V,i}$, $\tilde{g}^L_{D,i}$ by solving problem (QP_i).
3: Function evaluation: Compute $f^L_V(\lambda_{i+1})$, $g^L_V(\lambda_{i+1})$, I_i, $f^{L,I_i}_D(\lambda_{i+1})$, $g^{L,I_i}_D(\lambda_{i+1})$.
4: Stopping Criterion: If $\hat{f}^L_i(\lambda_{i+1}) - f^{L,I_i}(\mu_i) < \epsilon(1 + \left|f^{L,I_i}(\mu_i)\right|)$ output μ_i, terminate.
5: Bundle Update: Select $J_{V,i+1}$, $J_{D,i+1}$ as stated in 1, 2.
6: Ascent Test: $\mu_{i+1} \leftarrow f^{L,I_i}(\lambda_{i+1}) - f^{L,I_i}(\mu_i) > m(\hat{f}^{L,I_i}_i(\lambda_{i+1}) - f^{L,I_i}(\mu_i))$? λ_{i+1} : μ_i.
7: Weight Update: Set u_{i+1}.
8: $i \leftarrow i + 1$, goto step 2.

Algorithm 2: Inexact Proximal Bundle Method (PBM) with Column Generation.

Combined Function f^I. The combined approximate functions are

$$\begin{aligned} f^{L,I_i} &:= f^L_V + f^{L,I_i}_D \\ \hat{f}^L_i &:= \hat{f}^L_{V,i} + \hat{f}^L_{D,i}. \end{aligned}$$

Column generation. This is the most time consuming part of our algorithm, and we therefore enter this phase only if significant progress can be expected. Details about the column generation itself can be found in Borndörfer, Grötschel & Löbel (2003) [2]. Our strategy when to generate new columns is basically to recompute the duty set when the stability center changes; we call such an iteration a *serious step*, all other iterations are called *null steps*.

The reasoning behind this strategy is as follows. The quadratic penalty term in the quadratic program QP_i ensures that the next trial value for the dual multipliers λ_{i+1} stays in the vicinity of the current stability center. When the multipliers change only little, one has reason to believe that the number and the potential effect of improving duties is also small. We therefore hope that the current duty set I_i, which has been updated when the stability center was set, does still provide a good representation of the duty space also for the new multipliers λ_{i+1}. In practice, we reduce the number of column generation phases even further by requiring a certain minimum increase ε in the objective function at the new stability center; the larger ε, the less column generation phases will occur.

Algorithm 2 gives a listing of our bundle algorithm using inexact evaluations

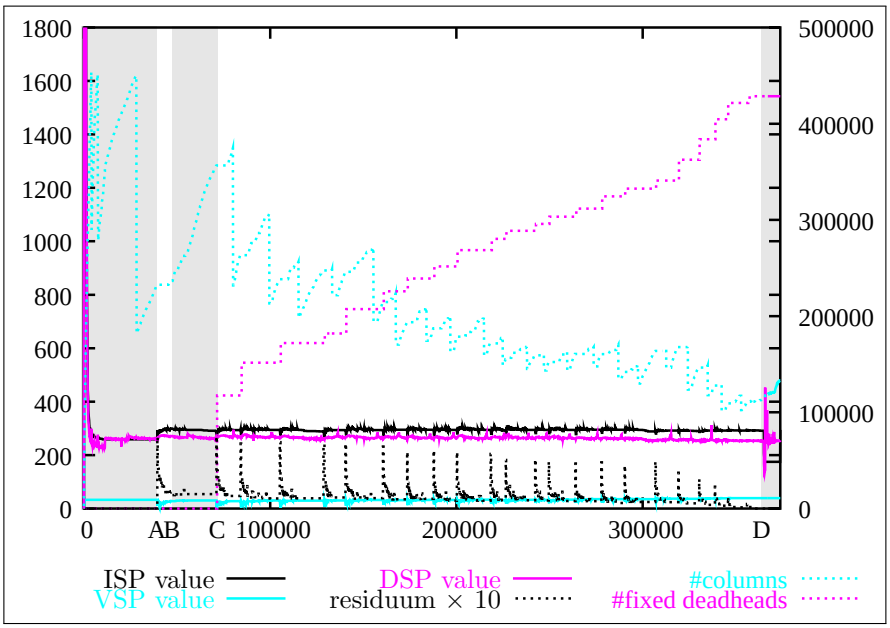


Figure 1: IS-OPT Runtime Chart.

of the component functions and column generation in the duty scheduling component.

4.4 Backtracking Procedure

The inexact proximal bundle method that we have described in this section is embedded in a backtracking procedure that aims at the generation of integer solutions. This procedure makes use of the primal information produced by the bundle method, namely, the sequence $(\tilde{y}_i, \tilde{x}_i)$. As in an LP-approach, fractional values can be interpreted as probabilities for the inclusion/exclusion of a deadhead trip or duty in an optimal integer solution.

Our computational experiments revealed that it is advantageous to fix the deadhead trips first, until the vehicle scheduling part of the problem is decided. The remaining duty scheduling problem can then be solved with the duty scheduling module of the algorithm as described in Borndörfer, Grötschel & Löbel (2003) [2]. Our strategy for fixing the deadhead variables is to fix the deadheads in the order of largest y-values. Our algorithm also examines the consequences of such fixings and, if the increase in the objective function is too large, also reverses decisions. The details on how many variables to fix at a time, up to which threshold, etc. have been determined experimentally; in general, the algorithm fixes more boldly in the beginning and more carefully towards the end.

Figure 1 shows a typical runtime chart of our algorithm IS-OPT. The x-axis

measures time in seconds, the y-axis gives statistics in two different scales, namely, for the right scale, the number of duties generated (#columns), the number of deadheads fixed to one (#fixed deadheads), and the residuum of the coupling constraints (more precisely: the norm is the square of the Euclidean norm of $\tilde{g}_i$), as well as, for the left scale, the vehicle, duty, and the integrated scheduling objective values. Here the duty scheduling value is the lower bound of the restricted DSP calculated by the PBM, and integrated scheduling objective value is simply the sum of the VSP and the DSP value.

In the first phase of the algorithm until point A a starting set of columns was generated with Lagrangean multipliers λ all at zero. In principle the DSP objective value should be strictly decreasing here, while the number of columns should grow. However, we calculated in this initial phase only rough lower bounds for the restricted DSP, which may be more or less accurate. Additionally we deleted columns with large reduced cost if the total number of columns exceeded 450,000. Between points A and B, a series of null steps was performed, which resulted in a decreased norm and an increased ISP-value. Between points B and C, column generation phases alternated with PBM-steps, until an aggregated subgradient of small norm and thus also a "good" primal approximation of the LP-relaxation of ISP was calculated. Since the column generation process did not find enough improving columns at this point, we used the computed information to fix deadheads until (at point D) the vehicle scheduling part of the problem was completely decided. At that point, the duty scheduling component of the algorithm concluded by computing a feasible duty scheduling.

Serious steps of the PBM are marked by peaks of the norm statistic. This effect is due to the shift of the stability center in combination with the possible inclusion of additional columns in I_i. In fact, the new stability center may lie in a region where the model $\hat{f}^{L,I_i}$ of the previous iteration i is less accurate; also, new columns in I_i change the function f^{L,I_i}, what also worsens the model.

In our computational tests the algorithm rarely had to reverse a fixing decision for a deadhead and backtrack. In all our instances, the ISP objective value is very stable with respect to careful fixings of deadheads, see also Figure 1. In fact, the gap between our estimated lower bound, i.e., the objective value prior to the first fixings, and the final objective value was never larger than 5% and only 1-2% on the average. We do, however, not know the size of the gap between the estimated lower bound and the real minimum of (ISP); the mentioned behavior is therefore only a weak indicator for the quality of the final solution found by IS-OPT.

5 Computational Results

In this section, we report the results of computational studies with our integrated vehicle and duty scheduling optimizer IS-OPT for several medium- and large-scale real-world scenarios as well as for benchmark scenarios from the literature. Our code IS-OPT is implemented in C and has been compiled using `gcc` version 3.3.3 with switches `-O4`. All computations were made single-threaded on a Dell Precision 650 PC with 4 GB of main memory and a dual Intel Xeon 3.0 GHz CPU running SuSE Linux 9.0. The computation times in the following tables are in hours:minutes.

We compare our integrated scheduling method *is* with two sequential approaches. The first one, denoted by $v+d$, is a classical sequential vehicles-first duties-second approach, i.e., $v+d$ first solves the vehicle scheduling part of the problem using our optimizer VS-OPT (see Löbel (1998) [24]), fixes the deadheads chosen by the vehicle schedule, and solves the resulting duty scheduling problem in a second step using our optimizer DS-OPT (see Borndörfer, Grötschel & Löbel (2003) [2]). The second method $d+v$ uses kind of the contrary approach. A simplified integrated scheduling problem is set up that identifies drivers and vehicles, i.e., vehicle changes outside of the depot are forbidden. This "poor man's integrated scheduling model" is solved using the duty scheduling algorithm DS-OPT. The vehicle rotations resulting from this duty schedule are concatenated into daily blocks using the vehicle scheduling algorithm VS-OPT in a second step.

We calibrated the parameters of the bundle method, namely m and the series $(u_i)_{i=1,2,\ldots}$, such that about 20% of the iterations were serious steps. We never needed more than 50 iterations of the bundle method before the first fixing of variables.

5.1 RVB Instances

The Regensburger Verkehrsbetriebe GmbH (RVB) is a medium sized public transportation company in Germany. We consider two instances that contain the entire RVB operation for a Sunday and for a workday. The structure of the RVB data is mostly urban with only four relief points. In fact, the network of the RVB is mostly star-shaped with nearly all lines meeting in a small area around the main railway station. Only there, at two stations nearby, and at the also nearby garage the drivers can change buses and begin or end duties. The RVB uses only one type of vehicles on Sundays, and three types on workdays, i.e., the Sunday scenario is fleet homogenous, while the workday scenario is a multi-depot problem. The vehicle types can only be used on trips on certain sets of (non-disjoint) lines. The Sunday scenario

	Sunday	workday
vehicle types	1	3
timetabled trips	794	1414
tasks on tt	1248	3666
deadhead trips	47523	57646
duty types	3	4
break rules	4	4

Table 1: Statistics on the RVB Instances.

involves three different types of early, mid, and late duties, each with four different types of break rules. In Germany, detailed legal regulations exist about the number, the length, and the feasible positions of breaks in a duty. These regulations may also differ from one company to the other by works council agreements. We use in the RVB instances block breaks of 1×30, 2×20, and 3×15 minutes plus 1/6-quotient breaks. The most important regulations valid for all these break rules are: There is no interval without break with more than six hours working time. There is no interval without break with more than four and a half hours driving time. Between two breaks is at least half an hour working time. A duty fulfills the 1/6-quotient break rule if every continuous segment of a duty contains at least a sixth part break time, and every break must be at least eight minutes.

The workday scenario contains in addition a type of split duties, again with the mentioned break rules per part of work. Table 1 reports further statistics on the number of timetabled trips, tasks, and deadhead trips (also equal to the number of Lagrangean multipliers). The Sunday scenario is medium-sized, while the workday scenario is, as far as we known, the largest and most complex instance that has been attacked with integrated scheduling techniques.

Table 2 gives computational results for the Sunday scenario. The column 'reference' lists statistics for the solution that RVB planners had generated by hand. The next four columns give the results of two sequential v+d-optimizations and two integrated *is*-optimizations; we do not report results for the method d+v, because we could not produce a feasible solution for this scenario with this method. The objective function consists of a weighted sum of the number of duties, the number of pieces of work, the paid time of the duty schedule, and penalties for exceeding an average duty time. A *piece of work* is an inclusion-maximal continuous segment of a duty where a driver does not change the vehicle. Changes of vehicles should be avoided because they may lead to operational problems in case of delays of vehicles.

In the optimization runs "v+d 2" and "*is* 2", emphasis was placed on the minimization of the number of duties, while runs "v+d 1" and "*is* 1" tried

	reference	*v+d* 1	*v+d* 2	*is* 1	*is* 2
time on vehicles	518:33	472:12	472:12	501:42	512:55
paid time	545:25	562:58	565:28	518:03	531:31
paid break time	112:36	131:40	85:41	74:17	64:27
number of duties(slacks)	82	83	74(1)	76	66
number of vehicles	36	32	32	32	35
average duty duration	6:39	6:48	7:38	6:40	8:03
computation time	—	0:33	5:13	35:44	37:26

Table 2: Results for the RVB Sunday Scenario.

to reproduce the average duty time of the reference solution.

As expected the sequential methods reduce the number of vehicles and the time on vehicle rotations since these are the primary optimization objectives. Also they produce quite reasonable results in terms of duty scheduling. "*v+d* 1" suffers from a slight increase in duties and paid time, "*v+d* 2" yields substantial savings in duties; however the price for this reduction is a raised average paid time. Also one task was not covered by duties in the solution (remarked by the one in brackets). Even better are the results of the integrated optimizations. "*is* 1" is perfect with respect to any statistic and produces *large* savings. These stem from the use of short duties involving less than 4:30 hours of driving time, which don't need a break; this potential improvement of the Sunday schedule is one of the most significant results of this optimization project for the RVB. Even more interesting is solution "*is* 2". This solution trades three vehicles and an increased average for another 10 duties; as longer duties must have breaks, the paid time (breaks are paid here) increases as well. Solution "*is* 2" revived a discussion at the RVB whether drivers prefer to have less, but longer duties on weekends or whether they want to stay with more, but short duties.

Table 3 lists the results of the workday optimizations. Method *d+v* could again not produce a feasible solution and is therefore omitted from the table. The objective in this scenario is far from obvious; it is given as a complicated mix of fixed and variable vehicle costs, fixed costs and paid time for duties, and various penalties for several pieces of work, split duties, etc., that can compensate each other such that one cannot really compare the solutions by means of single statistics. Doing it nevertheless, we see that both optimization approaches clearly improve the reference solution substantially. The outcome is close. In fact, *v+d* has less paid time than *is*; in the end, however, *is* is better in terms of the composite objective function.

	reference	*v+d*	*is*
time on vehicles	1037:18	960:29	1004:27
paid time	1103:48	1032:20	1040:11
granted break time	211:53	109:11	105:23
number of duties	140	137	137
number of vehicles	91	80	82
number of pieces of work	217	290	217
number of split duties	29	39	36
average duty duration	7:56	8:03	7:55
obj. value	—	302.32	291.16
computation time	—	8:02	125:55

Table 3: Results for the RVB Workday Scenario.

5.2 RKH Instances

The Regionalverkehrsbetrieb Kurhessen (RKH) is a regional carrier in the middle of Germany. They provided data for the subnetworks of Marburg and Fulda which is not (yet) in industrial use; some deadheads are missing, while for some others travel times have only been estimated by means of distance calculations. In our opinion the data still captures to a large degree the structure of a regional carrier and we therefore deem it worthwhile to report the results of the conceptual study that we did with it.

Figure 2 shows the spatial structure of the line network of Fulda, which is one part of the RKH service area. The black arcs denote the timetabled trips (drawn straight from the line's start to the end), the gray arcs indicate the potential deadhead trips. It can be seen that the trip network is hub-and-spoke-like, connecting several cities and villages among themselves and with the rural regions around them. While the deadhead network is almost complete, there are only a few relief opportunities for drivers to leave or enter a vehicle.

Table 4 gives further statistics on the RKH instances. They are similar to the RVB Sunday scenario in terms of timetabled trips and tasks, but contain much more deadhead trips. The scenarios involve three duty types, two types of split duties that differ in the maximum duty length and one type of continuous duties. Each duty type can have 1×30, 2×20, or 3×15 minutes block breaks or 1/6-quotient breaks.

Table 5 reports the results of our optimizations. We do not report results for the method *v+d* as we were not able to produce a feasible solution for either scenario with this method. Method *d+v* yields useful results, but it is not able to cover all tasks/trips of the Fulda-scenario with duties and vehicles; in fact, *d+v* left 3 tasks and 6 timetabled trips uncovered (numbers

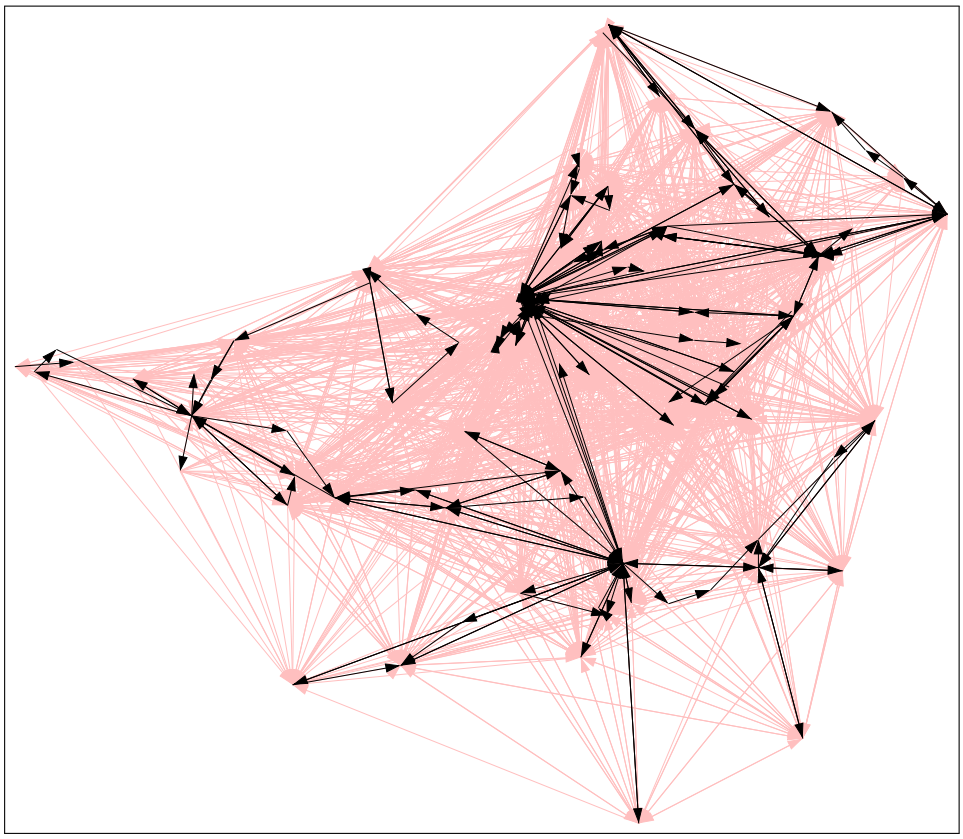


Figure 2: Scheduling Graph for Scenario *Fulda*.

in parentheses). These deficiencies are resolved in the *is*-solutions, which also look better in terms of numbers of vehicles.

5.3 ECOPT Instances

Finally, we compare IS-OPT with the approach of Huisman et. al. on the randomly generated benchmark data proposed in their article Huisman, Freling & Wagelmans (2005) [20]. These data consist of two sets of instances involving two and four depots, respectively. Each set contains 10 instances of 80,

	Marburg	Fulda
depots	3	1
vehicle types	5	1
timetabled trips	634	413
tasks on TT	1022	705
deadhead trips	142,668	67,287

Table 4: RKH Scenarios *Marburg* and *Fulda*.

	Marburg		Fulda	
	d+v	*is*	*d+v*	*is*
time on vehicles	772:02	642:41	365:41	387:37
paid time	620:27	606:30	390:08	374:53
granted break time	120:51	103:27	88:13	57:44
number of duties	73	70	41(3)	41
number of vehicles	62	50	45(6)	37
average duty duration	10:35	10:18	10:59	11:18
computation time	5:29	17:18	1:42	7:05

Table 5: Solutions for Scenarios *Marburg* and *Fulda*.

100, 160, 200, 320, and 400 trips, see again Huisman, Freling & Wagelmans (2005) [20] for a detailed description. The duty scheduling rules associated with these examples are relatively simple. Duties are allowed to have at most one break, which must be outside of a vehicle, i.e., each break also begins a new piece of work. The only other rule is that each piece of work must be of certain minimum and maximum length. It is shown in Huisman, Freling & Wagelmans (2005) [20] that in this situation one can solve the duty generation subproblem in polynomial time, i.e., exact column generation is applicable.

Tables 6 and 7 report average solution values for each of the 10 instances of each problem class for the problem variant A; similar results for variant B have been omitted. All computations were done with the same set of parameters, which was optimized for speed. Row *reference* gives the sum of the numbers of vehicles and duties as published in Huisman, Freling & Wagelmans (2005) [20]; for the problems with 4 depots and 320 and 400 trips, no reference is given due to excessive computation time.

It can be seen that our algorithm IS-OPT performs worse than that in Huisman, Freling & Wagelmans (2005) [20] for the small instances, but produces better results with increasing problem size and complexity; it can also solve the largest problem instances. We remark that IS-OPT can also produce slightly better solutions for the small instances than those reported

trips	080	10 0	160	200	320	400
vehicles	9.4	11.2	15.0	18.6	27.0	33.3
duties	21.2	25.1	33.9	40.6	57.7	69.8
total	30.6	36.3	48.9	59.2	84.7	103.1
reference	29.8	35.6	48.3	59.1	86.8	106.1
time	00:05	00:08	00:17	00:31	01:58	03:19

Table 6: Results for ECOPT-Instances with 2 Depots Variant A.

trips	080	100	160	200	320	400
vehicles	9.2	11.2	15.0	18.5	26.7	33.1
duties	20.4	24.5	32.7	40.5	56.1	68.9
total	29.6	35.7	47.7	59.0	82.8	102.0
reference	29.6	36.2	49.5	60.4	—	—
time	00:13	00:21	00:44	01:46	05:28	12:00

Table 7: Results for ECOPT-Instances with 4 Depots Variant A.

in Huisman, Freling & Wagelmans (2005) [20] by changing the optimality parameter ϵ in Algorithm 2 and by raising the threshold for deadhead fixes. This leads, of course, to longer computation times.

6 Conclusions

We have shown that it is possible to tackle large-scale, complex, real-world integrated vehicle and duty scheduling problems using a novel "bundle" algorithm for integrated vehicle and duty scheduling. The solutions produced by such an integrated approach can be decidedly better *in several respects at once* than the results of various types of sequential planning.

References

[1] M. O. Ball, L. Bodin & R. Dial. *A matching based heuristic for scheduling mass transit crews and vehicles.* Transportation Sci. **17**: 4–31, 1983. Cited on page 93.

[2] R. Borndörfer, M. Grötschel & A. Löbel. *Duty scheduling in public transit.* In W. Jäger & H.-J. Krebs, (Eds.), *MATHEMATICS – Key Technology for the Future*, pp. 653–674. Springer Verlag, Berlin, 2003. ISBN 3-540-44220-0. ZIB Report 01-02 available at http://opus.kobv.de/zib/volltexte/2001/629/. Cited on pages 95, 98, 103, 104, 106, 107, 109.

[3] J. R. Daduna & M. Völker. *Fahrzeugumlaufbildung im ÖPNV mit unscharfen Abfahrtszeiten.* Der Nahverkehr **11**:39–43, 1997. Cited on page 93.

[4] J. R. Daduna & A. Wren, (Eds.). *Computer-Aided Transit Scheduling*, vol. 308 of *Lecture Notes in Economics and Mathematical Systems*, 1988. Springer Verlag, Berlin. ISBN 3-540-19441-X. Cited on pages 93, 116.

[5] J. R. Daduna, I. Branco & J. M. P. Paixão, (Eds.). *Computer-Aided Transit Scheduling*, vol. 430 of *Lecture Notes in Economics and Mathematical Systems*, 1995. Springer Verlag, Berlin. ISBN 3-540-60193-7. Cited on page 93.

[6] K. Darby-Dowman, R. L. L. J. K. Jachnik & G. Mitra. *Integrated decision support systems for urban transport scheduling: Discussion of implementation and experience.* In Daduna & Wren (1988) [4], pp. 226–239. ISBN 3-540-19441-X. Cited on page 94.

[7] M. Desrochers & J.-M. Rousseau, (Eds.). *Computer-Aided Transit Scheduling*, vol. 386 of *Lecture Notes in Economics and Mathematical Systems*, 1992. Springer Verlag, Berlin. ISBN 3-540-55634-6. Cited on pages 93, 116.

[8] M. Desrochers & F. Soumis. *A column generation approach to the urban transit crew scheduling problem.* Transportation Sci. **23**(1):1–13, 1989. Cited on page 97.

[9] J. C. Falkner & D. M. Ryan. *EXPRESS: Set partitioning for bus crew scheduling in Christchurch.* In Desrochers & Rousseau (1992) [7], pp. 359–378. ISBN 3-540-55634-6. Cited on page 94.

[10] R. Freling. *Models and Techniques for Integrating Vehicle and Crew Scheduling.* PhD thesis, Erasmus Universiteit Rotterdam, 1997. Cited on pages 94, 98.

[11] R. Freling, D. Huisman & A. P. M. Wagelmans. *Applying an integral approach to vehicle and crew scheduling in practice.* In Voß & Daduna (2001) [28], pp. 73–90. ISBN 3-540-42243-9. Cited on page 94.

[12] R. Freling, A. P. M. Wagelmans & J. M. P. Paixao. *Models and algorithms for single-depot vehicle scheduling.* Transportation Sci. **35**: 165–180, 2001. Cited on page 94.

[13] R. Freling, D. Huisman & A. P. M. Wagelmans. *Models and algorithms for integration of vehicle and crew scheduling.* J. Scheduling **6**: 63–85, 2003. Cited on page 94.

[14] C. Friberg & K. Haase. *An exact algorithm for the vehicle and crew scheduling problem.* In Wilson (1997) [29], pp. 63–80. ISBN 3-540-65775-4. Cited on page 94.

[15] A. Gaffi & M. Nonato. *An integrated approach to ex-urban crew and vehicle scheduling.* In Wilson (1997) [29], pp. 103–128. ISBN 3-540-65775-4. Cited on page 94.

[16] K. Haase, G. Desaulniers & J. Desrosiers. *Simultaneous vehicle and crew scheduling in urban mass transit systems.* Transportation Sci. **35**(3):286–303, August 2001. Cited on page 94.

[17] J. Hanisch. *Die Regionalverkehr Köln GmbH und HASTUS.* Webpage, 1990. URL http://www.giro.ca/Deutsch/Publications/publications.htm. Cited on page 93.

[18] C. Helmberg. *Semidefinite programming for combinatorial optimization*. Habilitation thesis, Technische Universität Berlin, 2000. ZIB Report 00-34 available at http://opus.kobv.de/zib/volltexte/2000/603/. Cited on page 102.

[19] M. Hintermüller. *A proximal bundle method based on approximate subgradients*. Comput. Optim. Appl. **20**:245–266, 2001. Cited on page 104.

[20] D. Huisman, R. Freling & A. P. M. Wagelmans. *Multiple-depot integrated vehicle and crew scheduling*. Transportation Sci. **39**(4):491–502, 2005. ISSN 1526-5447. Cited on pages 113, 114, 115.

[21] K. C. Kiwiel. *Proximity control in bundle methods for convex nondifferentiable minimization*. Math. Programming **46**:105–122, 1990. Cited on page 100.

[22] K. C. Kiwiel. *Approximation in proximal bundle methods and decomposition of convex programs*. J. Optim. Theory Appl. **84**(3):529–548, 1995. Cited on pages 98, 100, 103.

[23] C. Lemaréchal. *Lagrangian relaxation*. In M. Jünger & D. Naddef, (Eds.), *Computational Combinatorial Optimization*, vol. 2241 of *Lecture Notes in Computer Science*, pp. 112–156. Springer Verlag, Berlin, 2001. ISBN 3-540-42877-1. Cited on page 98.

[24] A. Löbel. *Optimal Vehicle Scheduling in Public Transit*. Berichte aus der Mathematik. Shaker Verlag, Aachen, 1998. ISBN 978-3-8265-3504-8. URL http://www.shaker.de/Online-Gesamtkatalog/Details.asp?ISBN=978-3-8265-3504-8. PdD thesis, Technische Universität Berlin, 1998. Available at http://www.zib.de/bib/books/Loebel.disser.ps. Cited on pages 95, 98, 103, 109.

[25] I. Patrikalakis & D. Xerocostas. *A new decomposition scheme of the urban public transport scheduling problem*. In Desrochers & Rousseau (1992) [7], pp. 407–425. ISBN 3-540-55634-6. Cited on page 94.

[26] D. Scott. *A large scale linear programming approach to the public transport scheduling and costing problem*. In J.-M. Rousseau, (Ed.), *Computer Scheduling of Public Transport 2*. Elsevier Science B. V., Amsterdam, 1985. Cited on page 94.

[27] E. Tosini & C. Vercellis. *An interactive system for extra-urban vehicle and crew scheduling problems*. In Daduna & Wren (1988) [4], pp. 41–53. ISBN 3-540-19441-X. Cited on page 94.

[28] S. Voss & J. R. Daduna, (Eds.). *Computer-Aided Transit Scheduling*, vol. 505 of *Lecture Notes in Economics and Mathematical Systems*, 2001. Springer Verlag. ISBN 3-540-42243-9. Cited on pages 93, 116.

[29] N. Wilson, (Ed.). *Computer-Aided Transit Scheduling*, vol. 471 of *Lecture Notes in Economics and Mathematical Systems*, 1997. Springer Verlag, Berlin. ISBN 3-540-65775-4. Cited on pages 93, 116.

Paper V

Models for Railway Track Allocation

R. Borndörfer & T. Schlechte.
Models for railway track allocation.
In C. Liebchen, R. K. Ahuja & J. A. Mesa, (Eds.), *Proceeding of the 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS 2007)*, Dagstuhl, Germany, 2007. Internationales Begegbnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

Abstract. The optimal track allocation problem (OPTRA) is to find, in a given railway network, a conflict free set of train routes of maximum value. We study two types of integer programming formulations for this problem: a standard formulation that models block conflicts in terms of packing constraints, and a novel formulation of the 'extended' type that is based on additional 'configuration' variables. The packing constraints in the standard formulation stem from an interval graph and can therefore be separated in polynomial time. It follows that the LP-relaxation of a strong version of this model, including all clique inequalities from block conflicts, can be solved in polynomial time. We prove that the LP-relaxation of the extended formulation can also be solved in polynomial time, and that it produces the same LP-bound. Albeit the two formulations are in this sense equivalent, the extended formulation has advantages from a computational point of view. It features a constant number of rows and is amenable to standard column generation techniques. Results of an empirical model comparison on mesoscopic data for the Hanover-Fulda-Kassel region of the German long distance railway network involving up to 570 trains are reported.

Mathematics Subject Classification (MSC 2000). 90C06, 90B06

Keywords. Railways, track allocation, timetabling, path packing, configuration model, column generation

1 Introduction

Routing trains in a conflict-free way through a network of tracks is one of the basic and at the same time most difficult questions in railway scheduling. The need to coordinate the use of shared infrastructure and the complex operation of this infrastructure using switches and signals impose a great variety of technical constraints, that give rise to a complex problem in which many factors have to be considered simultaneously.

Among the earliest optimization approaches to *track allocation problems* are integer programming formulations that model train routes as paths in appropriate networks. As early as 1956, Charnes & Miller (1956) [5] propose a set covering formulation, in which 'crew and engine packages' are assigned to circular routes in a railway network; the model is solved with what we would call today a column generation procedure. Later, set packing versions of this formulation, which can rule out block conflicts between train routes, have been proposed and studied by a number of authors including Brännlund et al. (1998) [2], Caprara et al. (2001) [3], Caprara, Fischetti & Toth (2002) [4], and Borndörfer et al. (2006) [1]. The main difficulty with this type of formulation is that it contains a very large number of constraints which makes these models computationally hard, if not intractable, beyond a certain size.

We propose in this article a novel formulation for train routing in an attempt to resolve this difficulty. Our formulation is of the 'extended' type; it rules out block conflicts using additional 'configuration' variables. It can be shown that such a model is equivalent to a strong version of the standard packing model (including all clique constraints from 'block conflicts') with respect to both quality and computational complexity of the LP-bound. From a practical point of view, the extended model has the advantage that it is amenable to standard column generation techniques and therefore well suited to solve large-scale problems.

The article is organized as follows. Section 2 gives a formal statement of the optimal track allocation problem. For the sake of clarity of exposition, we concentrate here on a basic version that considers a very simple type of 'block conflicts' between trains. We remark, however, that most our results carry over to more general situations with 'headway conflicts' from 'quadrangle-linear' matrices, see Lukac (2004) [8] for details of such a model. Packing IP-formulations for the track allocation problem are studied in Section 3.1. We show that block conflicts arise from an interval graph, that cliques from block conflicts can be separated in polynomial time, and that the LP-relaxation of a packing model including all such clique constraints can be solved in polynomial time. Section 3.2 introduces our extended formulation. We show that the pricing problem for configuration variables can

symbol	description	symbol	description
S	stations	$G = (S, J)$	infrastructure digraph
J	tracks	$D = (V, A)$	train routing digraph
I	trains	$D_i = (V, A_i)$	individual routing digraph
w	arc weights	s_i, t_i	source, sink of train i

Table 1: Notation for the Optimal Track Allocation Problem (OPTRA).

be solved by computing a longest path in an appropriately defined acyclic digraph, and that the LP-relaxation of the extended model can also be solved in polynomial time. Section 3.3 compares both models analytically; it turns out that they produce the same LP-bound. The final Section 4 contains a computational model comparison on data for the Hanover-Kassel-Fulda part of the long distance network of the German railway company Deutsche Bahn AG with up to 570 trains.

2 The Optimal Track Allocation Problem

The *optimal track allocation problem*, also known as the *train routing problem* or the *train timetabling problem*, can be formally described as follows. We are given a set I of *requests* to route *trains* in a *train routing digraph* $D = (V, A)$; we allow that D contains multiple arcs between two nodes. D is based on an *infrastructure digraph* $G = (S, J)$, whose nodes and arcs model *stations* and *tracks*, respectively. The train routing digraph is a time expansion of the infrastructure digraph, i.e., the nodes of D model possible *departures* and *arrivals* of trains at stations at certain points in time, the arcs possible timetabled *trips* of specific trains. Formally, we associate with each node $v \in V$ a station $s(v) \in S$ and a discrete time $t(v) \in \mathbb{Z}$. An arc $uv \in A$ models a trip on track $s(u)s(v) \in J$ for a train $i(uv) \in I$, which departs at time $t(u)$ and arrives at time $t(v)$; we assume $t(u) < t(v)$ for all trips $uv \in A$ such that D is acyclic. We associate with train $i \in I$ the trips $A_i := \{a \in A : i(a) = i\} \subseteq A$ that this train can run and the *individual train routing digraph* $D_i := (V, A_i) \subseteq D$, which we assume to contain two special (if need be artificially constructed) nodes s_i and t_i, called *source* and *sink*, that represent the departure and the arrival of train i; we therefore assume $\delta_i^-(s_i) = \delta_i^+(t_i) = \emptyset$ (where $\delta_i^\pm(W) := \delta^\pm(W) \cap A_i$, $\forall W \subseteq V$), and denote $W_i := V \setminus \{s_i, t_i\}$. A *route* for train i is an $s_i t_i$-path in D_i. Denote the set of all routes for train i by P_i, and the set of all possible routes by P (let P be the disjoint union of the sets P_i, i.e., we distinguish identical routes for different trains). Figure 1 illustrates this construction.

We say that an arc $uv \in A$ occupies or *blocks* its associated track $s(u)s(v)$ for the time interval $[t(u), t(v)-1]$, and that there is a *block conflict* between two

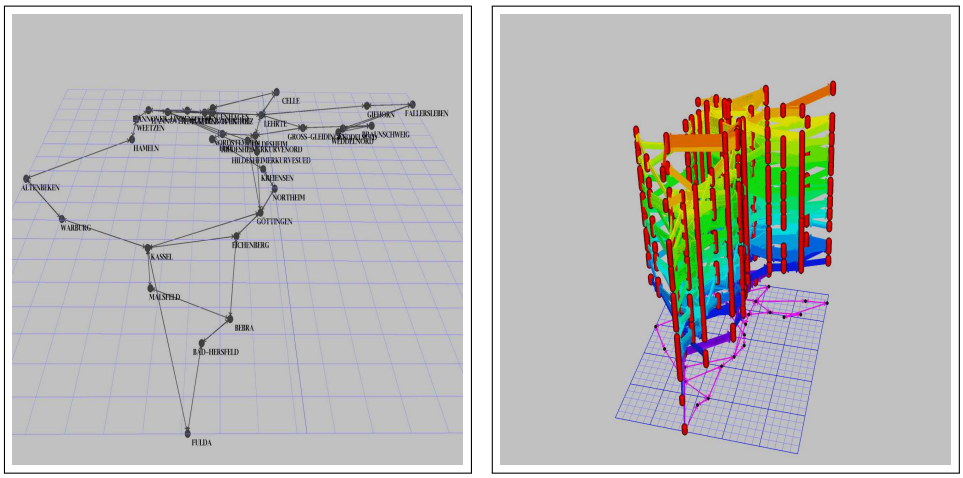


Figure 1: Optimal Track Allocation Problem: Infrastructure Network (left), and Train Routing Digraph (right); Individual Train Routing Digraphs Bear Different Colors.

arcs u_1v_1 and u_2v_2 on the same track if their track occupation time intervals overlap, i.e., if $s(u_1)s(v_1) = s(u_2)s(v_2)$ and $[t(u_1), t(v_1) - 1] \cap [t(u_2), t(v_2) - 1] \neq \emptyset$. There is a block conflict between two train routes if any of their arcs have a block conflict. A timetable or *schedule* is a set $X \subseteq P$ of conflict-free routes, at most one for each train request, i.e., $|X \cap P_i| \leq 1$, $i \in I$. Assigning weights $w_{uv} \in \mathbb{Z}$ to the arcs $uv \in A$ (modeling 'profits' for individual trips), the weight of route $p \in P$ is $w_p := \sum_{a \in p} w_a$, and the weight of a schedule $X \subseteq P$ is $w(X) := \sum_{p \in X} w_p$. The *optimal track allocation problem* (OPTRA) is to find a schedule of maximum weight.

Caprara, Fischetti & Toth (2002) [4] have shown that the stable set problem can be reduced to OPTRA, such that the problem is $\mathcal{NP}$-hard. Indeed, OPTRA can be seen as a problem to find a maximum weight packing (with respect to block conflicts) of train routes in a time-expanded digraph. This framework is fairly general, see the articles of Caprara et al. (2001) [3], Caprara, Fischetti & Toth (2002) [4], and Borndörfer et al. (2006) [1] for comprehensive discussions how such a model can be used to deal with various kinds of technical constraints.

There is, however, one point where our exposition resorts to a genuine simplification, namely, by considering only block conflicts arising from time overlaps. Such a model obviously ignores important aspects such as different block occupation times for the head and the tail of a train, safety margins to open and close a block after a train has left a track and before it can enter, different driving times of trains (a fast train following a slow train needs a larger safety margin than a slow train following a fast train) etc. Such considerations can be expressed in terms of 'headway matrices' that

specify a minimum time difference between every ordered pair of trains on each track, see Lukac (2004) [8] for a discussion of such a model. One can show that most of the results of the following sections carry over to more general situations of this type. We do, however, not give the details here, because they would result in a more technical and complicated discussion.

3 Integer Programming Models

3.1 Packing Models

The standard formulation for the track allocation problem models train routes as a multi-commodity flow and rules out block conflicts using additional packing constraints. We need the following additional terminology. Let $B = \{\{a, b\} \in 2^A : a \neq b \text{ have a block conflict}\}$ be the set of all block conflicts between any two arcs, $H = (A, B)$ the associated (undirected) *(block) conflict graph*, and $C = C(H)$ be the set of all (inclusion) maximal *cliques* in H; Figure 2 illustrates the construction of a block conflict graph for a single track.

The packing model comes in two versions, one with 0/1 arc variables x_a, $a \in A$, for the use of trip a in a route, and the other with 0/1 path variables x_p, $p \in P$, for the use of route p. The resulting formulations, we call them *arc packing problem* (APP) and *path packing problem* (PPP), read as follows:

$$\text{(APP)} \quad \max \sum_{a \in A} w_a x_a$$

$$\text{(i)} \quad \sum_{a \in \delta_i^+(v)} x_a - \sum_{a \in \delta_i^-(v)} x_a = 0 \quad \forall i \in I, v \in W_i$$

$$\text{(ii)} \quad \sum_{a \in \delta_i^+(s_i)} x_a \leq 1 \quad \forall i \in I$$

$$\text{(iii)} \quad \sum_{a \in c} x_a \leq 1 \quad \forall c \in C$$

$$\text{(iv)} \quad x_a \geq 0 \quad \forall a \in A$$

$$\text{(v)} \quad x_a \in \mathbb{Z} \quad \forall a \in A$$

$$\text{(PPP)} \quad \max \sum_{p \in P} w_p x_p$$

$$\text{(ii)} \quad \sum_{p \in P_i} x_p \leq 1 \quad \forall i \in I$$

$$\text{(iii)} \quad \sum_{p \cap c \neq \emptyset} x_p \leq 1 \quad \forall c \in C$$

$$\text{(iv)} \quad x_p \geq 0 \quad \forall p \in P$$

$$\text{(v)} \quad x_p \in \mathbb{Z} \quad \forall p \in P.$$

Equalities (APP) (i) are *flow conservation constraints*; they route train i on $s_i t_i$-paths; note that D_i is acyclic such that no cycles can come up. Constraints (APP)/(PPP) (ii) ensure a train is routed at most once. The *clique inequalities* (APP)/(PPP) (iii) rule out block conflicts. Finally, (APP)/(PPP) (iv) and (v) are the *nonnegativity* and the *integrality constraints*. Note that all constaints together imply that all variables are 0/1.

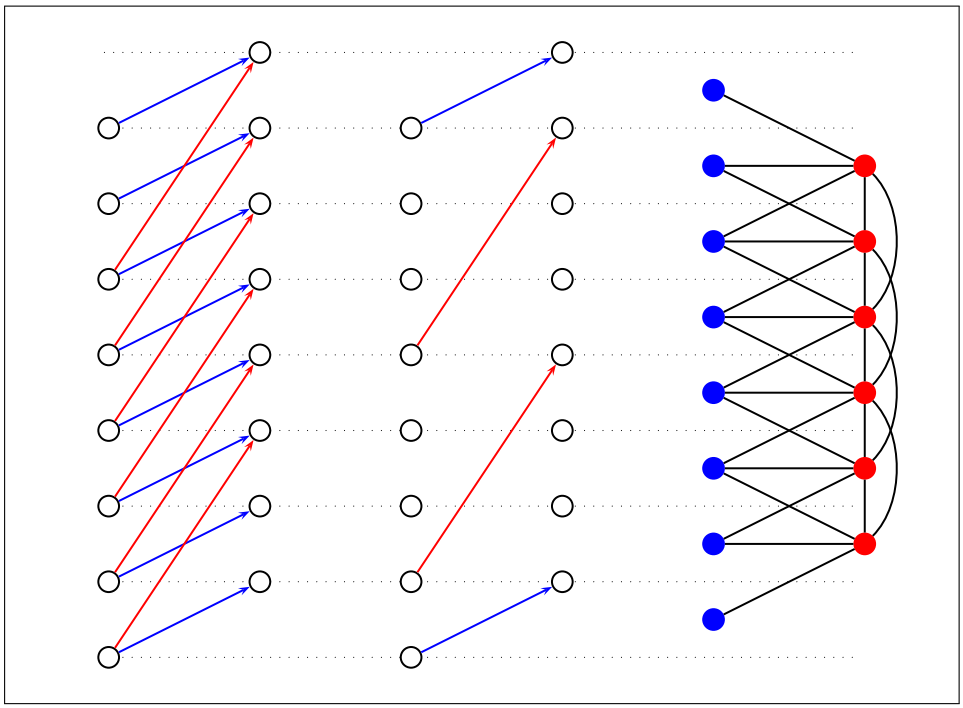


Figure 2: Block Conflicts on a Single Track: Trips for a Slow (blue) and a Fast (red) Train (left), a Conflict-Free Configuration of Four Trips on this Track (middle), and the Block Conflict Graph Associated With the Track (right).

The formulations (APP) and (PPP) are strong in the sense that they include all clique constraints from block conflicts. The literature usually considers models that replace (APP)/(PPP) (iii) by weaker constraints

$$\text{(iii}') \qquad x_a + x_b \leq 1 \quad \forall ab \in B \qquad \text{(iii}') \quad \sum_{p \cap \{a,b\} \neq \emptyset} x_p \leq 1 \quad \forall ab \in B$$

that rule out block conflicts on pairs of arcs; let us denote these variants by (APP$'$) and (PPP$'$). Here are some basic properties of the packing models. By definition:

Observationdd 3.1. *The block conflict graph $H = (A, B)$ that is associated with an optimal track allocation problem is an interval graph.*

The cliques in the conflict graph are collections of compact real intervals. By Helly's Theorem, see Helly (1923) [7], the intervals of each such clique $c \in C$ constain a common point $t(c)$, and it is easy to see that we can assume $t(c) \in t(V) = \{t(v) : v \in V\}$. It follows that the block conflict graph H has $O(V)$ inclusion maximal cliques, which can be enumerated in polynomial time, and that the packing formulations of the optimal track allocation problem

formulation	variables	non-trivial constraints
APP	$O(A)$	$O(A)$
PPP	$O(P)$	$O(V)$
APP′	$O(A)$	$O(A^2)$
PPP′	$O(P)$	$O(A^2)$

Table 2: Sizes of Packing Formulations for the Track Allocation Problem.

have the sizes listed in Table 2; here, $O(I \times V) + O(I) + O(C) = O(A)$, and we write $O(A) = O(|A|)$ etc.

The LP-relaxation of (APP) can then be solved in polynomial time. To obtain the same result for (PPP), consider a column generation approach. Note that no two arcs in a route are in conflict, i.e., $p \cap c \leq 1$ for all routes $p \in P$ and all cliques $c \in C$. Introducing dual variables γ_i, $i \in I$, for the constraints (PPP) (ii), and η_c, $c \in C$, for the constraints (PPP) (iii), the pricing problem for a route $p \in P_i$, for some train $i \in I$, is

$$\exists\, p \in P_i : \gamma_i + \sum_{p \cap c \neq \emptyset} \eta_c < w_p \iff \sum_{a \in p} (w_a - \sum_{c \ni a} \eta_c) > \gamma_i.$$

This is a longest $s_i t_i$-path problem in the acyclic digraph $D_i = (V, A_i)$ w.r.t. arc weights $w_a - \sum_{a \in c} \eta_c$; this problem can be solved in polynomial time (in fact, in linear time). By the polynomial equivalence of separation and optimization, see Grötschel, Lovász & Schrijver (1988) [6], here applied to the dual of (PPP), i.e., the polynomial equivalence of pricing and optimization, we obtain the desired result.

Theoremdd 3.2. *The LP-relaxations associated with the strong arc packing formulation APP and the strong path packing formulation PPP of the optimal track allocation problem can be solved in polynomial time.*

3.2 Extended Models

We propose in this section an alternative formulation for the optimal track allocation problem that guarantees a conflict free routing by allowing only feasible route combinations, and not by excluding conflicts. The formulation is based on the concept of feasible arc *configurations* , i.e., sets of arcs on a track without block conflicts. Formally, we define a configuration for some track $j = xy \in J$ as a set of arcs $q \subseteq A_j := \{uv \in A : s(u)s(v) = xy\}$ such that

$$|q \cap c| \leq 1 \quad \forall c \in C.$$

Denote by Q_j the set of all such configurations for track j, $j \in J$, and by Q the set of all such configurations. The idea of the extended model is to

introduce 0/1 variables y_q for choosing a configuration on each track and to force a conflict free routing of trains through these configurations by means of inequalities

$$\sum_{p \ni a} x_p \leq \sum_{q \ni a} y_q \quad \forall a \in A.$$

Instead of directly writing down a corresponding model, however, we propose a version that will model configurations as paths in a certain acyclic routing digraph. The advantages of such a formulation will become clear in a minute. The construction extends the routing digraph $D = (V, A)$ to a larger digraph $\overline{D} = (\overline{V}, \overline{A})$ by adding nodes and arcs as illustrated in Figure 3. The details are as follows. Consider a track $xy \in J$ and the trips $A_{xy} = \{uv \in A : s(u)s(v) = xy\}$ on this track. Denote by $L_{xy} := \{u : uv \in A_{xy}\}$ and $R_{xy} := \{v : uv \in A_{xy}\}$ the associated set of departure and arrival nodes. Construct two new, artificial nodes s_{xy} and t_{xy} by setting $s(s_{xy}) = y$, $t(s_{xy}) := \min t(R_{xy}) - 1$, and $s(t_{xy}) = x$, $t(t_{xy}) := \max t(R_{xy}) + 1$, i.e., s_{xy} marks an event at station y before the departure of the earliest trip on xy, and t_{xy} marks an event at station x after the arrival of the latest trip on xy. Let $\overline{L}_{xy} := L_{xy} \cup \{t_{xy}\}$ and $\overline{R}_{xy} := R_{xy} \cup \{s_{xy}\}$; note that all arcs in A_{xy} go from $\overline{L}_{xy}$ to $\overline{R}_{xy}$ (actually from L_{xy} to R_{xy}). Now let $\overline{A}_{xy} := \{vu : t(v) \leq t(u), v \in \overline{R}_{st}, u \in \overline{L}_{st}\}$ be a set of 'return' arcs that go in the opposite direction; they connect the arrival of a trip on xy (or node s_{xy}) with all possible follow-on trips (or node t_{xy}) on that track. It is easy to see that the *configuration routing digraph* $\overline{D}_{xy} := (\overline{L}_{xy} \cup \overline{R}_{xy}, A_{xy} \cup \overline{A}_{xy})$ is bipartite and acyclic, and that $s_{xy}t_{xy}$-paths $a_1, \overline{a}_1, \ldots, \overline{a}_{k-1}, a_k$ in $\overline{D}_{xy}$ and configurations $a_1, \ldots, a_k$ in Q_{st} are in 1-1 correspondence. Let us formally denote this isomorphism by a mapping

$$\bar{\cdot} : Q_j \to \overline{Q}_j, \quad q \mapsto \overline{q}, \qquad j \in J,$$

where $\overline{Q}_j$ denotes the set of all $s_j t_j$-paths in $\overline{D}_j$; however, we will henceforth identify paths $\overline{q} \in \overline{Q}_j$ and configurations $q \in Q_j$. Let us also denote by $W_j := L_j \cup R_j$ the structural nodes of $\overline{D}_j$, and by $\overline{D} := (\overline{V}, \overline{A}) := (V \cup \{s_j, t_j : j \in J\}, A \cup \bigcup_{j \in J} \overline{A}_j) = \bigcup_{j \in J} \overline{D}_j$ the *extended train routing digraph*, i.e., the routing digraph D extended by the artificial nodes and return arcs described above, and $\delta_j^{\pm}(W) := \delta^{\pm}(W) \cap A_j \cup \overline{A}_j$, $\forall W \subseteq \overline{V}$.

The extended model also comes in two versions, one using new 0/1 arc variables y_a, $a \in \overline{A}$, for the use of arc a in a configuration-path, and the other with 0/1 path variables y_q, $q \in Q$, for the use of configuration-path $q \in Q$. The resulting formulations, which we call *arc configuration problem*

(ACP) and *path configuration problem* (PCP), read as follows:

$$
\begin{array}{lll|lll}
\text{(ACP)} & \max \sum_{a\in A} w_a x_a & & \text{(PCP)} & \max \sum_{p\in P} w_p x_p & \\
\text{(i)} & \sum_{a\in\delta_i^+(v)} x_a - \sum_{a\in\delta_i^-(v)} x_a = 0 & \forall i\in I, v\in W_i & & & \\
\text{(ii)} & \sum_{a\in\delta_i^+(s_i)} x_a \le 1 & \forall i\in I & \text{(ii)} & \sum_{p\in P_i} x_p \le 1 & \forall i\in I \\
\text{(iii)} & \sum_{a\in\delta_j^+(v)} y_a - \sum_{a\in\delta_j^-(v)} y_a = 0 & \forall j\in J, v\in W_j & & & \\
\text{(iv)} & \sum_{a\in\delta_j^+(s_j)} y_a \le 1 & \forall j\in J & \text{(iv)} & \sum_{q\in Q_j} y_q \le 1 & \forall j\in J \\
\text{(v)} & x_a - y_a \le 0 & \forall a\in A & \text{(v)} & \sum_{p\ni a} x_p - \sum_{q\ni a} y_q \le 0 & \forall a\in A \\
\text{(vi)} & x_a \ge 0 & \forall a\in A & \text{(vi)} & x_p \ge 0 & \forall p\in P \\
\text{(vii)} & y_a \ge 0 & \forall a\in A & \text{(vii)} & y_q \ge 0 & \forall q\in Q \\
\text{(viii)} & x_a \in \mathbb{Z} & \forall a\in A & \text{(viii)} & x_p \in \mathbb{Z} & \forall p\in P \\
\text{(ix)} & y_a \in \mathbb{Z} & \forall a\in A & \text{(ix)} & y_q \in \mathbb{Z} & \forall q\in Q.
\end{array}
$$

Equalities (ACP) (i) and (iii) are *flow conservation constraints*; they route trains i on s_it_i-paths and configurations j on s_jt_j-paths; note that both D_i and $\overline{D}_j$ are acyclic such that no cycles can come up. Constraints (ACP)/(PCP) (ii) and (iv) ensure a train is routed at most once and that at most one configuration can be chosen for each track. The *coupling constraints* (ACP)/(PCP) (v) synchronize routes and configurations. Finally, (APP)/(PPP) (iv) and (v) are the *nonnegativity* and the *integrality constraints*. Note that, again, all variables are implicitly 0/1.

The extended models have the sizes listed in Table 3. Then the LP-relaxation of (ACP) can be solved in polynomial time. For (PCP), consider the pricing problems for routes and configurations. With dual variables γ_i, $i\in I$, π_j, $j\in J$, and λ_a, $a\in A$, for constraints (PCP) (ii), (iv), and (v), respectively, the pricing problem for a route $p\in P_i$ for train $i\in I$ is

$$\exists\, p\in P_i : \gamma_i + \sum_{a\in p}\lambda_a < w_p \iff \sum_{a\in p}(w_a-\lambda_a) > \gamma_i.$$

formulation	variables	non-trivial constraints
ACP	$O(A)$	$O(A)$
PCP	$O(P)+O(Q)$	$O(I)+O(J)$

Table 3: Sizes of Packing Formulations for the Track Allocation Problem.

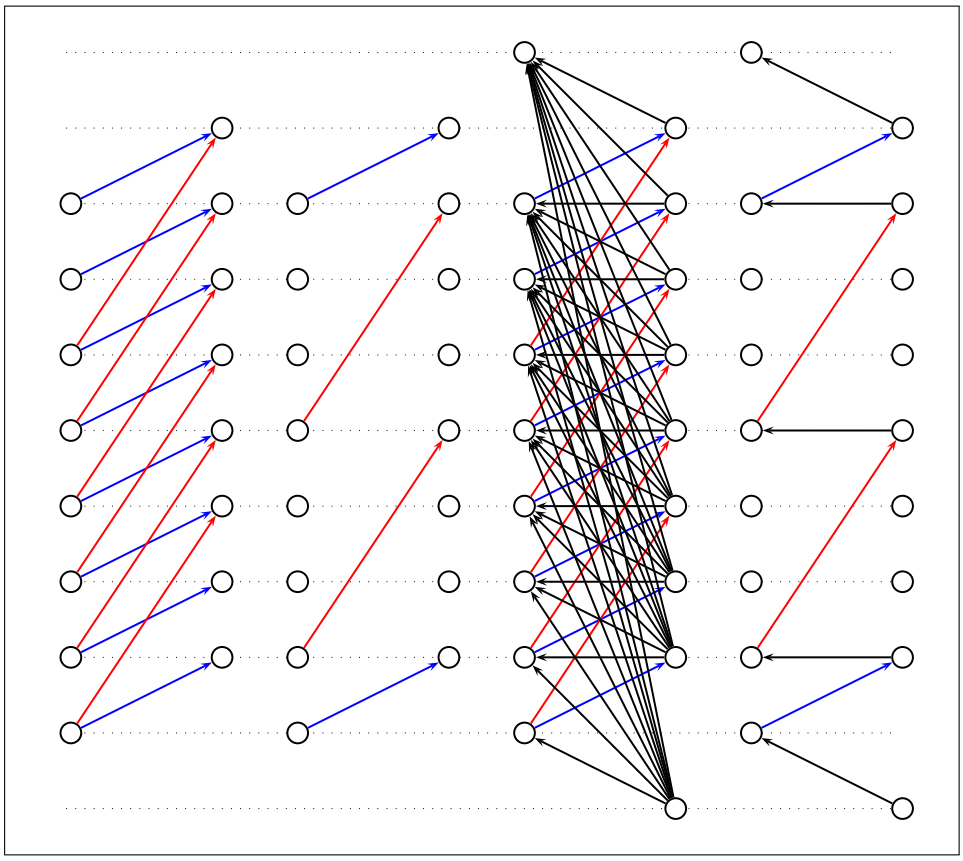


Figure 3: Configuration Routing Digraph for a Single Track: Train Routing Digraph (left), Configuration (half-left), Configuration Routing Digraph (half-right), and the Corresponding Path (right).

This is the same as finding a longest $s_i t_i$-path in D_i w.r.t. arc weights $w_a - \lambda_a$; as D_i is acyclic, this problem can be solved in polynomial time. The pricing problem for a configuration $q \in Q_j$ for track $j \in J$ is

$$\exists\, q \in Q_j : \pi_j - \sum_{a \in q} \lambda_a < 0 \iff \sum_{a \in q} \lambda_a > \pi_j.$$

Using arc weights λ_a, $a \in A_j$, and 0, $a \in \overline{A}_j$, pricing configurations in Q_j is the same as finding longest $s_j t_j$-paths in the acyclic digraph $\overline{D}_j$. This is polynomial. We conclude:

Theoremdd 3.3. *The LP-relaxations associated with the arc configuration formulation ACP and the path configuration formulation PCP of the optimal track allocation problem can be solved in polynomial time.*

Let us quickly state in this pricing context a simple bound on the LP-

value of the path configuration formulation PCP that is useful in practice to overcome tailing-off effects in a column generation procedure. Namely, computing the path lengths $\max_{p\in P_i}\sum_{a\in p}(w_a-\lambda_a)$ and $\max_{q\in Q_j}\sum_{a\in q}\lambda_a$ yield the following LP-bound $\beta=\beta(\gamma,\pi,\lambda)$.

Lemmadd 3.4. *Let $\gamma,\pi,\lambda\geq 0$ be dual variables[1] for PCP and $v_{LP}(PCP)$ the optimum objective value of the LP-relaxation of PCP. Define*

$$\begin{aligned}\eta_i &:= \max_{p\in P_i}\sum_{a\in p}(w_a-\lambda_a)-\gamma_i, && \forall i\in I,\\ \theta_j &:= \max_{q\in Q_j}\sum_{a\in q}\lambda_a-\pi_j, && \forall j\in J,\\ \beta(\gamma,\pi,\lambda) &:= \sum_{i\in I}\max\{\gamma_i+\eta_i,0\}+\sum_{j\in J}\max\{\pi_j+\theta_j,0\}.\end{aligned}$$

Then

$$v_{LP}(PCP)\leq\beta(\gamma,\pi,\lambda).$$

Proof.

- $\gamma_i+\eta_i\geq\sum_{a\in p}(w_a-\lambda_a)\Longrightarrow\gamma_i+\eta_i+\sum_{a\in p}\lambda_a\geq w_p\quad\forall i\in I,p\in P_i.$
- $\pi_j+\theta_j\geq\sum_{a\in q}\lambda_a\Longrightarrow\pi_j+\theta_j-\sum_{a\in q}\lambda_a\geq 0\quad\forall j\in J,q\in Q_j.$
- $(\max\{\gamma+\eta,0\},\max\{\pi+\theta,0\},\lambda)$ (the maximum taken component-wise) is dual feasible for the LP-relaxation of PCP.

□

3.3 Model Comparison

We finally compare the two types of models that we have stated. Starting points are the LP-relaxations of the configuration formulations and those of the packing formulations. As the LP-relaxations of APP and PPP, and of ACP and PCP are obviously equivalent via flow decomposition into paths, it suffices to compare, say, APP and ACP.

Lemmadd 3.5. *Let*

$$\begin{aligned}P_{\mathrm{LP}}(APP) &:= \{x\in\mathbb{R}^A : (\mathrm{APP})\ (\mathrm{i})\text{–}(\mathrm{iv})\}\\ P_{\mathrm{LP}}(ACP) &:= \{(x,y)\in\mathbb{R}^{A\times\overline{A}} : (\mathrm{ACP})\ (\mathrm{i})\text{–}(\mathrm{vii})\}\\ \pi_x &: \mathbb{R}^{A\times\overline{A}}\to\mathbb{R}^A,\quad (x,y)\mapsto x\end{aligned}$$

[1] Note that these will be infeasible during a column generation.

be the polyhedra associated with the LP-relaxations of APP and ACP, respectively, and a mapping that produces a projection onto the coordinates of the train routing variables. Then

$$\pi(P_{\mathrm{LP}}(ACP)) = P_{\mathrm{LP}}(APP).$$

Proof. Let $C_j := \{c \in C : c \subseteq A_j\}$, $j \in J$, be the set of block conflict cliques associated with track j. Consider the polyhedra

$$\begin{aligned}
P &:= \{x \in \mathbb{R}^A : \text{(APP) (i), (ii), (vi)}\},\\
P^j &:= \{x \in \mathbb{R}_+^{A_j} : \sum_{a \in c} x_a \leq 1 \quad \forall c \in C_j\}, \quad j \in J,\\
Q^j &:= \{y \in \mathbb{R}_+^{A_j \times \overline{A}_j} : \sum_{a \in \delta_j^+(v)} y_a = \sum_{a \in \delta_j^-(v)} y_a, \forall v \in W_j, \sum_{a \in \delta_j^+(s_j)} y_a \leq 1\}, \quad j \in J,\\
R^j &:= \{x \in \mathbb{R}_+^{A_j} : \exists y \in Q^j : x \leq y\}, \quad j \in J.
\end{aligned}$$

P^j is integer, because C_j is the family of all maximal cliques of an interval graph, which is perfect; Q_j is integer, because it is the path polytope associated with an acyclic digraph; finally, R^j is integer, because it is the anti-dominant of an integer polytope. Consider integer points, it is easy to see that P^j and R^j coincide, i.e., $P^j = R^j$, $j \in J$. It follows

$$P_{\mathrm{LP}}(\mathrm{APP}) = P \cap \bigcap_{j \in J} P^j = P \cap \bigcap_{j \in J} R^j = \pi(P_{\mathrm{LP}}(\mathrm{ACP})).$$

□

This immediately implies our main Theorem.

Theoremdd 3.6. *Denote by $v(P)$ and $v_{LP}(P)$ the optimal value of problem P and its LP-relaxation, respectively, $P \in \{APP, PPP, ACP, PCP\}$. Then:*

- $v_{LP}(APP) = v_{LP}(PPP) = v_{LP}(ACP) = v_{LP}(PCP)$.
- $v(APP) = v(PPP) = v(ACP) = v(PCP)$.

4 Computational Results

We have implemented model generators for the static formulations APP′ and ACP, and a column generation algorithm for model PCP. This choice is motivated as follows. APP′ is the dominant model in the literature, which we want to benchmark. APP and ACP are equivalent models that improve APP′, both arc-based. ACP is easy to implement. We didn't implement the strong packing model APP, and also not PPP, because these models are

not robust w.r.t. changes in the problem structure, namely, their simplicity depends on the particular clique structure of interval graphs. If more complex constraints are considered, these models can become hard to adapt. In fact, the instances that we are going to consider involve headway matrices that give rise to more numerous and more complex clique structures, such that an implementation of suitably extended models APP and PPP would have been much more difficult than an implementation of the basic versions that we have considered in the theoretical part of this paper. On the other hand, headway constraints are easy to implement in a configuration model, because they specify possible follow-on trips on a track, which is precisely what a configuration does. Formulation PCP is in this sense robust. It is also well suited for column generation to deal with large instances.

In our experiments, we consider the Hanover-Kassel-Fulda area of the German long-distance railway network. All our instances are based on the mesoscopic infrastructure network that is illustrated in Figure 1. It includes data for 37 stations, 120 tracks and 6 different train types (ICE, IC, RE, RB, S, ICG). Because of various possible turnover and driving times for each train type, this produces an infrastructure digraph with 146 nodes, 1480 arcs, and 4320 headway constraints.

Based on the 2002 timetable of Deutsche Bahn AG, we constructed three scenarios that we denote by 146, 285, and 570. The name of the instance gives the number of train requests, which consist of long distance trains (IC, ICE), synchronized regional and suburban passenger trains (S, RE, RB), and freight trains (ICG). The main objective is to maximize the total number of trains in the schedule; on a secondary level, we slightly penalize deviations from certain desired departure and arrival times. Flexibility to reroute trains is controlled by departure and arrival time windows of length at most τ, where τ is a parameter. Increasing τ from 0 to 30 minutes in steps of 2 minutes increases flexibility, but also produces larger train routing digraphs and IPs. After some preprocessing (eliminating arcs and nodes which cannot be part of a feasible train route), the resulting 48 instances have the sizes listed in Table 4.

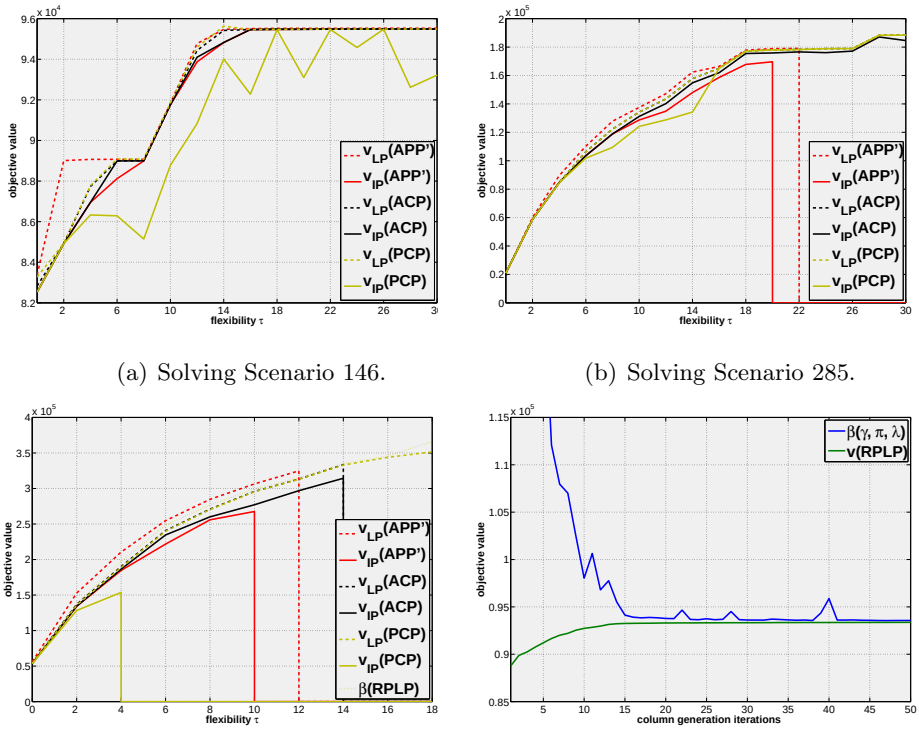
These 48 instances were solved as follows. The root LP-relaxations of the static models APP$'$ and ACP were solved with the dual simplex method of CPLEX 10.0. Then, CPLEXMIP was called for a maximum of at most 1h of running time or 10.000 nodes[2]. Model PCP is solved by column generation, with a limit of at most 100 iterations. The reduced master-LPs were solved with the barrier or the dual simplex method of CPLEX 10.0, depending on the column generation progress. Then, a heuristic integer solution is

[2]That means that we do not always report optimal integer solutions; however, we remark that all instances of scenario 146, of scenario 285 up to $\tau = 24$, and of scenario 570 up to $\tau = 4$ can be solved to proven optimality by running CPLEX long enough.

	146		285		570	
τ	$\#nodes$	$\#arcs$	$\#nodes$	$\#arcs$	$\#nodes$	$\#arcs$
0	2877	3297	362	422	1284	1412
2	4953	6414	1501	1846	5858	6894
4	7428	10131	3262	4284	10912	13334
6	9766	13673	5243	7140	19484	25220
8	12143	17300	8070	11289	28038	37128
10	15617	22476	11126	15840	38380	51944
12	19574	28632	15226	22014	50768	70160
14	24142	35886	19970	29325	65056	91648
16	28877	43673	26201	38985	80376	115212
18	33694	51799	32599	49137	97954	142780
20	38953	60707	39854	60920	116886	173516
22	44072	69636	47486	73473	138512	209040
24	50287	80556	56502	88475	161590	247072
26	56156	91019	65579	103979	186458	289266
28	62035	101581	75820	121840	212722	334878
30	69813	115838	87883	143374	241224	383914

Table 4: Test Scenarios.

constructed, namely, by simply computing an optimal integer solution to the last reduced master-LP, again using CPLEXMIP. All computations were made single threaded on a Dell Precision 650 PC with 2GB of main memory



(a) Solving Scenario 146.

(b) Solving Scenario 285.

(c) Solving Scenario 570.

(d) Generating Columns for Model PCP.

Figure 4: Solving Models APP′, ACP, and PCP.

τ	$\#rows$	$\#cols$	v_{LP}	v_{IP}	$t_{\sum}$	t_{IP}
0	1441	1412	56264.17	53676.00	290.27	0.10
2	8760	6894	152778.29	134190.00	400.88	19.97
4	19369	13334	210479.74	184636.00	658.59	42.14
6	44272	25220	254676.53	221725.00	401.15	103.54
8	81313	37128	284689.94	255870.00	538.52	213.84
10	143917	51944	306437.88	267569.00	1210.23	415.15
12	252530	70160	324781.31	-	1761.22	1360.30
14	413828	91648	-	-	-	-
16	637237	115212	-	-	-	-
18	965427	142780	-	-	-	-
20	1436049	173516	-	-	-	-
22	2094272	209040	-	-	-	-
24	2895176	247072	-	-	-	-
26	3999163	289266	-	-	-	-
28	5422512	334878	-	-	-	-
30	7048470	383914	-	-	-	-

Table 5: Solving Model APP′ for Scenario 570.

τ	$\#rows$	$\#cols$	v_{LP}	v_{IP}	$t_{\sum}$	t_{IP}
0	2332	3875	53968.00	53676.00	216.51	0.21
2	11106	19926	136944.50	134311.00	540.97	6.44
4	21772	39967	189997.08	186467.00	622.68	22.60
6	41498	79234	240622.38	234535.00	1495.82	931.92
8	60390	120957	270900.38	260063.00	2170.88	1401.25
10	83398	170277	295798.29	277073.00	4203.54	3488.38
12	111270	231613	313179.33	296917.00	4760.91	3819.11
14	143270	303302	333515.08	314348.00	4361.18	3943.13
16	177622	377312	-	-	-	-
18	215888	461844	-	-	-	-
20	257378	549535	-	-	-	-
22	304326	649176	-	-	-	-
24	354762	754888	-	-	-	-
26	409556	869796	-	-	-	-
28	467950	985555	-	-	-	-
30	529518	1107237	-	-	-	-

Table 6: Solving Model ACP for Scenario 570.

and a dual Intel Xeon 3.8 GHz CPU running SUSE Linux 10.1.

Figures 4(a), 4(b), and 4(c) summarize our results on the three scenarios 146, 285, and 570, increasing the flexibility from 0 to 30 minutes per train in steps of 2 minutes. It turns out that, in fact, model APP′ produces a noticeably weaker LP-bound (upper bound) than the bounds from the other two models, which are more or less identical. This shows that it is possible to solve the LP-relaxation of model PCP by column generation almost to proven optimality. Figure 4(d) provides a closer look at the master-LP associated with model PCP. Indeed, the upper bound $\beta(\gamma, \pi, \lambda)$ and the value v(RPLP) of the reduced master-LP converge in the column generation process.

With increasing flexibility the models become larger, and at some point the LPs could not be solved any more, because we ran out of memory; the

τ	$\#rows$	$\#cols$	β	v(RPLP)	gap in %	$t_{\sum}$	$\#CG_{iter}$
0	1248	11715	54727.00	53767.00	1.78	468.11	51
2	3314	66012	137376.07	135729.48	1.21	5883.12	100
4	6160	166133	197333.08	188757.73	4.54	13687.55	100
6	11300	238837	248480.85	239768.92	3.63	28258.23	82
8	16414	272565	276867.11	270234.28	2.45	43199.62	82
10	22846	168492	299070.52	295415.44	1.24	73891.21	100
12	30770	214259	314654.48	312960.40	0.54	183123.49	100
14	40696	355918	335061.01	332970.27	0.63	336374.07	57
16	51562	346564	345445.44	343802.93	0.48	198590.48	100
18	63998	266214	366323.70	351502.63	4.22	463379.15	46
20	78478	-	-	-	-	-	
22	94994	-	-	-	-	-	
24	112816	-	-	-	-	-	
26	132826	-	-	-	-	-	
28	154706	-	-	-	-	-	
30	177914	-	-	-	-	-	

Table 7: Solving Model PCP for Scenario 570.

vertical bars in Figures 4(a), 4(b), and 4(c) indicate the largest scenarios that could be solved. $O(A^2)$ constraints kill model APP$'$ early. Model ACP reaches somewhat farther. However, the dynamic model PCP is the one that is able to solve the largest scenarios. It is, in our opinion, also the model that offers the biggest potential for further algorithmic improvements to deal with even larger instances; we are currently working in this direction.

The best integral solutions for our instances were always provided by model ACP. This is no surprise, because this model outperforms APP$'$ in terms of the LP-bound, while the simple IP heuristic that we have applied to PCP is obviously improvable. Tables 5 and 6 list the details for the largest scenario 570 for models APP$'$ and ACP. In addition to the size of the respective LPs we report the LP and IP values, the overall time $t_{\sum}$, and the time t_{IP} spent on finding integral solutions, both in seconds. The dashes in the tables indicate the inability to compute a solution due to an out of memory error. Table 7 gives similar results for model PCP. Here, the LP sizes refer to the final restricted master-LP, and instead of LP and IP values, we list the lower and upper LP-bounds v(RPLP); instead of IP time, we give the number $\#CG_{iter}$ of column generation iterations. Again, the dashes in the tables report out of memory errors. Altogether, Tables 5, 6, and 7 give an impression of the current performance and the limits of our implementations.

References

[1] R. Borndörfer, M. Grötschel, S. Lukac, K. Mitusch, T. Schlechte, S. Schultz & A. Tanner. *An auctioning approach to railway slot allocation.* J. Competition and Regulation in Network

Industries **1**(2):163–196, 2006. ZIB Report 05-45 available at http://opus.kobv.de/zib/volltexte/2005/878/. Cited on pages 119, 121.

[2] U. Brännlund, P. Lindberg, A. Nou & J.-E. Nilsson. *Railway timetabling using Langangian relaxation.* Transportation Sci. **32**(4):358–369, 1998. Cited on page 119.

[3] A. Caprara, M. Fischetti, P. L. Guida, M. Monaci, G. Sacco & P. Toth. *Solution of real-world train timetabling problems.* In *Proc. 34rd Hawaii International Conference on System Sciences*, vol. 3. IEEE Computer Society Press, 2001. Cited on pages 119, 121.

[4] A. Caprara, M. Fischetti & P. Toth. *Modeling and solving the train timetabling problem.* Oper. Res. **50**(5):851–861, 2002. Cited on pages 119, 121.

[5] A. Charnes & M. Miller. *A model for the optimal programming of railway freight train movements.* Management Sci. **3**(1):74–92, 1956. Cited on page 119.

[6] M. Grötschel, L. Lovász & A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, vol. 2 of *Algorithms and Combinatorics*. Springer Verlag, Berlin, 1988. ISBN 3-540-13624-X, 0-387-13624-X (U.S.). Cited on page 124.

[7] E. Helly. *Über Mengen von konvexen Körpern mit gemeinschaftlichen Punkten.* Jahresber. Deutsch. Math.-Verein. **32**:175–176, 1923. Cited on page 123.

[8] S. G. Lukac. Holes, antiholes and maximal cliques in a railway model for a single track. ZIB Report 04-18, Zuse Institute Berlin, 2004. URL http://opus.kobv.de/zib/volltexte/2004/794/. Cited on pages 119, 122.

Paper VI

A Column Generation Approach to Line Planning in Public Transport

R. Borndörfer, M. Grötschel & M. E. Pfetsch.
A column-generation approach to line planning in public transport.
Transportation Science **41**(1):123–132, 2007.

Abstract. The *line planning problem* is one of the fundamental problems in strategic planning of public and rail transport. It consists in finding lines and corresponding frequencies in a transport network such that a given travel demand can be satisfied. There are (at least) two objectives: the transport company wishes to minimize operating costs, the passengers want to minimize traveling times. We propose a new multicommodity flow model for line planning. Its main features, in comparison to existing models, are that the passenger paths can be freely routed and that the lines are generated dynamically. We discuss properties of this model, investigate its complexity, and present a column generation algorithm for its solution. Computational results with data for the city of Potsdam, Germany, are reported.

Mathematics Subject Classification (MSC 2000). 90C06, 90B06

Keywords. Public transit, line planning, column generation, longest path problem

1 Introduction

The *strategic planning* process in public and rail transport is usually divided into consecutive steps of *network design*, *line planning*, and *timetabling*.

Each of these steps can be supported by operations research methods, see for instance the survey articles of Odoni, Rousseau & Wilson (1994) [21] and of Bussieck, Kreuzer & Zimmermann (1997) [6].

This article is about the *line planning problem* (LPP) in public transport. The problem is to design line routes and their frequencies in a street or track network such that a transportation volume, given by a so-called *origin-destination matrix* (OD-matrix), can be routed. The frequency of a line is supposed to indicate a basic timetable period and controls the lines' transportation capacity. There are two competing objectives: on the one hand to minimize the operating costs of lines and on the other hand to minimize user discomfort. User discomfort is usually measured by the total passenger traveling time or the number of transfers during the ride, or both.

The recent literature on the LPP mainly deals with railway networks. One common assumption is the so-called *system split*, which fixes the traveling paths of the passengers *before* the lines are known. A second common assumption is that an optimal line plan can be chosen from a (small) precomputed set of lines. Third, maximization of *direct travelers* (that travel without transfers) is often considered as the objective. In such an approach, transfer waiting times do not play a role.

This article proposes a new, extended multicommodity flow model for the LPP. The model minimizes a combination of total passenger traveling time and operating costs. It generates line routes dynamically, handles frequencies by means of continuous frequency variables, and allows passengers to change their routes according to the computed line system; in particular, we do not assume a system split. These properties aim at line planning scenarios in public transport, where we see less justification for a system split and fewer restrictions in line design than one seems to have in railway line planning. The goal of this article is to show that such a model is tractable and can be used to optimize the line plan of a medium sized town.

The paper is organized as follows. Section 2 surveys the literature on the LPP. Section 3 introduces and discusses our model. Section 4 presents a column generation solution approach. We show that the pricing problem for the passenger variables is a shortest path problem, while the pricing problem for the lines turns out to be an $\mathcal{NP}$-hard longest path problem. However, if only lines of logarithmic length with respect to the number of nodes are considered, the pricing problem can be solved in polynomial time. In Section 5, computational results on a practical problem for the city of Potsdam, Germany, are reported. We end with conclusions in Section 6.

2 Related Work

This section provides a short overview of the literature for the line planning problem. Additional information can be found in the survey article of Ceder & Israeli (1992) [8], which covers the literature up to the beginning of the 1990ies; see also Odoni, Rousseau & Wilson (1994) [21] and Bussieck, Kreuzer & Zimmermann (1997) [6].

The first approaches to the line planning problem had the idea to assemble lines from short pieces in an iterative (and often interactive) process. An early example is the so-called skeleton method described by Silman, Barzily & Passy (1974) [26], that chooses the endpoints of a route and several intermediate nodes which are then joined by shortest paths with respect to length or traveling time; for a variation see Dubois, Bel & Llibre (1979) [14]. In a similar way, Sonntag (1979) [27] and Pape, Reinecke & Reinecke (1995) [22] constructed lines by adjoining small pieces of streets/tracks in order to maximize the number of direct travelers.

Successive approaches precompute some set of lines in a first phase and choose a line plan from this set in a second phase; all articles discussed in the remainder of this section use this idea. For example, Ceder & Wilson (1986) [9] described an enumeration method to generate lines whose length is within a certain factor from the length of the shortest path, while Mandl (1980) [20] proposed a local search strategy to optimize over such a set. Ceder & Israeli (1992) [8]; Israeli & Ceder (1995) [19] introduced a quadratic set covering approach.

An important line of developments is based on the concept of the so-called *system split*. Its starting point is a classification of the links of a transportation system into levels of different speed, as common in railway systems. Assuming that travelers are likely to change to fast levels as early and leave them as late as possible, the passengers are distributed onto several paths in the system, using Kirchhoff-like rules at the transit points, before any lines are known. This fixes the passenger flow on each individual link in the network. The system split was promoted by Bouma & Oltrogge (1994) [3], who used it to develop a branch-and-bound based software system for the planning and analysis of the line system of the Dutch railway network.

Recently, advanced integer programming techniques have been applied to the line planning problem. Bussieck, Kreuzer & Zimmermann (1997) [5] (see also Bussieck (1997) [4]) and Claessens, van Dijk & Zwaneveld (1998) [10] both propose cut-and-branch approaches to select lines from a previously generated set of potential lines and report computations on real world railway data. Both articles deal with homogeneous transport systems, which can be assumed after a system-split is performed as a preprocessing step.

Bussieck, Lindner & Lübbecke (2004) [7] extend this work by incorporating nonlinear components into the model. Goossens, van Hoesel & Kroon (2002) [17]; Goossens, van Hoesel & Kroon (2004) [18] show that practical railway problems can be solved within reasonable time and quality by a branch-and-cut approach, even for the simultaneous optimization of several transportation systems. Schöbel & Scholl (2005) [24]; Scholl (2005) [25] study a Dantzig-Wolfe decomposition approach to route passengers through an expanded line-network in order to minimize the number of transfers or the transfer time.

3 Line Planning Model

We typeset vectors in bold face, scalars in normal face. If $\boldsymbol{v} \in \mathbb{R}^J$ is a real valued vector and I a subset of J, we denote by $\boldsymbol{v}(I)$ the sum over all components of $\boldsymbol{v}$ indexed by I, i.e., $\boldsymbol{v}(I) := \sum_{i \in I} v_i$.

For the line planning problem (LPP) we are given a number M of transportation *modes* (bus, tram, subway, etc.), an undirected multigraph $G = (V, E) = (V, E_1 \dot\cup \ldots \dot\cup E_M)$ representing a multi-modal transportation network, *terminal sets* $\mathcal{T}_1, \ldots, \mathcal{T}_M \subseteq V$ of nodes for each mode where lines can start and end, line *operating costs* $\boldsymbol{c}^1 \in \mathbb{Q}_+^{E_1}, \ldots, \boldsymbol{c}^M \in \mathbb{Q}_+^{E_M}$ on the edges, *fixed costs* $C_1, \ldots, C_M \in \mathbb{Q}_+$ for the set-up of a line for each mode, *vehicle capacities* $\kappa_1, \ldots, \kappa_M \in \mathbb{Q}_+$ for each mode, and *edge capacities* $\boldsymbol{\Lambda} \in \mathbb{Q}_+^E$. Denote by $G_i = (V, E_i)$ the subgraph of G corresponding to mode i. See Figure 1 for an example network.

A *line of mode* i is a path in G_i connecting two (different) terminals of $\mathcal{T}_i$. Note that paths are always *simple*, i.e., the repetition of nodes is not allowed; it is possible to consider additional constraints on the formation of lines such as a maximum length etc. Let $c_\ell := \sum_{e \in \ell} c_e^i$ be the operating cost of line ℓ of mode i, $C_\ell := C_i$ be its fixed cost, and $\kappa_\ell := \kappa_i$ be its vehicle capacity. Let $\mathcal{L}$ be the set of all feasible lines. Furthermore, $\mathcal{L}_e := \bigcup\{\ell \in \mathcal{L} : e \in \ell\}$ is the set of lines that use edge $e \in E$.

The problem formulation further involves a (not necessarily symmetric) *origin-destination matrix* (OD-matrix) $(d_{st}) \in \mathbb{Q}_+^{V \times V}$ of travel demands, i.e., d_{st} is the number of passengers that want to travel from node s to node t. Let $D := \{(s,t) \in V \times V : d_{st} > 0\}$ be the set of all *OD-pairs*.

Finally, we derive a directed *passenger route graph* (V, A) from $G = (V, E)$ by replacing each edge $e \in E$ with two antiparallel arcs $a(e)$ and $\overline{a}(e)$; conversely, let $e(a) \in E$ be the undirected edge corresponding to $a \in A$. For simplicity of notation, we denote this digraph also by $G = (V, A)$. We are given *traveling times* $\tau_a \in \mathbb{Q}_+$ for every arc $a \in A$. For an OD-pair

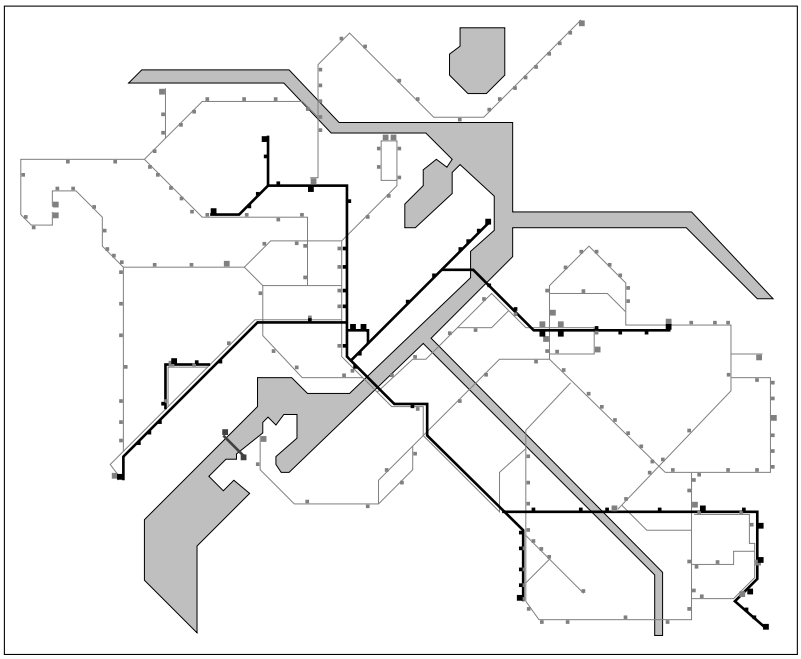


Figure 1: Multi-Modal Transportation Network in Potsdam. Black: Tram, Lightgray: Bus, Darkgray: Ferry, Large Nodes: Terminals, Small Nodes: Stations, Grey: Rivers and Lakes.

$(s,t) \in D$, an (s,t)-*passenger path* is a directed path in (V,A) from s to t. Let $\mathcal{P}_{st}$ be the set of all (s,t)-passenger paths, $\mathcal{P} := \bigcup\{p \in \mathcal{P}_{st} : (s,t) \in D\}$ the set of all passenger paths, and $\mathcal{P}_a := \bigcup\{p \in \mathcal{P} : a \in p\}$ the set of all passenger paths that use arc a. The *traveling time* of a passenger path p is defined as $\tau_p := \sum_{a \in p} \tau_a$.

With this notation, the line planning problem can be modeled using three kinds of variables:

$y_p \in \mathbb{R}_+$	the flow of passengers traveling from s to t on path $p \in \mathcal{P}_{st}$,
$f_\ell \in \mathbb{R}_+$	the frequency of line $\ell \in \mathcal{L}$,
$x_\ell \in \{0,1\}$	a decision variable for using line $\ell \in \mathcal{L}$.

$$\text{(LPP)} \quad \min\ \boldsymbol{\tau}^{\mathrm{T}}\boldsymbol{y} + \boldsymbol{C}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{c}^{\mathrm{T}}\boldsymbol{f}$$

$$\begin{aligned}
\boldsymbol{y}(\mathcal{P}_{st}) &= d_{st} && \forall\,(s,t) \in D && \text{(i)}\\
\boldsymbol{y}(\mathcal{P}_a) - \sum_{\ell : e(a) \in \ell} \kappa_\ell f_\ell &\le 0 && \forall\, a \in A && \text{(ii)}\\
\boldsymbol{f}(\mathcal{L}_e) &\le \Lambda_e && \forall\, e \in E && \text{(iii)}\\
\boldsymbol{f} &\le F\boldsymbol{x} && && \text{(iv)}\\
x_\ell &\in \{0,1\} && \forall\, \ell \in \mathcal{L} && \text{(v)}\\
f_\ell &\ge 0 && \forall\, \ell \in \mathcal{L} && \text{(vi)}\\
y_p &\ge 0 && \forall\, p \in \mathcal{P}. && \text{(vii)}
\end{aligned}$$

G	multi-modal transport network	G_i	subnetwork for mode i
$\mathcal{T}_i$	terminals for mode i	$\boldsymbol{c}^i$	line operating costs for mode i
c_ℓ	operating costs for line ℓ	C_i	line fixed costs for mode i
κ_i	vehicle capacity for mode i	κ_ℓ	vehicle capacity for line ℓ
$\mathcal{L}$	set of all lines	$\mathcal{L}_e$	lines using edge e
D	set of OD-pairs	d_{st}	travel demand between s and t
τ_a	traveling time on arc a	τ_p	traveling time on path p
$\mathcal{P}$	set of all passenger paths	$\mathcal{P}_{st}$	paths between s and t
y_p	passenger flow on path p	x_ℓ	whether line ℓ is used
f_ℓ	frequency of line ℓ	Λ_e	frequency bounds for edge e

Table 1: Notation and Terminology.

The *passenger flow constraints* (i) and the nonnegativity constraints (vii) model a multicommodity flow problem for the passenger flow, where the commodities correspond to the OD-pairs $(s,t) \in D$. This part guarantees that the demand is routed. The *capacity constraints* (ii) link the passenger paths with the line paths to ensure sufficient transportation capacity on each arc. The *frequency constraints* (iii) bound the total frequency of lines using an edge. Inequalities (iv) link the frequencies with the decision variables for the use of lines; they guarantee that the frequency of a line is 0 whenever it is not used. Here, F is an upper bound on the frequency of a line; for technical reasons, we assume that $F \geq \Lambda_e$ for all $e \in E$, see Section 4 for more information.

Let us discuss some properties of the model before we investigate its algorithmic tractability.

Objectives: The objective of the model has two competing parts, namely, to minimize total passenger traveling time $\boldsymbol{\tau}^{\mathrm{T}}\boldsymbol{y}$ and to minimize costs $\boldsymbol{C}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{c}^{\mathrm{T}}\boldsymbol{f}$. Here, $\boldsymbol{C}^{\mathrm{T}}\boldsymbol{x}$ is the fixed cost for setting up lines and $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{f}$ is the variable cost for operating these lines at frequencies $\boldsymbol{f}$. The model allows to adjust the relative importance of one part over the other by an appropriate scaling of the respective objective coefficients. Including fixed costs allows to consider objectives such as minimizing the number of lines; note that LPP is a linear program (LP) if all fixed costs are zero.

OD-Matrices: Each entry in an OD-matrix gives the number of passengers that want to travel from one point in the network to another point within a fixed time horizon. It is well known that such data have certain deficiencies. For instance, OD-matrices depend on the geometric discretization used, they are highly aggregated, they give only a snapshot type of view, it is often questionable how well the entries represent the real situation, and they should only be used when the transportation demand can be assumed to be fixed. However, OD-matrices are at present the industry standard

for estimating transportation demand. It is already quite an art and rather costly to assemble this data and there is currently no alternative in sight.

Time horizon: The LPP implicitly contains a time horizon via the OD-matrix. Usually, OD-data are aggregated over one day, but it is similarly appropriate to consider, for instance, peak traffic in rush hours. In fact, the asymmetry of demands in rush hours was one of the reasons why we consider directed passenger paths.

Passenger Routes: Since the traveling times $\boldsymbol{\tau}$ are nonnegative, we can assume passenger routes to be (simple) paths.

Our model does not fix passenger paths according to a system split, but allows to freely route passengers according to the computed lines. This is targeted at local public transport systems, where, in our opinion, people determine their traveling paths according to the line system and not only according to the network topology. Except for the work of Schöbel & Scholl (2005) [24]; Scholl (2005) [25], which has been done independently of ours, such routings have not been considered in the context of line planning before.

Our model computes a set of passenger paths that minimize the total traveling times $\boldsymbol{\tau}^{\mathrm{T}}\boldsymbol{y}$ in the sense of a system optimum. However, in our case, with a linear objective function and linear capacities, it can be shown that the resulting system optimum is also a user equilibrium, namely, the so-called Beckmann user equilibrium, see Correa, Schulz & Stier Moses (2004) [11]. We do not address the question why passengers should choose this equilibrium out of several possible equilibria that can arise in routing with capacities.

The routing in our model allows for passengers paths of arbitrary travel times, which may force some passengers to long detours. We remark that this problem could be handled by introducing appropriate bounds on the travel times of paths. This would, however, turn the pricing problem for the passenger paths into an $\mathcal{NP}$-hard resource constrained shortest path problem; see Section 4.1. Note also that such an approach would measure travel times with respect to shortest paths in the underlying network (independent of any line system). Ideally, however, one would like to compare to the shortest paths using only arcs covered by the computed line system.

Line Routes: The literature generally takes line routes as (simple) bidirected paths, and we do the same in this article. In fact, a restriction forcing some sort of simplicity is necessary in order to prevent repetitions around cycles. As a slight generalization of the concept of simplicity, one could investigate the case where one assumes that every line route is bounded in length or "almost" simple, i.e., no node is repeated within a given interval.

It is easy to incorporate additional constraints on the formation of individual lines and constraints on sets of lines, e.g., that the length of a line should not deviate too much from a shortest path between its endpoints or bounds on the number of lines using an edge. Such constraints are important in practice. In this article we consider bounds on the number of edges in a line. Let us give two arguments why this case is practically relevant.

The first argument is based on an idea of a transportation network as a planar graph, probably of high connectivity. Suppose this network occupies a square, in which n nodes are evenly distributed. A typical line starts in the outer regions of the network, passes through the center, and ends in another outer region; we would expect such a line to be of length $\mathcal{O}(\sqrt{n})$.

Real networks, however, are not only (more or less) planar, but often resemble trees. But in a *balanced* and preprocessed tree, where each node degree is at least 3, the length of a path between any two nodes is only $\mathcal{O}(\log n)$.

Transfers: Transfers between lines are currently ignored in our model, because constraints (iii) only control the total capacity on edges and not the assignment of passengers to lines. The problem are not transfers between different modes, which can be handled by linking the mode networks G_i with appropriate transfer edges, weighted by estimated transfer times. A similar trick could in principle be used for transfers between lines of the same mode, using an appropriate expansion of the graph. However, this greatly increases the complexity of the model, and it introduces degeneracy; it is unclear whether such a model remains tractable for practical data.

Frequencies: Frequencies indicate the (approximate) number of times vehicles need to be employed in order to serve the demand over the time horizon. In a real world line plan, frequencies often have to produce a regular timetable and hence are not allowed to take arbitrary fractional values. Our model, however, treats frequencies as continuous values. This is a simplification. We have introduced fixed costs in order to reduce the number of lines and decrease the likelihood of low frequencies. In addition, we could have forced our model to accept only a finite number of frequencies by enumerating lines with fixed frequencies in a similar way as Claessens, van Dijk & Zwaneveld (1998) [10] and Goossens, van Hoesel & Kroon (2002) [17]; Goossens, van Hoesel & Kroon (2004) [18]; but the resulting model would be much harder to solve. However, as the frequencies are mainly used to adjust line capacities, we do (at present) not care so much about "nice" frequencies and view the fractional values as approximations or clues to "sensible" values.

4 Column Generation

The LP relaxation of (LPP) can be simplified by eliminating the $\boldsymbol{x}$-variables. In fact, since (LPP) minimizes over nonnegative costs, one can assume w.l.o.g. that inequalities (iv) are satisfied with equality, i.e., there is an optimal LP solution such that $Fx_\ell = f_\ell \Leftrightarrow x_\ell = f_\ell/F$ for all lines ℓ. Substituting for $\boldsymbol{x}$, we observe that the inequalities $f_\ell \leq F$ remaining after the elimination are dominated by inequalities (iii) and hence can be omitted (recall that we assumed $F \geq \Lambda_e$). Setting $\gamma_\ell = C_\ell/F + c_\ell$, we arrive at the following equivalent, but simpler, linear program:

$$
\begin{aligned}
\text{(LP)} \qquad & \min\ \boldsymbol{\tau}^{\mathrm{T}}\boldsymbol{y} + \boldsymbol{\gamma}^{\mathrm{T}}\boldsymbol{f} \\
& \boldsymbol{y}(\mathcal{P}_{st}) = d_{st} && \forall\ (s,t) \in D && \text{(i)} \\
& \boldsymbol{y}(\mathcal{P}_a) - \sum_{\ell : e(a) \in \ell} \kappa_\ell f_\ell \leq 0 && \forall\ a \in A && \text{(ii)} \\
& \boldsymbol{f}(\mathcal{L}_e) \leq \Lambda_e && \forall\ e \in E && \text{(iii)} \\
& f_\ell \geq 0 && \forall\ \ell \in \mathcal{L} && \text{(iv)} \\
& y_p \geq 0 && \forall\ p \in \mathcal{P}. && \text{(v)}
\end{aligned}
$$

Note that (LP) contains only a polynomial number of inequalities (apart from the nonnegativity constraints (iv) and (v)).

We aim at solving (LP) with a column generation approach (see Barnhart et al. (1998) [2] for an introduction) and therefore investigate the corresponding pricing problems. These pricing problems are studied in terms of the dual of (LP). Denote the variables of the dual as follows: $\boldsymbol{\pi} = (\pi_{st}) \in \mathbb{R}^D$ (flow constraints (i)), $\boldsymbol{\mu} = (\mu_a) \in \mathbb{R}^A$ (capacity constraints (ii)), and $\boldsymbol{\eta} \in \mathbb{R}^E$ (frequency constraints (iii)). The dual of (LP) is:

$$
\begin{aligned}
\max\ & \boldsymbol{d}^{\mathrm{T}}\boldsymbol{\pi} - \boldsymbol{\Lambda}^{\mathrm{T}}\boldsymbol{\eta} \\
& \pi_{st} - \boldsymbol{\mu}(p) \leq \tau_p && \forall\ p \in \mathcal{P}_{st},\ (s,t) \in D \\
& \kappa_\ell\, \boldsymbol{\mu}(\ell) - \boldsymbol{\eta}(\ell) \leq \gamma_\ell && \forall\ \ell \in \mathcal{L} \\
& \boldsymbol{\mu},\ \boldsymbol{\eta} \geq 0,
\end{aligned}
$$

where

$$
\boldsymbol{\mu}(\ell) = \sum_{e \in \ell} \big(\mu_{a(e)} + \mu_{\overline{a}(e)}\big).
$$

It will turn out that the pricing problem for the line variables f_ℓ is a longest path problem; the pricing problem for the passenger variables y_p, however, is a shortest path problem.

4.1 Pricing of the Passenger Variables

The reduced cost $\overline{\tau}_p$ for variable y_p with $p \in \mathcal{P}_{st}$, $(s,t) \in D$, is

$$\overline{\tau}_p = \tau_p - \pi_{st} + \boldsymbol{\mu}(p) = \tau_p - \pi_{st} + \sum_{a \in p} \mu_a = -\pi_{st} + \sum_{a \in p} (\mu_a + \tau_a).$$

The pricing problem for the $\boldsymbol{y}$-variables is to find a path p such that $\overline{\tau}_p < 0$ or to conclude that no such path exists. This can easily be done in polynomial time as follows. For all $(s,t) \in D$, we search for a shortest (s,t)-path p with respect to the nonnegative weights $(\mu_a + \tau_a)$ on the arcs; we can, for instance, use Dijkstra's algorithm. If the length of this path p is less than π_{st}, then y_p is a candidate variable to be added to the LP, otherwise we proved that no such path exists (for the pair (s,t)). Note that each passenger path can assumed to be simple: just remove cycles of length 0 – or trust Dijkstra's algorithm, which produces only simple paths.

4.2 Pricing of the Line Variables

The pricing problem for line variables f_ℓ is more complicated. The reduced cost $\overline{\gamma}_\ell$ for a variable f_ℓ is

$$\overline{\gamma}_\ell = \gamma_\ell - \kappa_\ell \, \boldsymbol{\mu}(\ell) + \boldsymbol{\eta}(\ell) = \gamma_\ell - \sum_{e \in \ell} \left(\kappa_\ell \left(\mu_{a(e)} + \mu_{\overline{a}(e)} \right) - \eta_e \right).$$

The corresponding pricing problem consists in finding a (simple) path ℓ of mode i such that

$$\begin{aligned} 0 > \overline{\gamma}_\ell &= \gamma_\ell - \textstyle\sum_{e \in \ell} \left(\kappa_\ell \left(\mu_{a(e)} + \mu_{\overline{a}(e)} \right) - \eta_e \right) \\ &= C_\ell / F + c_\ell - \textstyle\sum_{e \in \ell} \left(\kappa_\ell \left(\mu_{a(e)} + \mu_{\overline{a}(e)} \right) - \eta_e \right) \\ &= C_i / F + \textstyle\sum_{e \in \ell} c_e^i - \sum_{e \in \ell} \left(\kappa_i \left(\mu_{a(e)} + \mu_{\overline{a}(e)} \right) - \eta_e \right) \\ &= C_i / F + \textstyle\sum_{e \in \ell} \left(c_e^i - \kappa_i \left(\mu_{a(e)} + \mu_{\overline{a}(e)} \right) + \eta_e \right) \\ \Leftrightarrow \; & \textstyle\sum_{e \in \ell} (\kappa_i \left(\mu_{a(e)} + \mu_{\overline{a}(e)} \right) - \eta_e - c_e^i) > C_i / F. \end{aligned}$$

This problem turns out to be a maximum weighted path problem, since the weights $(\kappa_i \left(\mu_{a(e)} + \mu_{\overline{a}(e)} \right) - \eta_e - c_e^i)$ are not restricted in sign. Hence, the pricing problem for the line variables is $\mathcal{NP}$-hard Garey & Johnson (1979) [16]. This shows that solving the LP relaxation (LP) is $\mathcal{NP}$-hard as well. In fact, we can prove the stronger result that the line planning problem itself is $\mathcal{NP}$-hard, even with fixed costs 0, independent of the model (Proposition 4.1 implies that (LP) is $\mathcal{NP}$-hard, because (LPP) is equivalent to (LP) for fixed costs 0).

Propositiondd 4.1. *The line planning problem LPP is $\mathcal{NP}$-hard, even with fixed costs 0.*

Proof. We reduce the Hamiltonian path problem, which is strongly $\mathcal{NP}$-complete Garey & Johnson (1979) [16], to the LPP with fixed costs 0. Let (H, s, t) be an instance of the Hamiltonian path problem, i.e., $H = (V, E)$ is a graph and s and t are two distinct nodes of H.

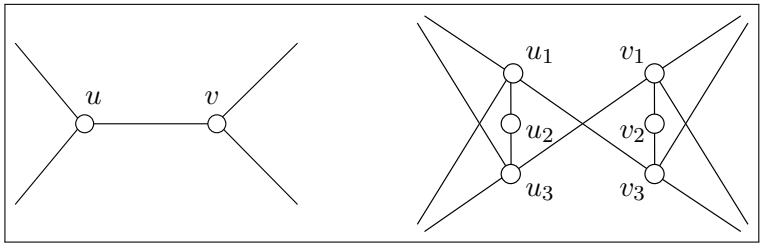


Figure 2: The Node Splitting Gadget in the Proof of Proposition 4.1

For the reduction, we are going to derive an appropriate instance of LPP. The underlying network is formed by a graph $H' = (V', E')$, which arises from H by splitting each node v into three copies v_1, v_2, and v_3. For each node $v \in V$, we add edges $\{v_1, v_2\}$ and $\{v_2, v_3\}$ to E' and for each edge $\{u, v\}$ the edges $\{u_1, v_3\}$ and $\{u_3, v_1\}$, see Figure 2. Our instance of LPP contains just a single mode with only two terminals s_1 and t_3 such that every line must start at s_1 and end at t_3. The demands are $d_{v_1 v_2} = 1$ $(v \in V)$ and 0 otherwise, and the capacity of every line is 1. For every $e \in E$, we set Λ_e to some high value (e.g., to $|V|$). The cost of all edges is set to 0, except for the edges incident to s_1, for which the costs are set to 1. The traveling times are set to 0 everywhere. It follows that the value of a solution to LPP is the sum of the frequencies of all lines.

Assume that $p = (s, v^1, \ldots, v^k, t)$ (for $v^1, \ldots, v^k \in V$) is an (s, t)-Hamiltonian path in H. Then $p' = (s_1, s_2, s_3, v_1^1, v_2^1, v_3^1, \ldots, v_1^k, v_2^k, v_3^k, t_1, t_2, t_3)$ is an (s_1, t_3)-Hamiltonian path in H', which gives rise to an optimal solution of LPP. Namely, we can take p' as the route of a single line with frequency 1 and route the demands $d_{v_1 v_2} = 1$ for every $v \in V$ on this line directly from v_1 to v_2. As the frequency of p' is 1, the objective value of this solution is also 1. On the other hand, every solution to LPP must have value at least one, as every line has to pass an edge incident to s_1 and the sum of the frequencies of lines visiting an arbitrary edge of type $\{v_1, v_2\}$, for $v \in V$, is at least 1. This proves that LPP has a solution of value 1, if (H, s, t) contains a Hamiltonian path.

For the converse, assume that there exists a solution to LPP of value 1, for which we ignore lines with frequency 0. We know that every edge $\{v_1, v_2\}$ $(v \in V)$ is covered by at least one line of the solution. If every line contains all edges $\{v_1, v_2\}$ $(v \in V)$, each such line gives rise to a Hamiltonian path (since the line paths are simple) and we are done. Otherwise, there must be an edge $e = \{v_1, v_2\}$ $(v \in V)$ which is not covered by all of the lines.

Since the lines have to provide enough capacity, the sum of the frequencies of the lines covering e is at least 1. However, the edges incident to s_1 are covered by the lines covering edge e plus at least one more line of nonzero frequency. Hence, the total sum of all frequencies is larger than one, which is a contradiction to the assumption that the solution has value 1.

This shows that there exists an (s,t)-Hamiltonian path in H if and only if an optimal solution of LPP with respect to H' has value 1. □

4.3 Pricing of Length Restricted Lines

Let us now consider the pricing problem for line planning problems with bounds on the lengths of the lines, i.e., the number of edges of a line. Consider for this purpose the graph $G = (V, E)$ (for simplicity of notation with only one mode) with arbitrary edge weights $w_e \in \mathbb{Q}$ for all $e \in E$, and a source node s and a sink node t. We let $n = |V|$ and $m = |E|$. In this setting the line pricing problem is to find a maximum weight path from s to t with respect to $\boldsymbol{w}$. We first show that this problem is $\mathcal{NP}$-hard for the case where the length of a line is bounded by $\mathcal{O}(\sqrt{n})$.

Propositiondd 4.2. *It is $\mathcal{NP}$-hard to compute a maximum weight path from s to t of length at most k, if $k \in \mathcal{O}\big(n^{1/N}\big)$ for any fixed $N \in \mathbb{N} \setminus \{0\}$.*

Proof. Let (H, s, t) be an instance of the Hamiltonian path problem, where H is a graph with n nodes. We add $(n^N - n)$ isolated nodes to H in order to obtain a graph H' with n^N nodes; note that n^N is polynomial in n for fixed N. Let the weights on the edges be 1. If we would be able to find a maximum weight path from s to t with at most $k = (n^N)^{1/N} = n$ edges, we could solve the Hamiltonian path problem for H in polynomial time. □

We now provide a result which shows that the maximum weighted path problem can be solved in polynomial time in the case when the lengths of the paths are at most $\mathcal{O}(\log n)$. Our method is a direct generalization of work by Alon, Yuster & Zwick (1995) [1] on the unweighted case; it works both for directed and undirected graphs.

Alon et al. consider the problem to find simple paths of fixed length $k-1$ in a graph. Their basic idea is to randomly color the nodes of the graph with k colors and only allow paths that use distinct colors for each node; such paths are called *colorful* with respect to the coloring and are necessarily simple. Choosing a coloring $c : V \to \{1, \ldots, k\}$ uniformly at random, every path using at most $k-1$ edges has a chance of at least $k!/k^k > e^{-k}$ to be colorful with respect to c. If we repeat this process $\alpha \cdot e^k$ times with $\alpha > 0$, the probability that a given path p with at most $k-1$ edges is never colorful is

less than

$$\left(1 - e^{-k}\right)^{\alpha \cdot e^k} < e^{-\alpha}.$$

Hence, the probability that p is colorful at least once is at least $1 - e^{-\alpha}$. The search for such colorful paths can be performed using dynamic programming, which leads to an algorithm running in $m \cdot 2^{\mathcal{O}(k)}$ expected time. This algorithm is then derandomized.

These arguments yield the following result for the weighted undirected case, which is easily seen to be valid for directed graphs as well.

Propositiondd 4.3. *Let $G = (V, E)$ be a graph with m edges, k be a fixed number, and $c : V \to \{1, \ldots, k\}$ be a coloring of the nodes of G. Let s be a node in G and (w_e) be edge weights. Then a colorful maximum weight path with respect to $\boldsymbol{w}$ using at most $k - 1$ edges from s to every other node can be found in time $\mathcal{O}\big(m \cdot k \cdot 2^k\big)$, if such paths exist.*

Proof. We find the maximum weight of such paths by dynamic programming. Let $v \in V$, $i \in \{1, \ldots, k\}$, and $C \subseteq \{1, \ldots, k\}$ with $|C| \leq i$. Define $w(v, C, i)$ to be the weight of the maximum weight colorful path with respect to $\boldsymbol{w}$ from s to v using at most $i - 1$ edges and using the colors in C. Hence, for each iteration i we store the set of colors of all maximum weight colorful paths from s to v using at most $i - 1$ edges. Note that we do not store the set of paths, only their colors. Hence, at each node we store at most 2^i entries. The entries of the table are initialized with minus infinity and we set $w(s, \{c(s)\}, 1) = 0$.

At iteration $i \geq 1$, let (u, C, i) be an entry in the dynamic programming table. If for some edge $e = \{u, v\} \in E$ we have $c(v) \notin C$, let $C' = C \cup \{c(v)\}$ and set

$$w(v, C', i + 1) = \max\big\{w(u, C, i) + w_e,\ w(v, C', i + 1),\ w(v, C', i)\big\}.$$

The term $w(v, C', i + 1)$ accounts for the cases where we already found a path to v (using at most i edges) with higher weight, whereas $w(v, C', i)$ makes sure that paths using at most $i - 1$ edges to v are accounted for. After iteration $i = k$, we take the maximum of all entries corresponding to each node v, which is the wanted result. The number of updating steps is bounded by

$$\sum_{i=0}^{k} i \cdot 2^i \cdot m = m \cdot \big(2 + 2^{k+1}(k - 1)\big) = \mathcal{O}\big(m \cdot k \cdot 2^k\big).$$

The sum on the left side of this equation arises as follows. In iteration i, m edges are considered; each edge $\{u, v\}$ starts at node u, to which at most 2^i

labels $w(u, C, i)$ are associated, one for each possible set C; for each such set, checking whether $c(v) \in C$ takes time $\mathcal{O}(i)$. The summation formula itself can be proved by induction, see also Petkovsek, Wilf & Zeilberger (1996) [23], Exc. 5.7.1, p. 95. The algorithm can be easily modified to actually find the maximum weight paths. □

We can use Proposition 4.3 to produce an algorithm which finds a maximum weight path in $\alpha\, e^k\, \mathcal{O}\big(mk2^k\big) = \alpha\, \mathcal{O}\big(m \cdot 2^{\mathcal{O}(k)}\big)$ time with high probability. Then a derandomization can be performed by a clever enumeration of colorings such that each path with at most $k-1$ edges is colorful with respect to at least one such coloring. Alon et al. combine several techniques to show that $2^{\mathcal{O}(k)} \cdot \log n$ colorings suffice. Applying this result we obtain:

Theoremdd 4.4. *Let $G = (V, E)$ be a graph with n nodes and m edges and k be a fixed number. Let s be a node in G and (w_e) be edge weights. Then a maximum weight path with respect to $\boldsymbol{w}$ using at most $k-1$ edges from s to every other node can be found in time $\mathcal{O}\big(m \cdot 2^{\mathcal{O}(k)} \cdot \log n\big)$, if such paths exist.*

If $k \in \mathcal{O}(\log n)$, this yields a polynomial time algorithm. Hence, by the discussion above, we get the following result.

Corollarydd 4.5. *The LP relaxation of (LPP) can be solved in polynomial time, if the lengths of the lines are most k, with $k \in \mathcal{O}(\log n)$.*

4.4 Algorithm

We used the results of the previous subsections to implement a column generation algorithm for the solution of the model (LPP) with length restricted lines. As an overall objective function, we used the weighted sum

$$\lambda\,(\boldsymbol{C}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{c}^{\mathrm{T}}\boldsymbol{f}) + (1-\lambda)\,\boldsymbol{\tau}^{\mathrm{T}}\boldsymbol{y},$$

where $\lambda \in [0, 1]$ is a parameter weighing the two parts.

The algorithm solves the LP relaxation in a first phase and constructs a feasible line plan using a greedy type heuristic in a second phase.

To solve the LP relaxation, our algorithm iteratively prices out passenger and line path variables until no improving variables are found. We solve the master LP with the barrier algorithm and, towards the end of the process, with the primal simplex algorithm of CPLEX 9.1. We check for new passenger path variables for all OD-pairs using Dijkstra's algorithm, see Section 4.1, until no more improving passenger paths are found. If we don't find an improving passenger path, we price out line variables for all line modes and all feasible terminal pairs. We have implemented two different methods

for the pricing of (simple) line paths, namely, we either use an enumeration or the randomized coloring algorithm of Section 4.3 (we do not derandomize the algorithm). If an improving passenger or line path has been found, another iteration is started; otherwise, the LP is solved.

In the second phase, our algorithm tries to construct a good integer solution from a line pool consisting of the lines having nonzero frequencies in the optimal LP solution. The heuristic is motivated by the observation that the solution of the LP relaxation of a line planning problem often contains lines with very low frequencies. We try to remove these lines by a simple greedy method based on a strong branching selection criterion. In the beginning the $\boldsymbol{x}$-variables of all lines in the pool are set to 1. In each iteration, we tentatively remove a line (set its x-variable to 0), compute the objective $\lambda\, \boldsymbol{c}^{\mathrm{T}}\boldsymbol{f} + (1-\lambda)\, \boldsymbol{\tau}^{\mathrm{T}}\boldsymbol{y}$ of the LP obtained by fixing the line variables as described, pricing passenger variables as needed, and add the fixed costs $\boldsymbol{C}^{\mathrm{T}}\boldsymbol{x}$ of all lines that are fixed to 1. After probing candidate lines with the smallest $\boldsymbol{f}$-values in this way, we permanently delete the line whose removal resulted in the smallest objective. We repeat this elimination as long as the remaining set of lines is still feasible, i.e., all demands can be routed, and the objective function decreases.

5 Computational Results

In this section we report on computational experience with line planning problems for the city of Potsdam, Germany. The experiments originate from a joint project with the two local public transport companies ViP Verkehrsgesellschaft GmbH and Havelbus Verkehrsgesellschaft mbH, the city of Potsdam, and the software company IVU Traffic Technologies AG.

Potsdam is a medium sized town near Berlin; it has about 150,000 inhabitants. Its public transportation system uses city buses and trams (operated by ViP) and regional buses (operated by Havelbus). Additionally, there are regional trains connecting Potsdam to its surroundings (operated by Deutsche Bahn AG) and a city railroad (operated by S-Bahn Berlin) which provides connections to Berlin. As regional trains and the city railroad are not operated by ViP and Havelbus, the associated lines routes are assumed to be fixed.

5.1 Data

Our data consists of a multi-modal traffic network of Potsdam and an associated OD-matrix, which had been used by IVU in a consulting project for planning the Potsdam network (Nahverkehrsplan). The data represents

the line system of Potsdam of 1998. It has 27 bus lines and 4 tram lines. Including line variants, the total number of lines was 80. The network has 951 nodes, including 111 OD-nodes, and 1321 edges. The maximum length of a line is 47 edges.

The network was preprocessed as follows. We removed isolated nodes. Then we iteratively removed "leaves" in the graph, i.e., nodes which have only one neighbor, and iteratively contracted nodes with two neighbors. The preprocessed graph has 410 nodes, 106 of which were OD-nodes, and 891 edges. We remark that although such preprocessing steps are conceptually easy, the data handling can be quite intricate in practice; for instance, our data included information on possible turnings of a line at road/rail crossings, which must be updated in the course of the preprocessing.

The OD-matrix was also modified. Nodes with zero traffic were removed. The original time horizon was one day, but we wanted to construct a line plan for the peak hour. We therefore scaled the matrix to 40% in an (admittedly rough) attempt to simulate afternoon traffic (3 p.m. to 6 p.m.). Note that the resulting matrix is still quite symmetric (the maximum difference between each of the two directions was 25) whereas a real afternoon OD-matrix would not be symmetric. The scaled OD-matrix had 4685 nonzeros and the total scaled travel demand was 42796.

All traveling times are measured in seconds and we always restricted the maximum length of a line to 55 edges. Since no data was available on line costs, we decided on $C_\ell = 10000$ (fixed costs) for each line ℓ and $c_e^i = 100$ (operating costs) for each edge e and mode i. Hence, we do not distinguish between costs of different modes (an unrealistic assumption in practice).

5.2 Experiments

Table 2 reports the results of several computational experiments with the data and implementation that we have described. All experiments were performed on a 3.4 GHz Pentium 4 machine running Linux. In the table, the *total traveling time* is $\boldsymbol{\tau}^{\mathrm{T}}\boldsymbol{y}$ and *total line cost* is $\boldsymbol{\gamma}^{\mathrm{T}}\boldsymbol{f}$, the *scaled* values are $(1-\lambda)\,\boldsymbol{\tau}^{\mathrm{T}}\boldsymbol{y}$ and $\lambda\,\boldsymbol{\gamma}^{\mathrm{T}}\boldsymbol{f}$, respectively; all four values refer to the LP relaxation (LP). The *LP objective value* is $\lambda\,\boldsymbol{\gamma}^{\mathrm{T}}\boldsymbol{f} + (1-\lambda)\,\boldsymbol{\tau}^{\mathrm{T}}\boldsymbol{y}$, the *integer objective value* refers to $\lambda\,(\boldsymbol{C}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{c}^{\mathrm{T}}\boldsymbol{f}) + (1-\lambda)\,\boldsymbol{\tau}^{\mathrm{T}}\boldsymbol{y}$. The last line in each block of results gives the number of active (i.e., nonzero) line and passenger variables, and the number of passenger transfers (first number) that were needed as well as the number of transfering passengers (second number). Note that we can compute transfers from passenger routes as an afterthought, although our optimization model is currently insensitive to them.

Let us point out explicitly that we do not claim that our results are already

Optimized LP solution – enumeration:		
total traveling time:	108,360,036.33	[scaled: 238,392.08]
total line cost:	233,776.86	[scaled: 233,262.55]
LP objective value:	471,654.63	
active line/pass. var.:	60/4879	transfers: 8777/64607
Optimized LP solution – randomized coloring – 5 trials:		
total traveling time:	108,396,741.75	[scaled: 238,472.83]
total line cost:	239,099.73	[scaled: 238,573.71]
LP objective value:	477,046.54	
active line/pass. var.:	61/4880	transfers: 9143/66546
Optimized LP solution – randomized coloring – 15 trials:		
total traveling time:	108,491,234.25	[scaled: 238,680.72]
total line cost:	237,422.50	[scaled: 236,900.17]
LP objective value:	475,580.88	
active line/pass. var.:	62/4885	transfers: 9387/68049
Optimized integer solution – greedy heuristic:		
total traveling time:	112,581,291.50	[scaled: 247,678.84]
total line cost:	287,060.90	[scaled: 286,429.37]
integer objective value:	818,491.68	
active line/pass. var.:	30/4767	transfers: 8638/60539
Reference LP solution:		
total traveling time:	105,269,846.00	[scaled: 231,593.66]
total line cost:	501,376.24	[scaled: 500,273.21]
LP objective value:	731,866.87	
active line/pass. var.:	61/4857	transfers: 8618/63310
Reference integer solution – greedy heuristic:		
total traveling time:	106,952,869.00	[scaled: 235,296.31]
total line cost:	562,964.54	[scaled: 561,726.02]
integer objective value:	1,213,221.49	
active line/pass. var.:	44/4814	transfers: 9509/70525

Table 2: Experimental Line Planning Results for $\lambda = 0.9978$.

practically significant; we only want to show that there is a potential to apply our methods to practical data. For example, our costs are not realistic. Therefore the frequencies that we compute cannot be compared to the ones used in practice. To allow some adaptation to our cost model, we let the frequencies of all lines be variable, in particular, the frequencies of the city railroad and regional train lines.

In our first experiment, we solved the LP relaxation (LP) of the Potsdam problem, pricing lines either by enumeration or by the randomized coloring method of Section 4.3, see top of Table 2. We set $\lambda = 0.9978$, which roughly balances the two parts of the objective function. The resulting LP had 5761 rows. Using enumeration, we obtained an optimal solution after 451 seconds and 283 iterations (i.e., solutions of the master LP), of which 15 were used

to price lines. The pricing problems needed a total time of 183 seconds of which most was used for the pricing of line paths. Hence, more than half of the time is spent for solving the master LPs.

We repeated this experiment using the randomized coloring algorithm with 5 and 15 trials for line pricing. With 5 trials, we needed 397 master LPs and 394 seconds in total; line pricing used only 99 seconds. One can see, however, that the objective is about 1% higher than for the enumeration variant. Using 15 trials resulted in 269 master LPs and 473 seconds in total. Line pricing now uses 265 seconds and the difference in the objective function relative to the enumeration variant is reduced to 0.8%. Hence, one can achieve a good approximation of the optimal value using randomized line pricing, although approaching the optimum solution comes at the cost of larger computation times.

We also investigated the passenger routing of our LP solution for the enumeration variant in more detail. To connect the 4685 OD-pairs only 4879 paths are needed, i.e., most OD-pairs are connected by a unique path. The total traveling time is 108,360,036.33 seconds, see Table 2. For comparison, when we ignore capacities and route all passengers between every OD-pair on the fastest path in the final line system, the total traveling time is 95,391,460 seconds. This is a relative difference of 12%. This seems to be an acceptable deviation.

In our second experiment, we computed two integer solutions for (LPP) associated with the parameter $\lambda = 0.9978$, as above. The first solution is obtained by rounding all nonzero $\boldsymbol{x}$-variables in the solution of the LP relaxation, computed with the enumeration variant, to 1. The (integer) objective of this rounded solution is 1,058,079.69, which leads to a gap of 55% compared to the LP relaxation value of 471,654.63. The second solution is obtained by the greedy algorithm described in Section 4.4, starting from the same LP solution (only lines for city buses, trams, and regional buses were removed). It has 30 lines (17 bus lines and 2 tram lines), down from 60 in the first solution, see Table 2; it took 1368 seconds to compute. The final (scaled) operating costs are 286,429.37, while the final fixed costs are $\lambda \cdot 300,000 = 299,340$. The integer objective of 818,491.68 has a gap of 42% with respect to the LP relaxation value of 471,654.63. Note that the results heavily depend on the cost structure: decreasing the fixed costs automatically reduces the gap. In our context, with high fixed costs, emphasis is put on reducing the number of lines (recall that the costs were artificial). The result obtained seems to be quite good, given that the original line system contained 27 bus lines and 4 tram lines; it seems unlikely that one can reduce the number much further. Furthermore, the lower bound of the LP relaxations is typically very weak for such fixed cost problems. Still, more research is needed to provide better lower bounds and primal solutions.

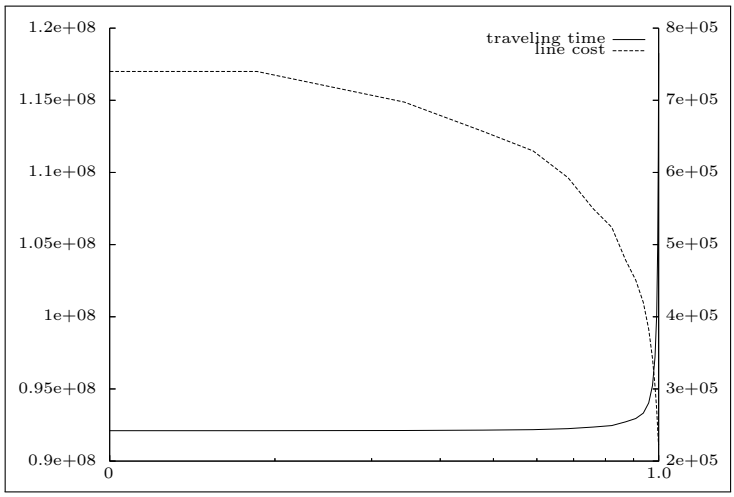


Figure 3: Total Traveling Time (solid, left axis) and Total Line Cost (dashed, right axis) in Dependence on λ (x-axis in logscale).

We compare the LP and integer solutions to "reference solutions" shown in the lower part of Table 2. The reference LP solution is obtained by fixing the paths of the original lines of Potsdam and then solving the resulting LP relaxation without generating new lines, but allowing the frequencies of the lines to change. The reference integer solution is obtained by applying the greedy heuristic to the reference LP solution. The results show that allowing the generation of new line paths reduces line costs in both cases to roughly 50% and the total objective to roughly 2/3 of the original values, while the total traveling time increases by a few percent. Hence, in these experiments the greedy algorithm has not changed the relative improvement obtained from optimizing lines.

Our third experiment investigates the influence of the parameter λ on the solution. We computed the solutions to the LP relaxation for 21 different values of λ_i, taking $\lambda_i = 1 - \big(1 - i/20\big)^4$, for $i = 0, \ldots, 20$. This collects increasingly more samples near $\lambda = 1$, a region where the total traveling time and the total line cost are about equal.

The results are plotted in Figure 3. This figure shows the total traveling time and the total line cost depending on λ. The extreme cases are as expected: For $\lambda = 0$, the line costs do not contribute to the objective and are therefore high, while the total traveling time is low. For $\lambda = 1$, only the total line cost contributes to the objective and is therefore minimized as much as possible at the cost of increasing the total traveling time. With increasing λ, the total line cost monotonically decreases, while the total traveling time increases. Note that each computed pair of total traveling time and line cost constitutes a Pareto optimal point, i.e., is not dominated by any other attainable combination. Conversely, any Pareto optimal solution of the LP

relaxation can be obtained as the solution for some $\lambda \in [0, 1]$, see, e.g., Ehrgott (2005) [15].

6 Conclusions

We proposed a new model for line planning in public transport that allows to generate lines dynamically and to freely route passengers according to the computed lines. The model allows to deal with manifold requirements from practice. We showed that line planning problems for a medium sized town can be solved within reasonable quality with integer programming techniques. Our computational results indicate significant optimization potentials. Our results on the polynomial time solvability of the LP relaxation for the case of logarithmic line lengths raises our hope that the model is suited to deal with larger problems as well.

Acknowledgment. We thank Volker Kaibel for pointing out Proposition 4.2.

References

[1] N. Alon, R. Yuster & U. Zwick. *Color-coding.* J. Assoc. Comput. Mach. **42**(4):844–856, 1995. Cited on page 146.

[2] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh & P. H. Vance. *Branch-and-price: Column generation for solving huge integer programs.* Oper. Res. **46**(3):316–329, 1998. Cited on page 143.

[3] A. Bouma & C. Oltrogge. *Linienplanung und Simulation für öffentliche Verkehrswege in Praxis und Theorie.* Eisenbahntechnische Rundschau **43**(6):369–378, 1994. Cited on page 137.

[4] M. R. Bussieck. *Optimal Lines in Public Rail Transport.* PhD thesis, Technische Universität Braunschweig, 1997. Cited on page 137.

[5] M. R. Bussieck, P. Kreuzer & U. T. Zimmermann. *Optimal lines for railway systems.* European J. Oper. Res. **96**(1):54–63, 1997. Cited on page 137.

[6] M. R. Bussieck, P. Kreuzer & U. T. Zimmermann. *Discrete optimization in public rail transport.* Math. Programming **1–3**(79B): 415–444, 1997. Cited on pages 136, 137.

[7] M. R. Bussieck, T. Lindner & M. E. Lübbecke. *A fast algorithm for near optimal line plans.* Math. Methods Oper. Res. **59**(2):205–220, 2004. Cited on page 138.

[8] A. Ceder & Y. Israeli. *Scheduling considerations in designing transit routes at the network level.* In Desrochers & Rousseau (1992) [13], pp. 113–136. ISBN 3-540-55634-6. Cited on page 137.

[9] A. Ceder & N. H. M. Wilson. *Bus network design.* Transportation Res. Part B **20**(4):331–344, 1986. Cited on page 137.

[10] M. T. Claessens, N. M. van Dijk & P. J. Zwaneveld. *Cost optimal allocation of rail passanger lines.* European J. Oper. Res. **110** (3):474–489, 1998. Cited on pages 137, 142.

[11] J. R. Correa, A. S. Schulz & N. E. Stier Moses. *Selfish routing in capacitated networks.* Math. Oper. Res. **29**:961–976, 2004. Cited on page 141.

[12] J. R. Daduna, I. Branco & J. M. P. Paixão, (Eds.). *Computer-Aided Transit Scheduling*, vol. 430 of *Lecture Notes in Economics and Mathematical Systems*, 1995. Springer Verlag, Berlin. ISBN 3-540-60193-7. Cited on page 155.

[13] M. Desrochers & J.-M. Rousseau, (Eds.). *Computer-Aided Transit Scheduling*, vol. 386 of *Lecture Notes in Economics and Mathematical Systems*, 1992. Springer Verlag, Berlin. ISBN 3-540-55634-6. Cited on page 154.

[14] D. Dubois, G. Bel & M. Llibre. *A set of methods in transportation network synthesis and analysis.* J. Oper. Res. Soc. **30**(9):797–808, 1979. Cited on page 137.

[15] M. Ehrgott. *Multicriteria Optimization.* Springer Verlag, Berlin, Berlin, 2nd edition, 2005. Cited on page 154.

[16] M. Garey & D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, New York, 1979. Cited on pages 144, 145.

[17] J.-W. H. M. Goossens, S. van Hoesel & L. G. Kroon. On solving multi-type line planning problems. METEOR Research Memorandum RM/02/009, Universiteit Maastricht, 2002. Cited on pages 138, 142.

[18] J.-W. H. M. Goossens, S. van Hoesel & L. G. Kroon. *A branch-and-cut approach for solving railway line-planning problems.* Transportation Sci. **38**(3):379–393, 2004. Cited on pages 138, 142.

[19] Y. Israeli & A. Ceder. *Transit route design using scheduling and multiobjective programming techniques.* In Daduna, Branco & Paixão (1995) [12], pp. 56–75. ISBN 3-540-60193-7. Cited on page 137.

[20] C. E. Mandl. *Evaluation and optimization of urban public transportation networks.* European J. Oper. Res. **5**:396–404, 1980. Cited on page 137.

[21] A. R. Odoni, J.-M. Rousseau & N. H. M. Wilson. *Models in urban and air transportation.* In S. M. Pollock et al., (Ed.), *Handbooks in Operations Research and Management Science*, vol. 6, chap. 5, pp.

107–150. Elsevier Science B. V., Amsterdam, 1994. Cited on pages 136, 137.

[22] U. Pape, Y.-S. Reinecke & E. Reinecke. *Line network planning.* In Daduna, Branco & Paixão (1995) [12], pp. 1–7. ISBN 3-540-60193-7. Cited on page 137.

[23] M. Petkovsek, H. S. Wilf & D. Zeilberger. $A = B$. A. K. Peters, Wellesley, Massachusetts, 1996. Cited on page 148.

[24] A. Schöbel & S. Scholl. *Line planning with minimal traveling time.* In L. G. Kroon & R. H. Möhring, (Eds.), *5th Workshop on Algorithmic Methods and Models for Optimization of Railways.* Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2005. Cited on pages 138, 141.

[25] S. Scholl. *Customer-Oriented Line Planning.* PhD thesis, Universität Göttingen, 2005. Cited on pages 138, 141.

[26] L. A. Silman, Z. Barzily & U. Passy. *Planning the route system for urban buses.* Comput. Oper. Res. **1**:201–211, 1974. Cited on page 137.

[27] H. Sonntag. *Ein heuristisches Verfahren zum Entwurf nachfrageorientierter Linienführung im öffentlichen Personennahverkehr.* Z. Oper. Res. **23**:B15–B31, 1979. Cited on page 137.

Index

C

D

H

I

K

L

M

N

R

S

T

Curriculum Vitae

Ralf Borndörfer

born on August 20, 1967, in Münster

married, two children, German

1973–1977	Elementary School in Oerlinghausen
1977–1986	Hans-Ehrenberg-Gymnasium in Bielefeld-Sennestadt
1986–1987	Military Service at the Armored Brigade 21 in Augustdorf
1987–1991	Studies of Mathematics with Economics at the University of Augsburg
Dec. 1991	Master's Degree in Mathematics with Economics at the University of Augsburg, Supervisor: Prof. Dr. Martin Grötschel
1992–1997	Teaching Assistant at the Technical University of Berlin
Apr. 1997	Scientific Assistant at the Zuse Institute Berlin
Mar. 1998	Ph.D. in Natural Sciences at the Technical University of Berlin, Supervisor: Prof. Dr. Dr. h.c. mult. Martin Grötschel
Jul. 2000	Foundation of Löbel & Borndörfer GbR
Mar. 2004	Extension to Löbel, Borndörfer & Weider GbR
Jan. 2007	Deputy Head of Department Optimization at the Zuse Institute Berlin

Berlin, 21.08.2009