Carnegie Mellon University
Tepper School of Business

Doctoral Dissertation

Essays on Postoptimality, Lift-and-Project, and Scheduling

Thiago Serra
April, 2018

Submitted to the Tepper School of Business in Partial Fulfillment of the Requirements for the Degree of Doctor in Operations Research

Dissertation Committee:
John N. Hooker (chair)
Egon Balas
Willem-Jan van Hoeve
Andrea Lodi

Abstract

This thesis offers methodological and computational contributions to integer and mixed-integer linear programming. The first area is postoptimality, in which we explore how decision diagrams can be used to compactly store and query near-optimal solutions. Our contribution is on characterizing a particular form of decision diagram relaxation, the sound decision diagram, where solutions worse than a given threshold are tolerated in order to obtain smaller decision diagrams. We introduce an operation that we denote as "sound reduction" that, if applied a sufficient number of times, leads to one among the sound decision diagrams of minimum size. The second area is lift-and-project, in which we explore alternative formulations of the cut generating linear program (CGLP) as well as the equivalence between lift-and-project cuts and intersection cuts. Our first contribution is the introduction of a new formulation, the reverse polar CGLP, where the conventional roles of the objective function and of the normalization constraint are switched. We show that cuts from CGLP optima always define supporting hyperplanes of the disjunctive hull and that, if the objective function minimizes the slack of the cut for a point in the interior of the disjunctive hull, then the resulting cut is a combination of facet-defining cuts separating a given fractional solution. Our second contribution in this area is a method to identify irregular lift-and-project cuts, which are cuts from non-split disjunctions that are not equivalent to intersection cuts from any basis of the linear relaxation. Based on the CGLP formulation, we present a mixed-integer formulation that determines if such an equivalence exists and we report computational results on several instances where many cuts from elementary t-branch disjunctions are not equivalent to cuts from t rows of the simplex tableau. The last area is an emergent scheduling application, in which we study a last-mile passenger transportation service using autonomous vehicles. Our contribution consists of an approach to solve this optimization problem within the required time frame. We show that a convenient restriction of this problem has optimal solutions with a certain structure, propose heuristics that preserve such structure for warm-starting a mixed-integer formulation, and report a significant improvement as a result.

Acknowledgements

I would like to thank Professors John Hooker and Egon Balas for sharing with me their research questions and academic experience as well as for offering invaluable feedback and opportunities throughout my time in the doctoral program. Likewise, I am grateful to Professors Willem-Jan van Hoeve and Andrea Lodi for agreeing to participate in the committee and, more importantly, for their interest and resourceful advice on particular projects as well as on academia in general. I am also grateful to Professors Yoshiko Wakabayashi, Arnaldo Vieira Moura, and Cláudio Leonardo Lucchesi for their guidance during my studies in Brazil and for their recommendation to join Carnegie Mellon. I have been very fortunate to work with and learn from so many inspiring scholars.

The work in this thesis and in other projects that I have participated while in the program has also benefited from the kind collaboration or advice from Doctors Gérard Cornuéjols, R. Ravi, François Margot, Fatma Kilinç-Karzan, Michael Perregaard, Pierre Bonami, Selvaprabu Nadarajah, David Bergman, André Augusto Ciré, Arvind Raghunathan, and Srikumar Ramalingam.

This has been a journey shared very closely with two colleagues in the program, Christian Tjandraatmadja and Aleksandr Kazachkov, to whom I am grateful for the collaborations, discussions, and insights on cutting planes, decision diagrams, and deep learning. I would like to thank many other colleagues, past and present, including Yang Jiao, Tarek Elgindy, Siddharth Singh, Wenting Yu, Diana Kotenko, Stelios Despotakis, Jeremy Karp, Sercan Yildiz, Negar Soheili, Vince Slaugh, Xin Wang, Leela Nageswaran, Joris Kinable, Gerdus Benade, Dabeen Lee, Nam Ho-Nguyen, Ryo Kimura, Amin Hosseininasab, Arash Haddadan, Neda Mirzaeian, Franco Berbeglia, Mehmet Aydemir, Bo Yang, Ziye Tang, and Danial Davarnia. I am also grateful for the amazing support from Lawrence Rapp and Laila Lee as well as from Professors Michael Trick, Alan Scheller-Wolf, and Nicola Secomandi.

I would like to thank my parents, Ana Maria and Joaquim Benedito, and my sister Ana Caroline for the affection, dedication, and encouragement that brought me here.

Ultimately, I would like to thank my daughter Isabel, my son Leonardo, and my wife Sabrina for giving me the best years of my life. They have offered me reason for happiness every day and hour, no matter how hard things were at times, and I cannot imagine how I would have made it without their companionship, love, and support. This thesis is dedicated to them.

Contents

Introduction

> “ The History of every major Galactic Civilization tends to pass through three distinct and recognizable phases, those of Survival, Inquiry and Sophistication, otherwise known as the How, Why, and Where phases. For instance, the first phase is characterized by the question ‘How can we eat?’ the second by the question ‘Why do we eat?’ and the third by the question ‘Where shall we have lunch?’ ”
>
> Douglas Adams, The Restaurant at the End of the Universe

Mathematical optimization has progressed considerably in the past decades. Problems that were once challenging are now solved reasonably fast thanks to better hardware and algorithms. For the problems that became easy, we may now think of possibilities beyond finding a single optimal solution and even use these problems to better understand the solving methods. For the problems that remained hard, it is sometimes possible to exploit structural properties to tackle hard familes of instances. We explore these different frontiers in this thesis, from performing postoptimality analysis on near-optimal solutions of integer programs to determining the regularity of lift-and-project cuts and to characterizing optimal solutions of a scheduling problem for warm-starting.

Postoptimality It is often useful in practice to explore near-optimal solutions of an integer programming problem, whereas the structure of a decision diagram facilitates rapid processing of a wide range of queries about the near-optimal solution space. Decision diagrams are rooted directed acyclic graphs mapping a Boolean function having finite domains with labeled paths towards true or false terminal nodes [Lee, 1959, Akers, 1978]. They can be efficiently minimized for a given order of variables by unifying nodes [Bryant, 1986b]. Decision diagrams have found varied uses in optimization [Bergman et al., 2016a], from supporting constraint propagation [Andersen et al., 2007a] to replacing the branching tree and deriving upper and lower bounds to some combinatorial problems [Bergman et al., 2016b]. They can describe solutions of optimization problems with finite and discrete domains, in which case arcs are labeled according to assignments of decision variables, weighted according to their impact in the objective function, and optimal solutions correspond to root-terminal paths minimizing or maximizing the sum of arc weights.

In Chapter 1, we show how all solutions within a given tolerance of the optimal value can be efficiently compressed in a weighted decision diagram. To obtain a more compact diagram, we exploit the property that such diagrams may become paradoxically smaller when they contain more solutions. In particular, we characterize sound decision diagrams, which innocuously admit some solutions that are worse than near-optimal. We describe a simple "sound reduction" operation that, when applied repeatedly in any order, yields a smallest possible sound diagram for a given problem instance. We find that sound reduction yields a structure that is typically far smaller than a tree that represents the same set of near-optimal solutions.

Lift-and-Project Lift-and-project cuts can be obtained by defining an elegant optimization problem over the space of valid inequalities for a given disjunction, the Cut Generating Linear Program (CGLP) [Balas et al., 1993]. A CGLP has two main ingredients: (i) an objective function, which invariably maximizes the violation with respect to a fractional solution $\bar{x}$; and (ii) a normalization constraint, which limits the scale in which cuts are represented. In the case of a split disjunction, Balas and Perregaard [2003] have shown that there is a correspondence between lift-and-project cuts obtained from basic solutions of the CGLP and intersection cuts from basic solutions of the linear relaxation, feasible or not, and thus also to Gomory fractional cuts [Gomory, 1958]. More recently, Balas and Kis [2016] have shown that such correspondence may also hold for some lift-and-project cuts from arbitrary disjunctions.

In Chapter 2, we propose the Reverse Polar CGLP (RP-CGLP), which switches the roles conventionally played by objective and normalization: violation with respect to $\bar{x}$ is fixed to a positive constant, whereas we minimize the slack for a point p that cannot be separated by the valid inequalities. Cuts from RP-CGLP optima define supporting hyperplanes of the immediate closure. When that closure is full-dimensional, the face defined by the cut lays on facets first intersected by a ray from $\bar{x}$ to p, all of which corresponding to cutting planes from RP-CGLP optima if p is an interior point. In fact, these are the cuts minimizing a ratio between the slack for p and the violation for $\bar{x}$, and we show that the same ratio in minimized by other proposed reformulations. Among those, RP-CGLP has the advantage that its feasible set does not depend on p, hence making it easier to generate other cuts for $\bar{x}$ by just changing p. We show how this formulation can be implemented on the tableau of the linear relaxation and report some computational results.

In Chapter 3, we explore the extent to which the equivalence between intersection cuts and lift-and-project cuts remains valid in the case of non-split disjunctions, in which case the lift-and-project cuts are denoted regular. First, we state a result that simplifies the verification of regularity for basic CGLP solutions from Balas and Kis [2016] and show that it can also be used with CGLP solutions that are not basic. Second, we introduce and prove the validity of a mixed-integer formulation that checks whether there is a regular CGLP solution for a given cut. Third, we describe a numerical procedure based on such formulation that verifies if a lift-and-project cut is regular or not. Finally, we present and analyze computational results on the regularity of cuts from not-split disjunctions for several instances.

Scheduling Last-mile transportation refers to any service that moves passengers from a hub of mass transportation, such as air, boat, bus, or train, to destinations, such as a home or an office.

In Chapter 4, we introduce the problem of scheduling passengers jointly on both services, with passengers sharing a car, van, or autonomous pod of limited capacity for the last-mile service. Passenger itineraries are determined so as to minimize total transit time for all passengers, with each passenger arriving at the her destination within a specified time window. The transit time includes the time spent traveling through both services and, possibly, waiting time for transferring between the services. We provide an integer linear programming formulation for this problem. Since this problem is NP-hard and instances of practical size are often difficult to solve, we study a restricted version where mass transportation trips are uniform, all passengers have time windows of a common size, and vehicles in the last-mile service visit one destination per trip. We prove that there is an optimal solution that sorts and groups passengers by their deadlines and, based on this result, we propose a constructive grouping heuristic and local search operators to generate high-quality solutions. The resulting groups are optimally scheduled in a few seconds using another integer formulation for the subproblem. Numerical results indicate that the solutions obtained by this heuristic are often close to optimal and that warm-starting an integer programming solver with such solutions decreases the overall computational times significantly.

Chapter 1

Compressing Sound Decision Diagrams of Integer Linear Programs

This chapter is based on the manuscript "Compact Representation of Near-Optimal Integer Programming Solutions" [Serra and Hooker, 2017], which is currently under review.

1.1 Introduction

An integer programming model contains a wealth of information about the phenomenon it represents. An optimal solution of the model, or even a set of optimal solutions, captures only a small portion of this information. In many applications, it is useful to probe the model more deeply to explore alternative solutions, particularly solutions that are suboptimal as measured by the objective function but attractive for other reasons.

For example, in a recent study [Camm, 2014] an integer programming (IP) model was formulated to relocate distribution centers across Europe. In the absence of reliable estimates for fixed costs, the client opted for a suboptimal solution that relocated one rather than three distribution centers for only a 0.4% increase over the optimal cost. One may also wish to know which decisions are invariant across all near-optimal solutions. This was a key question in a nature reserve planning study [Arthur et al., 1997] that sought to identify areas that are critical to protect native species. In addition, there are applications that require the solution of minor variations of a problem. In some combinatorial auctions [Goetzendorff et al., 2015], for example, a winners determination problem is first solved to maximize the sum of winning bids, and then re-solved with each winner removed by fixing certain variables to zero.

In general, one may wish to know which solutions are optimal or near-optimal when certain variables are fixed to desired values, or which values a given variable can take without sacrificing near-optimality. One may also wish to determine how much a cost coefficient can be perturbed without changing the optimal cost more than a certain amount.

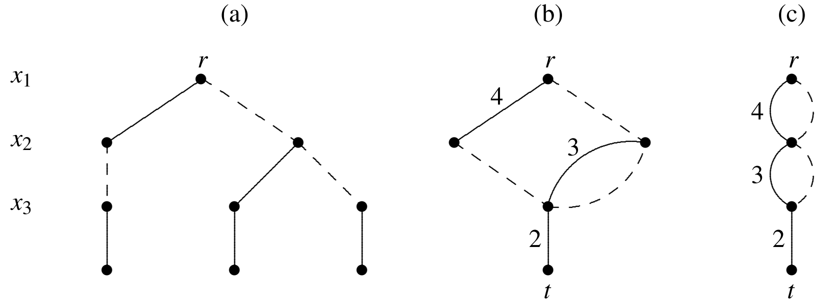


Figure 1.1: (a) Branching tree for near-optimal solutions of (1.1). (b) Reduced weighted decision diagram representing the same solutions. (c) Sound decision diagram for (1.1).

These questions can be answered if the space of near-optimal solutions is compactly represented in a transparent data structure; that is, a data structure that can be efficiently queried to find near-optimal (or optimal) solutions that satisfy desired properties. In fact, the task of solving an IP model can be more generally conceived as the process of transforming an opaque data structure to a transparent data structure. The constraint set and objective function comprise an opaque structure that defines the problem but does not make good solutions apparent. A conventional solver transforms the problem statement into a very simple transparent structure: an explicit list of one or more optimal solutions. The ideal would be to derive a more general data structure that compactly but transparently represents the space of near-optimal solutions and how they relate to each other.

We propose a *weighted decision diagram* for this purpose. Binary and multivalued decision diagrams have long been used for circuit design, formal verification, and other purposes [Akers, 1978, Bryant, 1986a, Hu, 1996, Lee, 1959, Wegener, 2000], but they can also compactly represent solutions of a discrete optimization problem [Andersen et al., 2007a, Bergman et al., 2016b, Hadžić and Hooker, 2006b, Hoda et al., 2010]. A *weighted* decision diagram represents the objective function values as well. Such a diagram can be built to represent only near-optimal solutions, and it can can be easily queried for solutions that satisfy desired properties. This is because solutions correspond straightforwardly to paths in the diagram, and their objective function values to the length of the paths.

A simple example illustrates the idea. The IP problem

$$\begin{array}{ll} \text{minimize} & 4x_1 + 3x_2 + 2x_3 \\ \text{subject to} & x_1 + x_3 \geq 1, \;\; x_2 + x_3 \geq 1, \;\; x_1 + x_2 + x_3 \leq 2 \\ & x_1, x_2, x_3 \in \{0, 1\} \end{array} \tag{1.1}$$

has optimal value 2. The branching tree of Fig. 1.1(a) represents the three feasible solutions that have a value within 4 of the optimum, namely $(x_1, x_2, x_3) = (1, 0, 1)$, $(0, 1, 1)$, $(0, 0, 1)$. A dashed arc represents setting $x_j = 0$, and a solid arc represents setting $x_j = 1$. Figure 1.1(b) is a decision diagram that represents the same solutions. The solid arcs are assigned weights (lengths) equal to the corresponding objective function coefficients, while the dashed arcs have length zero. Each path from the root r to the terminus t represents a feasible solution with cost at most 6, where the cost of the solution is the length of the path. The decision diagram is *reduced*, meaning that it is the smallest diagram that represents this set of solutions. It is well known that, for a given ordering of the variables, there is a unique reduced diagram representing a given set of solutions [Bryant, 1986a].

Although reduced decision diagrams tend to provide a much more compact representation than a branching tree, they can nonetheless grow rapidly. To address this issue, we take advantage of the fact that modifying a diagram to represent a larger solution set can, paradoxically, result in a smaller diagram. We adopt the concept of a *sound* decision diagram, introduced by Hadžić and Hooker [2007], which is a diagram that represents all near-optimal solutions along with some *spurious* solutions whose objective function values are worse than near-optimal. The spurious solutions may be feasible or infeasible. By judiciously admitting spurious solutions into the diagram, one can significantly reduce its size while maintaining soundness of the near-optimal solution set.

In particular, we show that a certain *sound reduction* operation, which replaces a pair of nodes with a single node, yields a smaller sound diagram. Our main theoretical result is that repeated application of sound reduction operations, in any order, results in a smallest possible sound diagram for a given problem and variable ordering. It is smallest in the sense that it has a minimum number of arcs and a minimum number of nodes. We call such a diagram *sound reduced*. A problem may have multiple sound-reduced diagrams, but they all have the same minimum size.

Sound diagrams have several advantages for postoptimality analysis. Aside from their smaller size, they allow for easy extraction of near-optimal solutions. One need only to enumerate paths in the diagram while discarding those that represent spurious solutions, which are easily identified by the fact that their values are too far from the optimum. In order to infer that solutions are spurious while constructing and when querying such diagrams, the optimal value is first obtained by solving the problem with a conventional solver.

For example, Fig. 1.1(c) illustrates a sound diagram for problem (1.1). It represents the three solutions within 4 of the optimal value, plus a spurious solution $(x_1, x_2, x_3) = (1, 1, 1)$ that is discarded because its value is greater than 6. This solution happens to be infeasible, but it is not necessary to check feasibility, which is time-consuming. It is only necessary to compute the path length.

A further advantage of sound diagrams is that the presence of spurious solutions has no effect whatever on the implementation or complexity of many types of postoptimality analysis. It is enough that the diagram represent all near-optimal solutions.

We begin below with a review of related work, followed by four sections that develop the underlying theory of sound diagrams. Section 1.3 introduces some basic concepts and properties of decision diagrams. Section 1.4 develops the idea of soundness and shows that it is a useful concept only when suboptimal (as well as optimal) solutions are represented. Section 1.5 proves the main result that sound reduction yields a sound diagram of minimum size. It also shows by counterexample that there need not be a unique sound-reduced diagram for a given problem. Section 1.6 explains why it is not practical to admit superoptimal solutions into sound diagrams, even though this may result in smaller diagrams.

The remaining sections apply the theory of sound diagrams to integer programming. Section 1.7 presents an algorithm that constructs a sound diagram for a given integer programming problem, assuming that the optimal value has been obtained by solving the problem with a conventional solver. Section 1.8 shows how to introduce sound reduction into the algorithm, thereby obtaining a smallest possible sound diagram for the problem. Section 1.9 then describes several types of postoptimality analysis that can efficiently be performed on a sound diagram. Section 1.10 reports computational tests that measure how compactly sound diagrams can represent near-optimal solutions, and the time required to compute the diagrams. Based on instances from MIPLIB, it is found that decision diagrams represent near-optimal solutions much more compactly than a branching tree, and that in most instances, sound reduction substantially reduces the size of the diagrams. The chapter concludes with a summary and agenda for future research.

1.2 Related Work

To our knowledge, no previous study addresses the issue of how to represent near-optimal solutions of IP problems in a compact and transparent fashion. A few papers have proposed methods for generating multiple solutions. Scatter search is used in Glover et al. [2000] to generate a set of diverse optimal and near-optimal solutions of mixed integer programming (MIP) problems. However, since it is a heuristic method, it does not obtain an exhaustive set of solutions for any given optimality tolerance. Diverse solutions of an MIP problem have also been obtained by solving a sequence of MIP models, beginning with the given problem, in which each seeks a solution different from the previous ones. This approach is investigated in Greistorfer et al. [2008], where it is compared with solving a much larger model that obtains multiple solutions simultaneously. However, neither method is scalable, as there may be a very large number of near-optimal solutions.

The "one-tree" method in Danna et al. [2007] generates a collection of optimal or near-optimal solutions of a mixed-integer programming problem by extending a branching tree that is used to solve the problem. While possible, the collection is not intended to be exhaustive, and there is no indication of how to represent the collection compactly or query more easily. A "branch-and-count" method is presented in Achterberg et al. [2008] for generating all *feasible* solutions of an IP problem, based on the identification of "unrestricted subtrees" of the branching tree. These are subtrees in which all values of the unfixed variables are feasible. We use a similar device as part of our mechanism for constructing sound decision diagrams. However, we focus on compact representation of near-optimal solutions.

The commercial solver CPLEX has offered a "solution pool" feature since version 11.0 [IBM Support, 2010] that relies on the one-tree method. The solution pool has been supported by the the GAMS modeling system since version 22.6 [GAMS Support Wiki, 2013]. By contrast, postoptimality software based on decision diagrams operates apart from the solution method, requiring only the optimal value from the solver. It also differs by generating an exhaustive set of near-optimal solutions and organizing them in a decision diagram that is convenient for postoptimality analysis.

Integer programming sensitivity analysis has been investigated for some time, as for example in Dawande and Hooker [2000], Gamrath et al. [2015], Geoffrion and Nauss [1977], Holm and Klein [1984], Kilinç-Karzan et al. [2009], Schrage and Wolsey [1985], Van Hoesel and Wagelmans [1999]. Sound decision diagrams can be used to analyze sensitivity to perturbations in objective coefficients, because these appear as arc lengths in the diagram, and we show how to do so. However, our main interest here is in probing the near-optimal solution set that results from the original problem data.

Decision diagrams were first proposed for IP postoptimality analysis in Hadžić and Hooker [2006a], and the concept of a sound diagram was introduced in Hadžić and Hooker [2007]. The present chapter extends this work in several ways. It proves several properties of sound diagrams, introduces the sound reduction operation, and proves that sound reduction yields a sound diagram of minimum size. It also presents algorithms for generating sound-reduced diagrams for IP problems and conducting postoptimality analysis on these diagrams, as well as reporting computational tests on the representational efficiency of the diagrams.

1.3 Decision Diagrams for Discrete Optimization Problems

For our purposes, we associate a decision diagram with a discrete optimization problem of the form

$$\min\{f(x) \mid x \in S\} \tag{P}$$

where $S \subseteq S_1 \times \ldots \times S_n$ and each variable domain S_j is finite. A *decision diagram* associated with (P) is a multigraph $D = (U, A, \ell)$ with the following properties:

- The node set U is partitioned $U = U_1 \cup \cdots \cup U_{n+1}$, where $U_1 = \{r\}$ and $U_{n+1} = \{t\}$. We say r is the *root node*, t the *terminal node*, and U_j is *layer j* of D for each j.
- The arc set A is partitioned $A = A_1 \cup \cdots \cup A_n$, where each arc in A_j connects a node in U_j with a node in U_{j+1}, for $j = 1, \ldots, n$.
- Each arc $a \in A_j$ has a *label* $\ell(a) \in S_j$ for $j = 1, \ldots, n$, representing a value assigned to variable x_j. The arcs leaving a given node must have distinct labels.

The labels on each path p of D from r to t represent an assignment to x, which we denote $x(p)$. We let $\mathrm{Sol}(D)$ denote the set of solutions represented by the r–t paths. We say that D *exactly represents* S when $\mathrm{Sol}(D) = S$.

A *weighted decision diagram* associated with (P) is a multigraph $D(U, A, \ell, w)$ that satisfies the above properties, plus the following:

- Each arc $a \in A$ has a *weight* $w(a)$, such that $\sum_{a \in p} w(a) = f(x(p))$ for any r–t path p of D. Thus the total weight $w(p)$ of an r–t path p is the objective function value of the corresponding solution.

A weighted decision diagram associated with problem (P) exactly represents (P) when $\mathrm{Sol}(D) = S$. In this case, the optimal value z^* of (P) is the weight of any minimum-weight r–t path of D, and the optimal solutions of (P) are those corresponding to minimum-weight r–t paths. From here out, we will refer to a weighted decision diagram simply as a decision diagram, and to a diagram without weights as an unweighted decision diagram.

An unweighted decision diagram D is *reduced* when redundancy is removed. To make this precise, let a *suffix* of $u \in U_j$ be any assignment to $x_j, \ldots, x_n$ represented by a u–t path in D, and let $\mathrm{Suf}(u)$ be the set of suffixes of u. Then D is reduced when $\mathrm{Suf}(u) \neq \mathrm{Suf}(v)$ for all $u, v \in U_j$ with $u \neq v$ and all $j = 1, \ldots, n$. As noted earlier, for any fixed variable ordering, there is a unique reduced unweighted decision diagram that exactly represents a given feasible set S, and this diagram is the smallest one that exactly represents S [Bryant, 1986a].

Given a path π from a node in layer j to a node in layer k, it will be convenient to denote by $x(\pi)$ the assignment to $(x_j, \ldots, x_{k-1})$ indicated by the labels on path π. We also let $x_i(\pi)$ denote the the assignment to x_i in particular, and we let $w(\pi)$ denote the weight of π. A summary of notation used throughout the chapter can be found in Table 1.1.

The following simple property of decision diagrams will be useful.

Lemma 1.1. *Given any pair of distinct nodes u, v in layer j of a decision diagram, let π be an r–u path and ρ an r–v path. Then $x(\pi) \neq x(\rho)$.*

Proof. If $x(\pi) = x(\rho)$, then in particular $x_1(\pi) = x_1(\rho)$. This implies that π and ρ lead from r to the same node u in U_2, since distinct arcs leaving r must have distinct labels. Arguing inductively, π and ρ lead from the same node in U_k to the same node in U_{k+1} for $k = 1, \ldots, j-1$. This implies that $u = v$, contrary to hypothesis. □ □

1.4 Sound Decision Diagrams

We are interested in constructing decision diagrams that represent near-optimal solutions of (P). Let x be a Δ-*optimal* solution of (P) when $x \in S$ and $f(x) \leq z^* + \Delta$, for $\Delta \geq 0$. We denote by

Table 1.1: List of symbols.

r	root node of a decision diagram
t	terminal node of a decision diagram
U_j	set of nodes in layer j of a diagram
A_j	set of arcs connecting nodes in U_j with nodes in U_{j+1}
$\ell(a)$	label of arc a, representing value of x_j if $a \in A_j$
$w(a)$	weight (cost, length) of arc a
$x(p)$	assignment to x represented by r–t path p
$w(p)$	weight of r–t path p
$x(\pi)$	assignment to $x_j, \ldots, x_{k-1}$ represented by u–v path π ($u \in U_j$, $v \in U_k$)
$x_i(\pi)$	assignment to x_i represented by π, where $j \leq i < k$
$w(\pi)$	weight of path π
$w(u, u')$	weight of minimum-weight path from node u to node u'
$\mathrm{Sol}(D)$	set of solutions represented by r–t paths in diagram D
z^*	optimal value of problem (P)
$\mathrm{P}(\Delta)$	problem of finding Δ-optimal solutions of (P)
$S(\Delta)$	set of Δ-optimal solutions of (P)
$\mathrm{ILP}(\Delta)$	problem of finding Δ-optimal solutions of (ILP)
$\mathrm{Pre}(u)$	set of prefixes of node u
$\mathrm{Suf}(u)$	set of suffixes of node u
$\mathrm{Suf}_\Delta(u)$	set of Δ-suffices of node u of a diagram D; i.e., set of suffixes of u that are part of some Δ-optimal solution represented by D
$\mathrm{lhs}.u$	left-hand-side state at node u
$\mathrm{LCDS}_j[u, v]$	weight of least-cost differing suffix when reducing u into v
W	maximum width of (number of nodes in) layers of a diagram
$S_{\max}$	size of largest variable domain

$S(\Delta)$ is the set of Δ-optimal solutions of (P), so that $S(0)$ is the set of optimal solutions. We let $\mathrm{P}(\Delta)$ denote the the problem of finding Δ-optimal solutions of (P).

We say that D exactly represents $\mathrm{P}(\Delta)$ when $\mathrm{Sol}(D) = S(\Delta)$. Since such a decision diagram can be quite large, we wish to identify smaller diagrams that approximately represent $\mathrm{P}(\Delta)$. We therefore study decision diagrams that are *sound* for $\mathrm{P}(\Delta)$, which represent a superset of $S(\Delta)$. Specifically, D is sound when

$$S(\Delta) = \mathrm{Sol}(D) \cap \{x \in S_1 \times \cdots \times S_n \mid f(x) \leq z^* + \Delta\}$$

Thus a sound diagram can represent, in addition to Δ-optimal solutions, feasible and infeasible solutions that are worse than Δ-optimal. We refer to these as *spurious* solutions. A *proper* sound diagram represents a proper superset of the Δ-optimal solutions and therefore represents some spurious solutions.

We prefer a sound diagram D that is *minimal* for P(Δ), meaning that every node of D, and every arc of D, lies on some r–t path that represents a solution in $S(\Delta)$. If a sound diagram is not minimal, nodes and/or arcs can be removed without destroying soundness. Since their removal does not enlarge the set represented by the diagram, we obtain a smaller diagram that is an equally accurate approximation of $S(\Delta)$.

It is easy to check whether a node or arc can be removed while preserving soundness. For any two nodes u, u' in different layers of D, let $w(u, u')$ be the weight of a minimum-weight path from u to u' (infinite if there is no path). Then node u can be removed if and only if

$$w(r, u) + w(u, t) > z^* + \Delta$$

An arc a connecting $u \in U_j$ with $u' \in U_{j+1}$ can be removed if and only if

$$w(r, u) + w(a) + w(u', t) > z^* + \Delta \tag{1.2}$$

Interestingly, a proper sound diagram for P(0) is never minimal. This implies that there is no point in considering sound diagrams to represent the set of optimal solutions. They are useful only for representing sets of near-optimal solutions.

Theorem 1.2. *No proper sound decision diagram is minimal for P*(0)*.*

Proof. Suppose to the contrary that diagram D is a minimal for P(0) and contains a suboptimal r–t path p. For any given node u in p, let $\pi(u)$ be the portion of p from r to u. Select a node u^* in p that maximizes the number of arcs in $\pi(u^*)$ subject to the condition that $\pi(u^*)$ is part of some optimal (minimum-weight) r–t path in D (Fig. 1.2). We note that $u^* \notin U_{n+1}$, since otherwise p would be an optimal r–t path. Thus p contains an arc a from u^* to some node u'. Furthermore, $u^* \notin U_1$ since otherwise arc a would prevent D from being minimal. Now since D is minimal, arc a belongs to some optimal r–t path, which we may suppose consists of π', a, and σ'. Hence, $\pi(u^*)$ and π' are both optimal r–u^* paths, and thus the r–t path consisting of $\pi(u')$ and σ' is also optimal. This implies that $\pi(u')$, which contains one more arc than $\pi(u^*)$, is part of an optimal r–t path, contrary to the definition of u^*. □

□

The following property of sound diagrams is easily verified.

Lemma 1.3. *If a decision diagram D is sound for P*(Δ)*, then D is sound for P*(δ) *for any $\delta \in [0, \Delta]$.*

Thus the set of sound decision diagrams of P(Δ) is a subset of that of P(δ) for any $\delta \in [0, \Delta]$.

Corollary 1.4. *The size of a smallest sound diagram for P*(Δ)*, as measured by the number of arcs or the number of nodes, is monotone nondecreasing in Δ.*

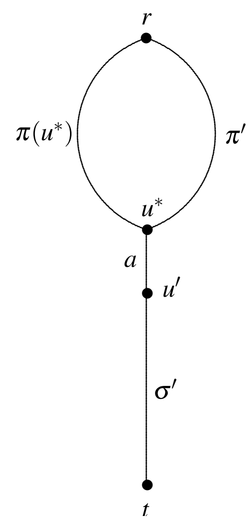


Figure 1.2: Illustration of the proof of Theorem 1.2.

1.5 Sound Reduction

Sound reduction is a tool for reducing the size of a given sound diagram, generally at the cost of increasing the number of spurious solutions it represents. Given distinct nodes $u, v \in U_j$ for $1 < j \leq n$, we can *sound-reduce* u *into* v when diverting to v the arcs coming into u, and deleting u from the diagram, removes no Δ-optimal solutions and adds only spurious solutions. Thus sound reduction removes at least one node without destroying soundness. In fact, we will see that repeated sound reduction yields the smallest sound diagram for a given Δ.

Let a Δ-*suffix* of node $u \in U_j$ be any suffix in $\mathrm{Suf}(u)$ that is part of a Δ-optimal solution, and let $\mathrm{Suf}_\Delta(u)$ be the set of Δ suffixes of u. Also let a *prefix* of u be any assignment to $(x_1, \ldots, x_{j-1})$ represented by an r–u path, and let $\mathrm{Pre}(u)$ be the set of prefixes of u. Then u can be sound-reduced into v if:

$$\mathrm{Suf}_\Delta(u) \subseteq \mathrm{Suf}(v) \tag{1.3}$$

$$w(\pi) + w(\sigma) > z^* + \Delta \;\text{ when } x(\pi) \in \mathrm{Pre}(u) \text{ and } x(\sigma) \in \mathrm{Suf}(v) \setminus \mathrm{Suf}(u) \tag{1.4}$$

Sound reduction is accomplished as follows. For every arc a from some node $q \in U_{j-1}$ to u, remove a and create an arc from q to v with label $\ell(a)$ and weight $w(a)$. Then remove u and any successor of u that is disconnected from r. That is, remove u and any successor u' of u for which all r–u' paths in D contain u.

Condition (1.3) ensures that any Δ-optimal solution whose r–t path passes through u remains in the diagram after sound reduction, with the same cost. Condition (1.4) ensures that only spurious solutions are added to the diagram. So we have,

Theorem 1.5. *Sound reduction preserves soundness.*

Figure 1.3 illustrates sound reduction. Figure 1.3(a) is a reduced diagram that is sound for a problem P(Δ) with $z^* = 2$ and $\Delta = 6$. Dashed arcs have label 0 and weight 0, and solid arcs have label 1 and weights as shown. Figure 1.3(b) shows the result of sound-reducing node u_1 into node v_1. Condition (1.3) is satisfied because $\mathrm{Suf}_\Delta(u_1) = \{(1,1,0,0)\} \subseteq \{(1,1,0,0),(1,1,0,1)\} = \mathrm{Suf}(v_1)$. Condition (1.4) is satisfied because $\mathrm{Pre}(u_1) = \{(1,1)\}$, $\mathrm{Suf}(v_1) \setminus \mathrm{Suf}(u_1) = \{(1,1,0,1)\}$, and the solution $(x_1,\ldots,x_6) = (1,1,1,1,0,1)$ has cost $9 > z^* + \Delta$. We could have also reduced u_2 into v_2, u_3 into v_3, or u_3 into q.

A sound diagram for P(Δ) is *sound-reduced* if no further sound reductions are possible. We can show that a minimal sound-reduced diagram is the smallest diagram that is sound for P(Δ). For example, the diagram in Fig. 1.3(b) is sound-reduced, and it is in fact the smallest sound diagram for P(Δ) with $\Delta = 6$. Establishing this result requires two lemmas.

Lemma 1.6. *Given a sound-reduced diagram D for P(Δ), any two distinct nodes $u, v \in U_j$ of D satisfy* $\mathrm{Suf}_\Delta(u) \neq \mathrm{Suf}_\Delta(v)$.

Proof. Suppose to the contrary that $\mathrm{Suf}_\Delta(u) = \mathrm{Suf}_\Delta(v)$, and assume without loss of generality that $w(r,u) \geq w(r,v)$. We will show that u can be sound-reduced into v, contrary to hypothesis. Condition (1.3) for sound reduction is obviously satisfied. Also condition (1.4) is satisfied, because if $x(\pi) \in \mathrm{Pre}(u)$ and $x(\sigma) \in \mathrm{Suf}(v) \setminus \mathrm{Suf}(u)$, then $x(\sigma) \notin \mathrm{Suf}_\Delta(u)$, and therefore $x(\sigma) \notin \mathrm{Suf}_\Delta(v)$. This implies $w(r,v) + w(\sigma) > z^* + \Delta$. But $w(\pi) + w(\sigma) \geq w(r,u) + w(\sigma) \geq w(r,v) + w(\sigma)$, and (1.4) follows. □ □

Lemma 1.7. *Let D be a minimal sound-reduced diagram for P(Δ). For any node u in layer j of D, and any other diagram D' with the same variable ordering that is sound for P(Δ), there is a*

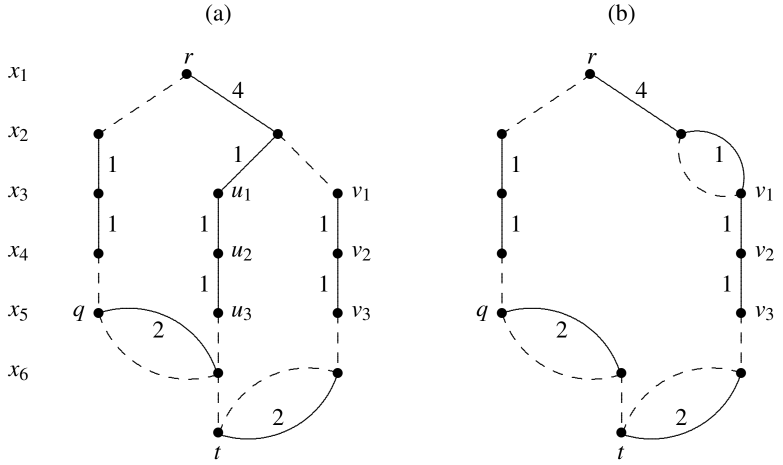


Figure 1.3: Reduced (a) and sound-reduced (b) decision diagrams for $z^* = 2$ and $\Delta = 6$.

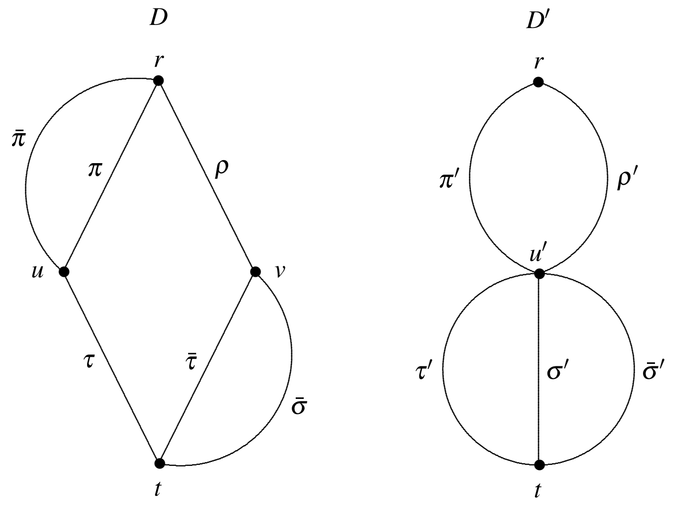


Figure 1.4: Illustration of the proof of Lemma 1.7.

node u' in layer j of D' with $\mathrm{Suf}_\Delta(u) = \mathrm{Suf}_\Delta(u')$.

Proof. Suppose to the contrary that there is a node u in layer j of D, and some sound diagram D' in which layer j contains no node with the same Δ-suffixes as u. We will show that D must then contain a node v into which u can be sound-reduced, contrary to hypothesis.

Let π be a minimum-weight r–u path in D. Since D is minimal, node u belongs to some path that represents a Δ-optimal solution. So since π is a minimum-weight r–u path, $x(\pi)$ is the prefix of some Δ-optimal solution. Thus since D' is sound for $\mathrm{P}(\Delta)$, layer j of D' must contain a node u' and an r–u' path π' with $x(\pi') = x(\pi)$. Since every Δ-optimal solution represented by D is also represented by D', we have that $\mathrm{Suf}_\Delta(u) \subseteq \mathrm{Suf}_\Delta(u')$, due to the fact that π is a minimum-weight path. However, by hypothesis $\mathrm{Suf}_\Delta(u) \neq \mathrm{Suf}_\Delta(u')$, and so we have $\mathrm{Suf}_\Delta(u') \setminus \mathrm{Suf}_\Delta(u) \neq \emptyset$.

Now consider any u'–t path σ' for which $x(\sigma') \in \mathrm{Suf}_\Delta(u') \setminus \mathrm{Suf}_\Delta(u)$. This implies that $x(\sigma')$ is the suffix of some Δ-optimal solution, and so there must be an r–u' path ρ' with

$$w(\rho') + w(\sigma') \leq z^* + \Delta \tag{1.5}$$

However, we can see as follows that $(x(\pi'), x(\sigma'))$ is not a Δ-optimal solution. Note that by Lemma 1.1, π is the only path in D representing $x(\pi) = x(\pi')$. Thus if $(x(\pi'), x(\sigma'))$ were Δ-optimal, the soundness of D would imply that $x(\sigma') = x(\sigma) \in \mathrm{Suf}_\Delta(u)$ for some u–t path σ, which contradicts the fact that $x(\sigma') \in \mathrm{Suf}_\Delta(u') \setminus \mathrm{Suf}_\Delta(u)$. So $(x(\pi'), x(\sigma'))$ is not Δ-optimal, which means $w(\pi') + w(\sigma') > z^* + \Delta$. This and (1.5) imply $w(\rho') < w(\pi')$. But (1.5) also implies that the sound diagram D must contain a node v and an r–v path ρ with $x(\rho') = x(\rho)$, so that $w(\rho) < w(\pi)$. Since π is a minimum-weight r–u path, this implies $u \neq v$.

We now show that u can be sound-reduced into v by verifying conditions (1.3) and (1.4). To show (1.3), consider any u–t path τ with $x(\tau) \in \mathrm{Suf}_\Delta(u)$. Since π is a minimum-weight r–u path, $(x(\pi), x(\tau))$ is a Δ-optimal solution. Now since D' is sound for $\mathrm{P}(\Delta)$ and $x(\pi) = x(\pi')$, there is a u'–t path τ' in D' for which $(x(\pi'), x(\tau'))$ is Δ-optimal. This means $(x(\rho'), x(\tau'))$ is Δ-optimal because $w(\rho') < w(\pi')$, which implies that $(x(\rho), x(\tau))$ is Δ-optimal. Since by Lemma 1.1, ρ is the only path representing $x(\rho)$, there must be a v–t path $\bar{\tau}$ with $x(\bar{\tau}) = x(\tau)$ and $x(\bar{\tau}) \in \mathrm{Suf}_\Delta(v)$. This implies $x(\tau) \in \mathrm{Suf}(v)$ and (1.3).

Finally, to show (1.4), let $\bar{\pi}$ be an r–u path with $x(\bar{\pi}) \in \mathrm{Pre}(u)$, and let $\bar{\sigma}$ be a v–t path with $x(\bar{\sigma}) \in \mathrm{Suf}(v) \setminus \mathrm{Suf}(u)$. Note that if $w(\rho) + w(\bar{\sigma}) > z^* + \Delta$, then since $w(\bar{\pi}) \geq w(\pi) > w(\rho)$, we have $w(\bar{\pi}) + w(\bar{\sigma}) > z^* + \Delta$, and (1.4) follows. We may therefore suppose $w(\rho) + w(\bar{\sigma}) \leq z^* + \Delta$, which means that $(x(\rho), x(\bar{\sigma}))$ is Δ-optimal because D is sound. Since D' is sound and $x(\rho) = x(\rho')$, by Lemma 1.1 there must be a u'–t path $\bar{\sigma}'$ for which $x(\bar{\sigma}') = x(\bar{\sigma})$ and $(x(\rho'), x(\bar{\sigma}'))$ is Δ-optimal. This means that D' represents the solution $(x(\pi'), x(\bar{\sigma}'))$, which is the same as $(x(\pi), x(\bar{\sigma}))$. But since $x(\bar{\sigma}) \notin \mathrm{Suf}(u)$, D does not represent the solution $(x(\pi), x(\bar{\sigma}))$, which therefore cannot be Δ-optimal. Thus since D' represents this solution, it must be spurious, and we have $w(\pi) + w(\bar{\sigma}) > z^* + \Delta$. This implies $w(\bar{\pi}) + w(\bar{\sigma}) > z^* + \Delta$ and (1.4). □ □

Theorem 1.8. *A sound decision diagram D for $P(\Delta)$ has a minimum number of nodes and a minimum number of arcs, among diagrams that are sound for $P(\Delta)$ and have the same variable ordering, if and only if D is minimal and sound-reduced.*

Proof. If D is not minimal, we can remove one or more nodes or arcs, and if D is not sound-reduced, we can remove at least one node. Thus D is minimal and sound-reduced if it has a minimum number of nodes and arcs.

To prove the converse, suppose D is minimal and sound-reduced. Due to Lemma 1.6, all nodes in any given layer j of D have sets of Δ-suffixes. By Lemma 1.7, these distinct sets of Δ-suffixes exist for nodes in layer j of any sound diagram for $\mathrm{P}(\Delta)$. Thus any sound diagram for $\mathrm{P}(\Delta)$ has at least as many nodes as D. Furthermore, the minimality of D implies that any arc a leaving a node u in layer j of D is part of some Δ-optimal solution. Given any diagram D' that is sound for $\mathrm{P}(\Delta)$, the node u' in layer j of D' with $\mathrm{Suf}_\Delta(u') = \mathrm{Suf}_\Delta(u)$ must have an outgoing arc with the same label as a. Thus D' has at least as many arcs as D. □ □

Although all sound-reduced diagrams for a given $\mathrm{P}(\Delta)$ have minimum size, they are not necessarily identical. For example, while all sequences of sound reductions of Fig. 1.3(a) terminate in the same diagram Fig. 1.3(b), this is not the case for the slightly different diagram of Fig. 1.5(a). Sound-reducing u_3 into q yields the sound-reduced diagram of Fig. 1.5(b), and sound-reducing u_3 into v_3 yields Fig. 1.5(c). Both diagrams are of minimum size and satisfy Lemma 1.7, but they are distinct.

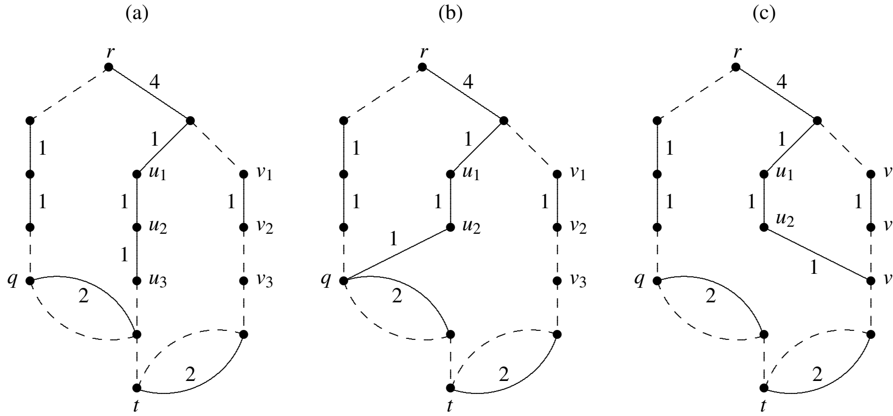


Figure 1.5: Distinct sound-reduced diagrams (b) and (c) obtained from diagram (a), where $z^* = 2$ and $\Delta = 6$.

1.6 Two-Sided Soundness

A sound diagram is permitted to represent feasible and infeasible solutions that are worse than Δ-optimal. A natural question is whether it would be useful to allow (infeasible) solutions that are *better than optimal.* Superoptimal solutions, like solutions that are worse than Δ-optimal, can be filtered out during postoptimality analysis by examining only objective function values.

Since such a diagram D requires excluding solutions with values on either side of the interval $[z^*, z^* + \Delta]$, we will say that it has *two-sided soundness*, meaning that it satisfies

$$S(\Delta) = \mathrm{Sol}(D) \cap \{x \in S_1 \times \cdots \times S_n \mid z^* \leq f(x) \leq z^* + \Delta\}$$

Conceivably, this weaker condition for including solutions could allow more flexibility for finding a small sound diagram.

There is a theoretical reason, however, that two-sided soundness is less suitable for practical application. In a one-sided sound diagram, it is easy to check whether a given r–u path of cost w can be completed to represent a δ-optimal solution. Namely, find a shortest u–t path and check whether its length is at most $z^* - w + \delta$. In a two-sided sound diagram, this decision problem is NP-complete. It is therefore difficult to extract δ-optimal solutions from a two-sided sound diagram.

Theorem 1.9. *Checking whether some r–t path in a given decision diagram has cost that lies in an given interval $[z^*, z^* + \delta]$ is NP-complete.*

Proof. The problem belongs to NP because an r–t path with cost in $[z^*, z^* + \delta]$ is a polynomial-size certificate. It is NP-complete because we can reduce the subset sum problem to it. Given a set $S = \{s_1, \ldots, s_n\}$ of integers, the subset sum problem is to determine whether some nonempty subset of these integers sums to zero. We can solve the problem by constructing a decision diagram as follows. Let $U_1 = \{r\}$, $U_j = \{u_{j1}, \ldots, u_{jn}\}$ for $j = 2, \ldots, n$, and $U_{n+1} = \{t\}$. There is one arc with weight s_k from r to each u_{2k}, $k = 1, \ldots, n$. There are two arcs from each u_{jk} to $u_{j+1,k}$ for $j = 2, \ldots, n-1$, one with weight zero, and the other with weight s_j if $j - 1 < k$ and weight s_{j+1} otherwise. There are two arcs from each u_{nk} to t, one with weight zero, and the other with weight s_{n-1} if $k = n$ and weight s_n otherwise. Then there is a one-to-one correspondence between r–t paths and nonempty subsets of S. The subset sum problem has a solution if and only if there is an r–t path with cost in the interval $[0, 0]$. □ □

Corollary 1.10. *Checking whether a given r–u path can be extended to a path with cost in $[z^*, z^*+\delta]$ is NP-complete.*

Proof. Let the given r–u path in diagram D have cost w, and consider the decision diagram D' consisting of all u–t paths of D. The path extension problem is equivalent to checking whether some u–t path in D' has cost in the interval $[z^* - w, z^* - w + \delta]$, which by Theorem 1.9 is an NP-complete problem. □ □

1.7 Sound Diagrams for Bounded Integer Linear Programs

We now specialize problem (P) to an integer linear program:

$$\min\{cx \mid Ax \geq b,\ x \in S_1 \times \ldots \times S_n\} \tag{ILP}$$

in which A is an $m \times n$ matrix and each S_j is a finite set of integers. We wish to build a sound decision diagram that represents Δ-optimal solutions of (ILP); that is, a sound diagram for ILP(Δ). We assume that (ILP) has been solved to optimality and the optimal value z^* is known. This will accelerate the construction of a sound diagram.

We build the diagram by constructing a branching tree, identifying nodes that necessarily have the same set of Δ-suffixes, and removing some nodes that cannot be part of a Δ-optimal solution. To accomplish this, we associate with every node $u \in U_j$ a *state* $(u.\text{lhs}, w(r, u))$. In the simplest case, $u.\text{lhs}$ is an m-tuple in which component i is the sum of left-hand-side terms of inequality constraint i that have been fixed by branching down to layer j. The root node r initially has state $(\mathbf{0}, 0)$, where $\mathbf{0}$ is a tuple of zeros. We can identify nodes u, u' that have the same lhs state, because they necessarily have the same Δ-suffixes. The resulting node v has state $(v.\text{lhs}, w(r, v))$, where $v.\text{lhs} = u.\text{lhs}$ and $w(r, v) = \min\{w(r, u), w(r, u')\}$. Thus the state variable $w(r, v)$ maintains the weight of a minimum-weight r–v path in the current diagram.

We can also observe that inequality constraint i is satisfied at node $u \in U_j$, for any values of $x_j, \ldots, x_n$, when the sum of the fixed terms on the left-hand side is sufficiently large. Specifically, inequality i is necessarily satisfied when the sum of these terms is at least $b_i - M_{ij}$, where

$$M_{ij} = \sum_{k=j}^{n} \min\big\{A_{ik}x_k \mid x_k \in S_k\big\}$$

This allows us to update the lhs state to $\min\{u.\text{lhs},\ b - M_j\}$ and still identify nodes that have the same state. Here, $M_j = (M_{1j}, \ldots, M_{mj})$, and the minimum is taken componentwise.

We can remove a node $u \in U_j$ when the cost of any r–t path through u must be greater than $z^* + \Delta$, based on the linear relaxation of (ILP) at node u. We therefore remove u when $w(r,u) + \text{LP}_j(u.\text{lhs}) > z^* + \Delta$, where

$$\text{LP}_j(u.\text{lhs}) = \min\Big\{\sum_{k=1}^{j-1} c_k x_k \;\Big|\; \sum_{k=j}^{n} A_k x_k \geq b - u.\text{lhs},\;\; x_k \in I_k,\; k = j, \ldots, n\Big\}$$

and I_k is the interval $[\min S_k, \max S_k]$. This can remove some spurious solutions, but not necessarily all, because $w(r,u)$ can underestimate the weight of r–u paths, and $\text{LP}_j(u.\text{lhs})$ can underestimate the weight of u–t paths.

The diagram construction is controlled by Algorithm 1, which maintains unexplored nodes in a priority queue that determines where to branch next. When exploring node $u \in U_j$, the procedure invokes Algorithm 2 to create an outgoing arc for each value in the domain S_j of x_j. Some of these arcs may lead to dead-end nodes based the LP relaxation as described above. If all lead to dead ends, u and predecessors of u with no outgoing arcs are removed by the subroutine at the bottom of Algorithm 1.

Each surviving arc a out of u is processed as follows. Let the node q at the other end of arc a have state $(q.\text{lhs}, w(r,u) + c_j\ell(a))$, where

$$q.\text{lhs} = \min\big\{b - M_j,\ u.\text{lhs} + A_j\ell(a)\big\}$$

If no node currently in U_{j+1} has the same lhs state as q, add node q to U_{j+1}. Otherwise, some node $v \in U_{j+1}$ has $v.\text{lhs} = q.\text{lhs}$, and we let arc a run from u to v, updating $w(r,v)$ if necessary. If v has been explored already, it is revisited in Algorithm 3, because the updated value of $w(r,v)$ may affect which nodes and arcs can be deleted.

A key concept in the procedure is that of a *closed* node. The terminal node t is designated as closed ($t.\texttt{closed} = \textit{true}$) when it is first reached in Algorithm 2. Higher nodes in the diagram are recursively marked as closed when all of their successors are closed. The recursion is implemented by maintaining the number $u.\texttt{openArcs}$ of arcs from node u that do not lead to closed nodes. When Algorithm 1 pops u from the priority queue and processes it, $u.\texttt{openArcs}$ is set to the number $|S_j|$ of domain elements of x_j. Algorithm 1 decrements this number for each dead-end arc, and Algorithm 2 decrements it for each arc leading to a pre-existing node that is closed. Node u is

Algorithm 1 Builds sound diagram for (ILP) using an arbitrary search type

```
procedure PopulateSoundDiagram()
    $r \leftarrow$ new Node($\min\{\mathbf{0}, M_1\}, 0$)
    $U_1 \leftarrow \{r\}$
    PriorityQueue $\leftarrow \{(1, r)\}$                          ▷ Begins search at root node $r$
    RevisitBFSQueue $\leftarrow \{\}$
    while PriorityQueue $\neq \emptyset$ do
        $(j, u) \leftarrow$ PriorityQueue.pop()                    ▷ Queue policy defines search type
        $u$.openArcs $\leftarrow |S_j|$
        Deadend $\leftarrow$ true
        for $\alpha \in S_j$ do
            Success $\leftarrow$ TryBranching($j, u, \alpha$)         ▷ Algorithm 2
            Deadend $\leftarrow$ Deadend $\wedge \neg$Success
        end for
        if Deadend then                                            ▷ No branch succeeded
            RemoveDeadendNode($j, u$)                              ▷ Procedure below
        else if RevisitBFSQueue $\neq \emptyset$ then              ▷ Nodes following $u$ reopened
            RevisitNodes()                                         ▷ Algorithm 3
        else if $u$.openArcs $= 0$ then                            ▷ Nodes following $u$ are all closed
            CloseNode($j, u$)                                      ▷ Algorithm 4
        end if
        $u$.explored $\leftarrow$ true
    end while
end procedure
```

Subroutine: Removes nodes that cannot reach t recursively

```
procedure RemoveDeadendNode($j, u$)
    $U_j \leftarrow U_j \setminus \{u\}$
    for all $a = (v, u) \in A$ do
        $A \leftarrow A \setminus \{a\}$
        $v$.openArcs $\leftarrow v$.openArcs $- 1$
        if $\nexists a' = (v, u') \in A : u' \neq u$ then            ▷ Node above is a deadend
            RemoveDeadendNode($j - 1, v$)
        else if $v$.openArcs $= 0$ then
            CloseNode($j - 1, v$)
        end if
    end for
end procedure
```

closed when u.`openNodes` reaches zero.

One purpose of the node closing mechanism is to implement a possibly more effective test for removing nodes than the LP relaxation. When the terminal node t is reached, a third state variable $w(t,t)$ is set to 0. When a node u is closed, the state variable $w(u,t)$ is updated to indicate the weight of a minimum-weight path to t. Algorithm 4 then removes node u if $w(r,u)+w(u,t) > z^*+\Delta$. It also removes an outgoing arc a to a node v when $w(r,u) + c_j\ell(a) + w(v,t) > z^* + \Delta$. Even this test, however, may not remove all spurious solutions.

Algorithm 2 Tries to branch on value and creates a new node if needed

```
function TryBranching($j, u, \alpha$)
    nodeLhs ← $\min\{u.\text{lhs} + A_j\alpha, M_j\}$
    nodeWeight ← $w(r, u) + c_j\alpha$
    if nodeWeight + $\text{LP}_j(u.\text{lhs}) > z^* + \Delta$ then          ▷ LP = ∞ if infeasible
        return false
    end if
    if $\exists v \in U_{j+1} : v.\text{lhs}$ = nodeLhs then          ▷ Found node with same lhs
        if nodeWeight < $w(r, v)$ then          ▷ Improves minimum cost path to $r$
            $w(r, v)$ ← nodeWeight
            if $v$.explored then
                RevisitBFSQueue.add($j + 1, v$)          ▷ For Algorithm 3
            end if
        end if
        $A \leftarrow A \cup \{(u, v)\}$
        if $v$.closed then          ▷ Fails if not improving for a closed node
            $u$.openArcs ← $u$.openArcs − 1
        end if
    else          ▷ Creates node for new lhs
        $v$ ← new Node(nodeLhs, nodeWeight)
        $U_{j+1} \leftarrow U_{j+1} \cup \{v\}$
        $A \leftarrow A \cup \{(u, v)\}$
        if $j < n$ then          ▷ Adds non-terminal node to queue
            PriorityQueue.add($j + 1, v$)
        else          ▷ First reached terminal node $t$
            $v$.closed ← true
            $w(v, t) \leftarrow 0$
        end if
    end if
    return true
end function
```

1.8 Algorithm for Sound Reduction

Applying the conditions (1.3)–(1.4) for sound reduction presupposes that the suffixes of nodes u and v are known, as well as the weight of a minimum-weight path from v to the terminal node. Sound reduction is therefore attempted only when a node is closed, because it is at this point that the necessary information becomes available.

Algorithm 5 attempts to sound-reduce u into other closed nodes in the same layer, and to sound-reduce other nodes in the layer into u. It is invoked at line 23 in Algorithm 4. To check the conditions for sound-reducing u into v, Algorithm 5 recursively computes the weight of a minimum-weight suffix of v that is not a suffix of u (and similarly with u and v reversed). We refer to this as a *least-cost differing suffix* (LCDS) and denote its weight by $\text{LCDS}_j[v, u]$. The computation of $\text{LCDS}_j[v, u]$ and $\text{LCDS}_j[u, v]$ occurs in lines 4–17 of the algorithm.

The test for sound-reducing u into v occurs in lines 18–22. To break symmetry, we attempt the sound-reduction only when $w(r, v) \leq w(r, u)$. A failure of condition (1.3) for sound reduction occurs when a Δ-suffix of u is not a suffix of v, so that $w(r, u) + \text{LCDS}_j[u, v] \leq z^* + \Delta$. Condition

Algorithm 3 Revisits nodes already explored for updating and re-branching

```
procedure RevisitNodes()
    while RevisitBFSQueue $\neq \emptyset$ do
        $(j, u) \leftarrow$ RevisitBFSQueue.pop()
        if $u$.closed then
            ReopenNode($u$)                                        ▷ Procedure below
        end if
        for $\alpha \in S_j$ do
            if $\nexists a = (u, v) \in A : l(a) = \alpha$ then            ▷ Branches again on absent $\alpha$
                $u$.openArcs $\leftarrow$ $u$.openArcs $+ 1$
                TryBranching($j, u, \alpha$)                               ▷ Algorithm 2
            else
                if $w(r, u) + c_j\alpha < w(r, v)$ then                   ▷ Improves $w(r, v)$
                    $w(r, v) \leftarrow w(r, u) + c_i\alpha$
                    if $v$.explored then
                        RevisitBFSQueue.add($j + 1, v$)
                    end if
                end if
            end if
        end for
        if $u$.openArcs $= 0$ then                        ▷ Nodes following $u$ remained closed
            CloseNode($j, u$)                                           ▷ Algorithm 4
        end if
    end while
end procedure
```

Subroutine: Reopens nodes in bottom-up order recursively

```
procedure ReopenNode($u$)
    $u$.closed $\leftarrow$ false
    for all $a = (v, u) \in A$ do                                 ▷ Opens all nodes above
        if $v$.closed then
            ReopenNode($v$)
            $v$.openArcs $\leftarrow 1$
        else
            $v$.openArcs $\leftarrow$ $v$.openArcs $+ 1$
        end if
    end for
end procedure
```

(1.4) is violated when v has a suffix that is not a suffix of u and incurs a cost no greater than $z^* + \Delta$ when combined with some prefix of u. This occurs when $w(r, u) + \mathrm{LCDS}_j[v, u] \leq z^* + \Delta$. We can therefore sound-reduce u into v when

$$w(r, u) + \min\left\{\mathrm{LCDS}_j[u, v], \mathrm{LCDS}_j[v, u]\right\} > z^* + \Delta$$

The analogous test for sound-reducing v into u occurs in lines 23–26. The removal of a node during sound reduction may disconnect subsequent nodes in the diagram, which are removed by the subroutine at the bottom of Algorithm 5.

Sound reduction is relatively efficient. Algorithm 5 is called $O(nW)$ times, where W is the maximum width of a layer. Each call checks $O(W)$ nodes having $O(S_{\max})$ arcs each, where $S_{\max}$ is

Algorithm 4 Closes nodes and performs bottom-up processing recursively

```
procedure CloseNode($j, u$)
    $u$.closed $\leftarrow$ true
    for all $a = (u, v) \in A$ do                                   ▷ Computes minimum cost path to $t$
        $w(u, t) \leftarrow \min\{w(u, t),\ c_j\ell(a) + w(v, t)\}$
    end for
    if $w(r, u) + w(u, t) > z^* + \Delta$ then                      ▷ Node is not minimal
        RemoveDeadendNode($i, u$)                                   ▷ Subroutine in Algorithm 1
    end if
    for all $a = (u, v) \in A$ do
        if $w(r, u) + c_j\ell(a) + w(u, t) > z^* + \Delta$ then     ▷ Arc is not minimal
            $A \leftarrow A \setminus \{a\}$
        end if
    end for
    ClosingQueue $\leftarrow \{\}$
    for all $a = (v, u) \in A$ do
        if $\neg v$.closed then
            $v$.openArcs $\leftarrow v$.openArcs $- 1$
            if $v$.openArcs $= 0$ then                              ▷ Closes node above
                ClosingQueue $\leftarrow$ ClosingQueue $\cup \{(j, u)\}$
            end if
        end if
    end for
    CompressDiagram($j, u$)                                         ▷ Algorithm ?? or 5
    while ClosingQueue $\neq \emptyset$ do
        $(j, u) \leftarrow$ ClosingQueue.pop()
        CloseNode($j, u$)
    end while
end procedure
```

the size of the largest variable domain. This totals $O(nW^2S_{\max})$ operations before node removals. Each call of the bottom procedure requires time $O(S_{\max})$, for a total time of $O(nWS_{\max})$.

1.9 Postoptimality Analysis

Because a sound decision diagram transparently represents all near-optimal solutions, a wide variety of postoptimality analyses can be conducted with minimal computational effort. We describe a few of these here.

The most basic postoptimality task is to retrieve all feasible solutions whose cost is within a given distance of the optimal cost. That is, we wish to retrieve all δ-optimal solutions from a diagram D that is sound for $\mathrm{P}(\Delta)$, for a desired $\delta \in [0, \Delta]$. This is accomplished by Algorithm 6. The algorithm assumes that the weight $w(r, u)$ of a minimum-weight path from r to each node u has been pre-computed in a single top-down pass. Then, for each desired tolerance δ, the algorithm finds δ-optimal solutions in a bottom-up pass. It accumulates for each node u a set $\mathrm{Suf}^{\delta}(u)$ of suffixes that could be part of a δ-optimal solution, based on the weight of a minimum-weight r–u path. When the algorithm reaches the root r, $\mathrm{Suf}^{\delta}(r)$ is precisely the set $\mathrm{Suf}_{\delta}(r)$ of δ-optimal

Algorithm 5 Sound-reduces node u with another closed node if possible

```
procedure CompressDiagram($j, u$)
    for all $v \in U_j : v \neq u \land v$.closed, and $v$ ordered by nondecreasing $w(r, d)$ do
        $\mathrm{LCDS}_j[u, v], \mathrm{LCDS}_j[v, u] \leftarrow \infty$
        for all $\alpha \in S_j$ do
            if $\exists a_u = (u, u_+) : \ell(a_u) = \alpha$ then
                if $\exists a_v = (v, v_+) : \ell(a_v) = \alpha$ then
                    $\mathrm{LCDS}_j[u, v] \leftarrow \min\{\mathrm{LCDS}_j[u, v],\ w(a_u) + \mathrm{LCDS}_{j+1}[u_+, v_+]\}$
                    $\mathrm{LCDS}_j[v, u] \leftarrow \min\{\mathrm{LCDS}_j[v, u],\ w(a_u) + \mathrm{LCDS}_{j+1}[v_+, u_+]\}$
                else
                    $\mathrm{LCDS}_j[u, v] \leftarrow \min\{\mathrm{LCDS}_j[u, v], w(a_u) + w(u_+, t)\}$
                end if
            else
                if $\exists a_v = (v, v_+) : \ell(a_v) = \alpha$ then
                    $\mathrm{LCDS}_j[v, u] \leftarrow \min\{\mathrm{LCDS}_j[v, u], w(a_v) + w(v_+, t)\}$
                end if
            end if
        end for
        if $w(r, v) \leq w(r, u)$ then
            if $w(r, u) + \min\{\mathrm{LCDS}_j[u, v], \mathrm{LCDS}_j[v, u]\} > z^* + \Delta$ then
                SoundReduce($j, u, v$)                                   ▷ First procedure below
                break
            end if
        else if $w(r, v) + \min\{\mathrm{LCDS}_j[u, v], \mathrm{LCDS}_j[v, u]\} > z^* + \Delta$ then
            SoundReduce($j, v, u$)                                       ▷ First procedure below
            break
        end if
    end for
end procedure
```

Subroutine: Sound-reduces node u into node v at level j

```
procedure SoundReduce($j, u, v$)
    for all $a = (q, u) \in A$ do
        $a \leftarrow (q, v)$                                            ▷ Redirects arcs to $v$
    end for
    $v$.lhs $\leftarrow \emptyset$                                       ▷ Removes $v$'s state
    RemoveIfDisconnected($j, u$)                                         ▷ Next procedure below
end procedure
```

Subroutine: Removes node $u \in U_j$ and subsequent disconnected nodes in D

```
procedure RemoveIfDisconnected($j, u$)
    if $\nexists a = (v, u) \in A$ then
        $U_j \leftarrow U_j \setminus \{u\}$
        for all $a = (u, v) \in A$ do
            $A \leftarrow A \setminus \{a\}$
            RemoveIfDisconnected($j + 1, v$)
        end for
    end if
end procedure
```

solutions, because at this point the exact cost of solutions is known.

The worst-case complexity of the algorithm is proportional to the number of solutions D represents, including spurious solutions. However, many spurious solutions are screened out as the algorithm works its way up, particularly because a solution with cost greater than $z^* + \delta$ (where

Algorithm 6 Retrieves δ-optimal solutions from a sound diagram for $\delta \in [0, \Delta]$

```
function RetrieveSolutions($\delta$)
    $\mathrm{Suf}^\delta(t) = \{\mathrm{null}\}$                ▷ $\mathrm{Suf}^\delta(u)$ = set of possible $\delta$-suffixes of $u$
    $w(\mathrm{null}) = 0$                                  ▷ null is the zero-length suffix.
    for $j = n \to 1$ do                        ▷ Retrieve $\delta$-optimal solutions in bottom-up pass.
        for all $u \in U_j$ do
            $\mathrm{Suf}^\delta(u) = \emptyset$
            for all $a = (u, v) \in A_j$ do
                for all $s \in \mathrm{Suf}^\delta(v)$ do                        ▷ Examine suffixes of $v$.
                    if $w(r, u) + w(a) + w(s) \leq z^* + \delta$ then        ▷ Possible new $\delta$-suffix of $u$?
                        $\mathrm{Suf}^\delta(u) \leftarrow \mathrm{Suf}^\delta(u) \cup \{\ell(a)||s\}$       ▷ Append $\ell(a)$ to suffix $s$.
                        $w(\ell(a)||s) = w(\ell(a)) + w(s)$          ▷ Compute weight of new suffix.
                    end if
                end for
            end for
        end for
    end for
    return $\mathrm{Suf}^\delta(r)$                           ▷ Returns set of $\delta$-optimal solutions.
end function
```

possibly $\delta \ll \Delta$) can be discarded. Retrieval can therefore be quite fast for small δ.

The same algorithm can answer a number of postoptimality questions. For example, one might ask which solutions are δ-optimal when certain variables are fixed to certain values—or, more generally, when the domains S_j of certain variables are replaced by proper subsets S'_j of those domains. This is easily addressed by removing, for each S'_j, all arcs leaving layer j with labels that do not belong to S'_j. Algorithm 6 is then applied to the smaller diagram that results, after recomputing weights $w(r, u)$. Since the use of smaller domains does not add any Δ-optimal solutions, no δ-optimal solutions are missed. Methods for efficient updating of shortest paths are discussed in Miller-Hooks and Yang [2005].

One might also ask which solutions are δ-optimal when the objective function coefficients are altered, say to c'. Since changing the cost coefficients can introduce Δ-optimal solutions, the sound diagram may fail to represent some solutions that are Δ-optimal for the altered costs. However, we can identify all solutions that remain δ-optimal after the cost change for $\delta \leq \Delta - \sum_{j=1}^{n} |c'_j - c_j|$ since all of those were originally Δ-optimal. This is accomplished simply by modifying the arc weights to reflect the new costs and running Algorithm 6, again with recomputed weights $w(r, u)$.

Several additional types of postoptimality analysis can be performed, all with the advantage that spurious solutions have no effect on the computations. These types of analysis can therefore be conducted very rapidly.

For example, we can determine the values that a given variable can take such that the resulting minimum cost is within δ of the optimum. We refer to this as the *δ-optimal domain* of the variable. For each variable x_j, we need only scan the arcs leaving layer j and observe which ones pass test (1.2) when Δ is replaced with δ. This is done in Algorithm 7, which computes δ-optimal domains

Algorithm 7 Computes δ-optimal domains for all variables, where $\delta \in [0, \Delta]$

```
procedure ComputeNearOptimalDomains($\delta$)
    for $j = 1 \to n$ do
        $X_j \leftarrow \emptyset$                                          ▷ $X_j$ is the subset of $S_j$ in $\delta$-optimal solutions.
        for all $a = (u, v) \in A_j : \ell(a) \notin X_j$ do                 ▷ Loops on arcs of missing values
            if $w(r, u) + w(a) + w(v, t) \leq x^* + \delta$ then
                $X_j \leftarrow X_j \cup \{\ell(a)\}$                    ▷ Found a $\delta$-optimal solution where $x_j = \ell(a)$
            end if
        end for
    end for
end procedure
```

for all variables. In particular, the solution value of x_j is invariant across all δ-optimal solutions if its δ-optimal domain is a singleton. The algorithm assumes that shortest path lengths $w(r, u)$ and $w(u, t)$ have been pre-computed for each node u. Its complexity is dominated by the complexity $\mathcal{O}(nW^2)$ of computing the shortest path lengths. The algorithm can also be run after the domains of certain variables are replaced with proper subsets of those domains, to determine the effect on the δ-optimal domains of the other variables.

We can also perform range analysis for individual cost coefficients c_j. As with the previous analysis, the presence of spurious solutions has no effect. For each variable x_j, we can look for the values of c_j that would make each value in the domain of x_j optimal, provided that the other cost coefficients are unchanged. This idea is particularly simple and insightful when the domains are binary, as described in Algorithm 8: if there are solutions in the diagram for which $x_j = 0$ and $x_j = 1$, there is a unique value c'_j for which there are alternate optima with both values. Any $c_j > c'_j$ makes solutions where $x_j = 1$ suboptimal, and conversely any $c_j < c'_j$ makes solutions where $x_j = 0$ suboptimal. If applied to solutions that were originally Δ-optimal, the outcome remains valid as long as c_j does not change more than Δ.

1.10 Computational Experiments

The experiments were designed to assess the compactness of sound decision diagrams, based on 0–1 problem instances in MIPLIB. We constructed three data structures for each of 12 instances and a range of tolerances Δ. The first structure is a branching tree T that represents all Δ-optimal solutions. The second is the sound diagram U that is obtained from Algorithms 1–4, but omitting the sound-reduction step in Algorithm 5. The third is the sound-reduced diagram S that is obtained by applying Algorithms 1–5. Diagram S is therefore a smallest possible sound diagram for the problem instance. By comparing the size U with the size of T, we can see the advantage of representing solutions with a sound decision diagram in which equivalent states are unified. By comparing the size of S with the size of U, we can measure the additional advantage obtained by

Algorithm 8 Computes the cost coefficient c'_j for each variable x_j on 0–1 domains that yields optimal solutions with $x_j = 0$ and $x_j = 1$ among the solutions of the decision diagram, if the other cost coefficients remain the same

```
function ComputeIndifferentCostCoefficients()
    for $j = 1 \to n$ do
        for $\alpha \in \{0, 1\}$ do
            $z_\alpha \leftarrow \infty$                          ▷ $z_\alpha$ is min $\sum_{j' \neq j} c_{j'} x_{j'}$ when $x_j = \alpha$
        end for
        for all $a = (u, v) \in A_j$ do
            if $w(r, u) + w(v, t) < z_{\ell(a)}$ then
                $z_{\ell(a)} \leftarrow w(r, u) + w(v, t)$        ▷ Found a lower value of $\sum_{j' \neq j} c_{j'} x_{j'}$
            end if
        end for
        if $z_0 = \infty$ then                                  ▷ There is no solution with $x_j = 0$
            $c'_j = \infty$
        else if $z_1 = \infty$ then                             ▷ There is no solution with $x_j = 1$
            $c'_j = -\infty$
        else                                                    ▷ There are solutions for both assignments
            $c'_j = z_0 - z_1$                                  ▷ Coefficient for which $z_0 = z_1 + c'_j$
        end if
    end for
    return $c'$
end function
```

sound reduction.

We carried out the experiments for tolerances Δ that range over a wide interval from zero to $\Delta_{\max}$ in increments of $0.1\Delta_{\max}$. For the smaller instances, we set $\Delta_{\max}$ large enough to encompass all feasible solutions; that is, large enough so that all feasible solutions are $\Delta_{\max}$-optimal. These instances are bm23, enigma, p0033, p0040, stein9, stein15, and stein27. Thus for these instances, $\Delta_{\max}$ is the difference in value between the best and worst solutions. For the remaining instances, we set $\Delta_{\max}$ equal to the median absolute value of nonzero objective coefficients. This allows us to test variations of up to 100% in objective coefficients of half of these variables. If the runtime was less that 1000 seconds, we kept doubling $\Delta_{\max}$ until the runtime exceeded 1000 seconds, but stopped short of a doubling that resulted in a runtime of more than 24 hours.

In all experiments, the branching priority is DFS, variables are ordered by increasing index, and 0-arcs are explored before 1-arcs. The code is written in C++ (gcc version 4.8.24), uses the COIN-OR CLP solver[1] (version 1.16.10), and ran in Ubuntu 14.04.2 LTS on a machine with Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz processors and 128 GB of RAM.

Table 1.2 displays the statistics for the maximum tolerance $\Delta_{\max}$, including the number of optimal and $\Delta_{\max}$-optimal solutions, and the size and construction time for T, U, and S. The sound-reduced diagram S is dramatically smaller than the branching tree T for all the instances except enigma. It is also significantly smaller than the unified diagram U except in the cases of

[1] projects.coin-or.org/Clp

Table 1.2: Solution counts, with diagram sizes and construction times, for MIPLIB instances using a maximum tolerance $\Delta_{\max}$.

		Solutions		Size (nodes)			Runtime (s)	
Instance	$\Delta_{\max}$	Opt.	$\Delta_{\max}$-opt.	T	U	S	U	S
air01	3194	2	16,899	5,058,113	61,652	61,652	340	11,000
bm23*	59	1	2,168	23,620	20,356	5,460	31	40
enigma*	1	2	4	278	243	243	41	41
lseu	236.16	2	67,250	2,057,264	294,108	53,465	2,900	8,600
mod008	21	6	4,954	891,543	188,359	38,292	15,000	16,000
p0033*	2112	9	10,746	55,251	847	449	5.8	33
p0040*	7102	1	519,216	2,736,899	2,950	831	2.6	620
p0201	375	4	34,504	2,326,052	107,312	6,627	1,900	7,700
sentoy	280.8	1	85,401	1,868,562	1,754,681	101,618	3,800	12,000
stein9*	4	54	172	460	137	80	0.02	0.05
stein15*	6	315	2,809	8,721	2,158	816	0.54	1.6
stein27*	9	2,106	367,525	1,450,702	338,916	25,444	159	1,400

* The value of $\Delta_{\max}$ makes the set large enough to include all feasible solutions.

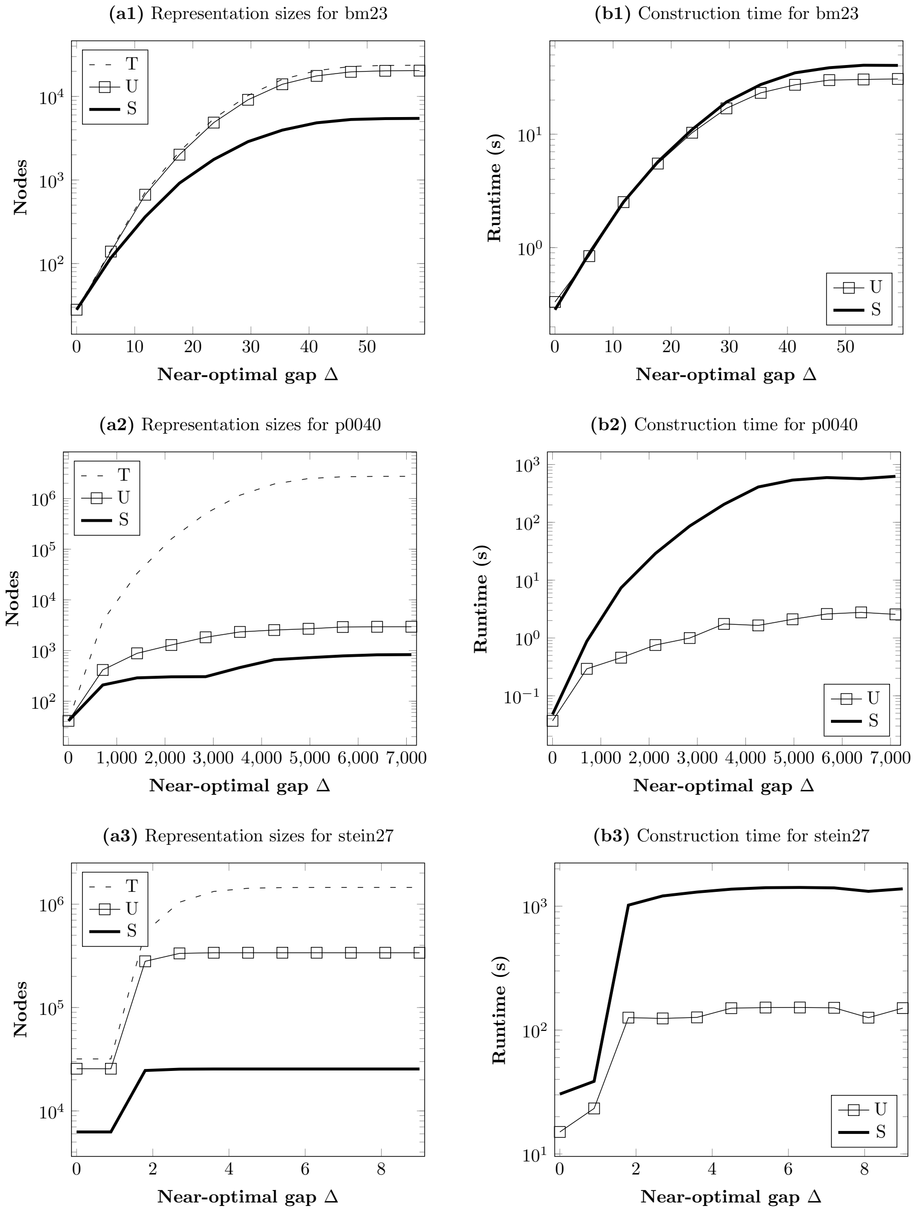
air01 and enigma, and smaller by at least an order of magnitude in three instances. On the other hand, sound reduction added significantly more computation time to the diagram construction in seven of the instances.

In practice, the desired tolerance Δ is typically much less than $\Delta_{\max}$. We therefore display in Figs. 1.6–1.7 how the diagram sizes and computation times depend on Δ for six of the instances. Note that the diagram sizes and runtimes are plotted on a logarithmic scale. As predicted by Corollary 1.4, the diagram size is monotone nondecreasing in Δ.

The sound-reduced diagram S is substantially more compact than the branching tree T in every instance. It is also smaller than U in all instances but one, although one must normally pay a higher computational price for this reduction. Of course, a sound-reduced diagram need only be generated once in order to carry out a large number of postoptimality queries. The sound-reduced diagrams for typical values of Δ are well within a practical size range for rapid postoptimality processing, normally a few hundred or a few thousand nodes. The computation times for constructing diagrams of this size are likewise modest, ranging from a few seconds to a few minutes.

1.11 Conclusion

We explored sound decision diagrams as a data structure for concisely and transparently representing near-optimal solutions of integer programming problems. We showed that repeated application of a simple sound-reduction step yields a smallest possible sound diagram for any given discrete optimization problem. Based on this result, we stated an algorithm for constructing sound-reduced diagrams for integer programming problems. We showed how the resulting diagrams permit several types of postoptimality analysis, and that the presence of spurious solutions in the diagrams has no effect on most types of analysis. Computational testing indicates that sound-reduced diagrams


(a1) Representation sizes for bm23

(b1) Construction time for bm23

(a2) Representation sizes for p0040

(b2) Construction time for p0040

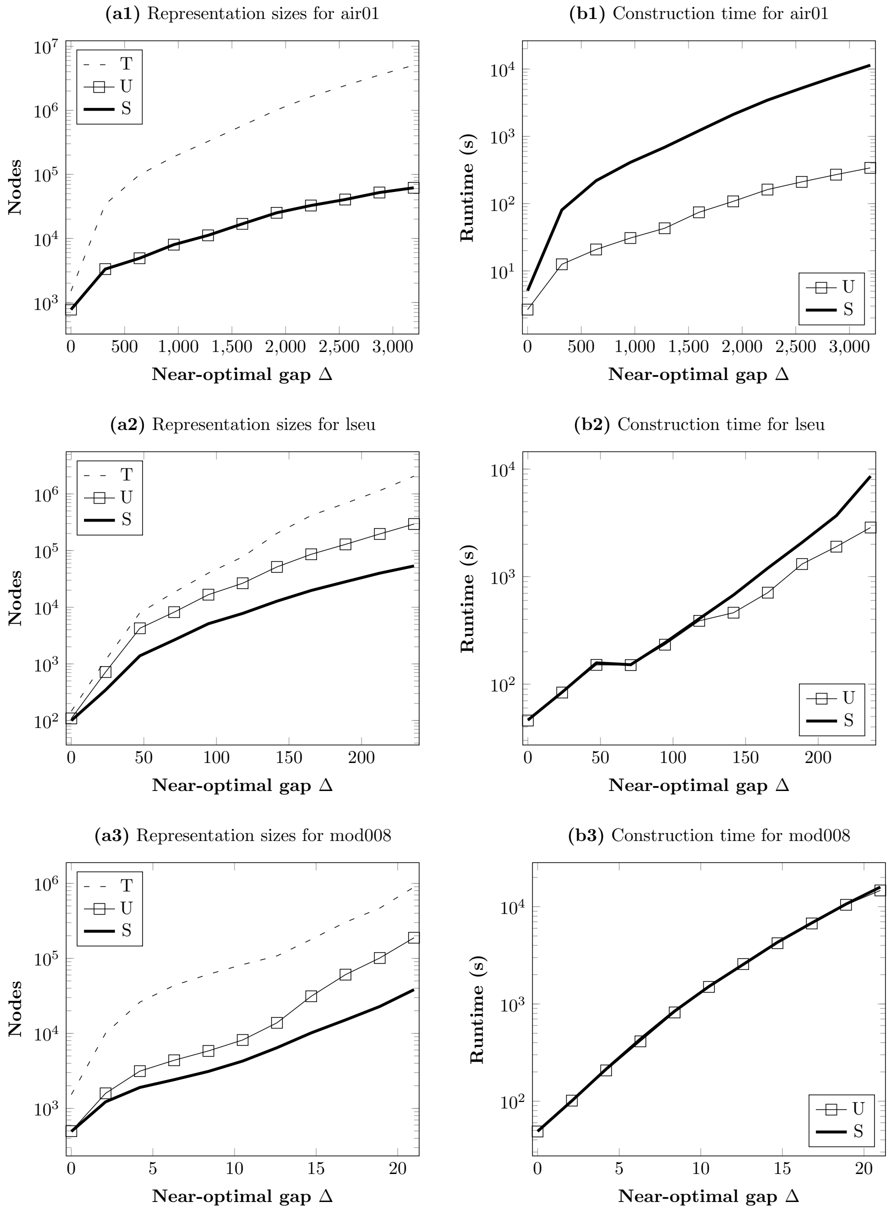
(a3) Representation sizes for stein27

(b3) Construction time for stein27

Figure 1.6: Diagram size and computation time vs. Δ for three smaller instances.

generally offer dramatic reductions in the space required to represent near-optimal solutions, relative to that required by a branching tree. For the MIPLIB instances tested, the resulting diagrams are well within a size range that permits rapid postoptimality processing.

This study is inspired by the idea that solution of an optimization problem should be viewed more broadly than merely generating one or more optimal solutions. Rather, it should be seen as transforming an opaque data structure that defines the problem but does not reveal its solutions, to a transparent data structure that provides ready access to optimal and suboptimal solutions of



(a1) Representation sizes for air01

(b1) Construction time for air01

(a2) Representation sizes for lseu

(b2) Construction time for lseu

(a3) Representation sizes for mod008

(b3) Construction time for mod008

Figure 1.7: Diagram size and computation time vs. Δ for three larger instances.

interest. We attempted to lay a foundation for this type of solution for integer programming, but an obvious research direction is to extend the method to mixed integer programming. Decision diagrams can continue to play a role, because paths in a diagram can represent values for the integer variables in the problem.

Chapter 2

Reverse Polar Normalization of Lift-and-Project Cuts

This chapter is based on the manuscript "Reformulating the Disjunctive Cut Generating Linear Program" [Serra, 2018], which was presented at the MIP workshop in 2016 and at the Interactive Presentation Award at the INFORMS 2016 Annual Meeting.

2.1 Introduction

Many optimization problems can be formulated as a Mixed Integer Linear Program (MILP) of the form $\min\{c^T x : Ax \geq b, x \in \{0,1\}^p \times \mathbb{R}_+^{n-p}\}$. Some are found more frequently and have been studied in more detail, such as the classic traveling salesman problem, for which families of valid inequalities are known [Applegate et al., 2006]. One can tackle an MILP problem by solving its Linear Program (LP) relaxation $\min\{c^T x : Ax \geq b, x \in \mathbb{R}_+^n\}$ and then iteratively branch to restrict the domains of integer variables or add inequalities of the form $\alpha^T x \geq \beta$ to separate solutions in which those variables are fractional. These inequalities are denoted as cuts with respect to the fractional solutions that they separate, and in many cases the cuts from different methods are equivalent.

In general, there is a greater and justified interest for facet-defining cuts, which are those essential to characterize a full-dimensional convex hull of feasible solutions. There is also a secondary interest in the broader family of cuts defining supporting hyperplanes, which cannot be strengthened by merely increasing the right-hand side β. When considering a general MILP instead of a special case, however, it is challenging to guarantee that a cut defines even a supporting hyperplane with respect to the convex hull of the feasible solutions. Hence, we aim for a compromise: which cuts would define facets or supporting hyperplanes of the immediate closure? In other words, among the many cuts that can be obtained in the first round of a method, can we select the essential ones or else those that are not comparatively weaker?

We investigate this question from the perspective of lift-and-project [Balas et al., 1993], which is a method to generate cuts by defining tighter relaxation of the MILP using a Disjunctive Program (DP) [Balas, 1998]. Such DP usually consists of a union of disjoint polyhedra covering the feasible set of the MILP. In a typical example, if a solution $\bar{x}$ of the LP is such that $0 < \bar{x}_i < 1$ for some $1 \leq i \leq p$, we can intersect the split disjunction $\{x : x_i \leq 0\} \cup \{x : x_i \geq 1\}$ with the LP feasible set $\{x : Ax \geq b, x \geq 0\} := \{x : \tilde{A}x \geq \tilde{b}\}$ to define a system on the inequalities that are valid for each term and thus for the disjunctive hull $\mathrm{P}^{\mathrm{K}} = conv\Big(\{x \in \mathbb{R}^n : \tilde{A}x \geq \tilde{b}, -x_i \geq 0\} \cup \{x \in \mathbb{R}^n : \tilde{A}x \geq \tilde{b}, x_i \geq 1\}\Big)$. In fact, we can define a restricted system without loss of non-dominated inequalities as follows:

$$\begin{array}{llll} \alpha & -u^T\tilde{A} & +u_0 e_k & = 0 \\ \alpha & -v^T\tilde{A} & -v_0 e_k & = 0 \\ \beta & -u^T\tilde{b} & & = 0 \\ \beta & -v^T\tilde{b} & -v_0 & = 0 \\ & u, v, & u_0, v_0 & \geq 0 \end{array} \qquad (\overline{\mathrm{C}})_k$$

Among these inequalities, we obtain one that separates $\bar{x}$ by solving a Cut Generating Linear Program (CGLP) [Balas et al., 1993, 1996] such as

$$\begin{array}{ll} \min & \alpha^T\bar{x} - \beta \\ \text{s.t.} & (\overline{\mathrm{C}})_{\mathrm{k}} \\ & u^T e + v^T e + u_0 + v_0 = 1 \end{array} \qquad (\mathrm{CGLP})_k$$

These formulations have invariably aimed at maximizing the cut violation for $\bar{x}$, i.e., making $\alpha^T\bar{x} - \beta$ as negative as possible. However, cuts from $(\mathrm{CGLP})_k$ optima may neither define a facet nor a supporting hyperplane of the immediate closure [Fischetti et al., 2011]. This paradox is due to how cuts are ranked by the CGLP, which relates to how these formulations restrict the algebraic representation of the valid inequalities. Since $(\lambda\alpha)^T x \geq (\lambda\beta)$ is the same cut for any $\lambda > 0$, one has to further limit the feasible set defined by the cone $(\overline{\mathrm{C}})_{\mathrm{k}}$ to guarantee the existence of an optimal solution. This is usually done by adding a so-called normalization constraint, such as $u^T e + v^T e + u_0 + v_0 = 1$. These constraints are as important to the CGLP outcome as the objective function: each normalization defines a different infeasibility certificate for $\bar{x}$ from solving the CGLP dual [Ceria and Soares, 1997], which in turn validates cuts from CGLP optima. Aiming for cuts with better guarantees, our goal is to understand which normalization to use and, more broadly, how to define the CGLP.

The name for these constraints comes from early approaches fixing the norm of cut coefficients through linear constraints, including $\|\alpha\|_\ell = 1$ for $\ell \in \{1, \infty\}$ and $\beta = \kappa$ for $\kappa \in \{-1, 0, +1\}$. Fixing an ℓ-norm of α implies that cuts from CGLP maximize the distance to $\bar{x}$ in that norm. While $\ell \in \{1, \infty\}$ can be defined with multiple constraints, Cadoux [2010] explored a nonlinear formulation to maximize Euclidean distance. However, cuts maximizing distance are not necessarily facet-defining. More generally, since facet-defining cuts correspond to extreme rays of $(\overline{\mathrm{C}})_{\mathrm{k}}$, it is preferable to apply a normalization consisting of a single linear constraint that intersects all rays to ensure that facet-defining cuts correspond to extreme points of the CGLP. In the case of fixing β, we are partitioning the cuts into three CGLPs and some of those might remain unbounded. Hence, more recent approaches have focused on constraining the Farkas multipliers instead. Fischetti et al. [2011] shows that the so-called Standard Normalization Constraint (SNC) $u^T e + v^T e + u_0 + v_0 = 1$ tends to generate sparser and lower-rank cuts, but also that the solutions depend on the scaling of the constraints and they might not define supporting hyperplanes. Variants and a generalization of SNC are discussed by Fischetti et al. [2011] and Balas and Bonami [2009], respectively. Notably, is has been shown that the so-called trivial normalization $u_0 + v_0 = 1$ yields the Gomory fractional cut when $\bar{x}$ is a basic solution of the LP (for example, in Fischetti et al. [2011]).

Finally, we note that explicitly solving a CGLP is considered prohibitive in practice because the CGLP is at least twice as large as the LP. For a system such as $(\overline{\mathrm{C}})_k$, there are two rows for each column of the LP and two columns for each row. However, it is possible to find cuts from CGLP optima through formulations with same size as the LP. In the case of split disjunctions, Balas and Perregaard [2003] have shown that there is a correspondence between cuts from CGLP optima and Gomory fractional cuts from basic solutions of the LP, which may or may not be feasible. Hence, one may pivot among LP basic solutions to find a cut deemed as optimal by the CGLP formulation [Balas and Perregaard, 2003, Balas and Bonami, 2009]. In the case of the trivial normalization on 2-term disjunctions, Bonami [2012] has shown how to project out the Farkas multipliers and obtain a dual problem known as the Membership Linear Program (MLP), which only differs from the LP by the objective and right-hand side.

2.1.1 Contribution

We propose a CGLP reformulation that we name the Reverse Polar CGLP (RP-CGLP), which switches the roles played by the objective function and the normalization constraint. Cuts are normalized by fixing their violation with respect to $\bar{x}$ and the objective function evaluates the cut at a point $p \in \mathrm{P}^{\mathrm{K}}$. To the best of our knowledge, this is the first CGLP formulation that uses a different objective, which can be leveraged to generate diverse cuts.

More interestingly, cuts derived from optimal solutions of RP-CGLP define supporting hyperplanes of the immediate closure. When the disjunctive hull P^{K} is full-dimensional, there is always a facet-defining cut from some RP-CGLP optimum. In fact, cutting planes from RP-CGLP optima are those exposed when a ray from $\bar{x}$ toward p first intersects P^{K}. If the point at which that ray first intersects P^{K} is at the interior of a facet, then that facet is the unique cutting plane from RP-CGLP optima. More generally, each cutting plane from RP-CGLP optima is a combination of facets separating and active at $\bar{x}$.

We note that related work by Balas and Perregaard [2002], Cadoux and Lemaréchal [2013], and Conforti and Wolsey [2016] can be framed as proposing CGLP variants yielding the same cuts. Compared to those, the main advantage of ours is that the feasible set of RP-CGLP does not depend on p, which facilitates generating multiple cuts by just reoptimizing the CGLP with a new objective function. This equivalence is shown by unveiling the true objective function of these CGLPs after normalization. To the best of our knowledge, a precise and meaningful objective function has only been previously reported for CGLPs fixing a norm of α.

Finally, we show that the solution of RP-CGLP can be mimicked over the tableau of the LP relaxation, hence requiring little adaptation to be incorporated in solvers generating lift-and-project cuts that way. We report computational results on the implementation.

2.1.2 Organization

First, we present the RP-CGLP and its properties in Section 2.2, and prove its equivalence to other recently proposed CGLP formulations in Section 2.3. In the sequence, we present results to solve the RP-CGLP using the LP tableau in Section 2.4, we show experiments comparing RP-CGLP to a conventional CGLP in Section 2.5. We draw some conclusions in Section 2.6.

2.2 The Reverse Polar Reformulation

We propose the Reverse Polar Cut Generating Linear Program (RP-CGLP) to generate a cut $\alpha^T x \geq \beta$ separating $\bar{x}$ with the orientation of some point $p \in \mathrm{P}^{\mathrm{K}}$:

$$\begin{aligned} \min \quad & \alpha^T p - \beta \\ \text{s.t.} \quad & (\overline{\mathrm{C}})_{\mathrm{k}} \\ & \beta - \alpha^T \bar{x} = 1 \end{aligned} \qquad \text{(RP-CGLP)}_k^p$$

Similarly to other CGLP formulations, (RP-CGLP)_k^p contains a single normalization constraint, $\beta - \alpha^T \bar{x} = 1$, which fixes the violation conventionally maximized by a CGLP. Moreover, the interplay between objective and normalization changes. While most normalizations bound the feasible set to guarantee that there is an optimum, we discuss in the next paragraph that the feasible set remains unbounded, whereas the normalization prevents the root $(\alpha, \beta, u, v, u_0, v_0) = \mathbf{0}$ of $(\overline{\mathrm{C}})_{\mathrm{k}}$ from being optimal. That is due to $\alpha^T p \geq \beta$ for any valid inequality, which implies that the objective is always nonnegative while $\alpha^T \mathbf{0} - 0 = 0$. Finally, the separability of $\bar{x}$ is guaranteed by CGLP feasibility instead of optimality.

By reformulating the MILP on coordinates centered at $\bar{x}$, say $x' = x - \bar{x}$, the corresponding RP-CGLP defines a cut of the form $\alpha'^T x' \geq \beta'$, where $\alpha' = \alpha$ and $\beta' = 1$. Hence, valid cuts can be characterized by their left-hand sides, which define a subset of the reverse polar set $\left(\mathrm{P}^{\mathrm{K}} - \bar{x}\right)^- := \left\{y : y^T(x - \bar{x}) \geq 1 \ \forall x \in \mathrm{P}^{\mathrm{K}}\right\}$. For any $p \in \mathrm{P}^{\mathrm{K}}$, (RP-CGLP)_k^p yields a cut $\alpha^T x \geq \beta$ for some

$y \in \left(\mathrm{P}^{\mathrm{K}} - \bar{x}\right)^{-}$, where $\alpha = y$ and $\beta = 1 + y^T\bar{x}$. More broadly, cone $(\overline{\mathrm{C}})_{\mathrm{k}}$ along with normalization $\beta - \alpha^T\bar{x} = 1$ defines an extended formulation of the reverse polar set.

When convenient to elucidate proofs, cuts from (RP-CGLP)$_k^p$ will be denoted in the form $y^T(x - \bar{x}) \geq 1$. Whenever we refer to a cut from (RP-CGLP)$_k^p$ or from any other formulation, we assume that these cuts come from optimal solutions of that CGLP.

Lemma 2.1. *Cuts from (RP-CGLP)$_k^p$ define supporting hyperplanes of* P^{K}.

Proof. Let us suppose, for contradiction, that there is a cut from (RP-CGLP)$_k^p$ where that does not hold. Since the distance from $\bar{x}$ to $\alpha^T\bar{x} = \beta$ is given by $\operatorname{dist}(\alpha^T x = \beta, \bar{x}) = \dfrac{|\alpha^T\bar{x} - \beta|}{\|\alpha\|}$ with respect to any norm, the norm of parallel cutting planes get smaller as they move away from $\bar{x}$. Hence, if we put that cut in the form $\bar{y}^T(x - \bar{x}) \geq 1$, then $\exists \varepsilon \in (0,1)$ for which $\varepsilon\bar{y}^T(x - \bar{x}) \geq 1$ is valid for P^{K}. However, the former cut would not be optimal since the objective function value of the latter is smaller: $\bar{y}^T(p - \bar{x}) - 1 > \varepsilon\bar{y}^T(p - \bar{x}) - 1$. □

From this point on, let us assume that P^{K} is full-dimensional and then characterize which of its facets intersect at cutting planes from (RP-CGLP)$_k^p$ optima. Let $\mathcal{F} = \left\{(\gamma^i)^T(x - \bar{x}) \geq \delta_i\right\}_{i \in F}$ be the set of facet-defining inequalities of P^{K}, with F finite for A, b rational. Without loss of generality, we partition $F = F^+ \cup F^0 \cup F^-$, where $\delta_i = 1$ if $i \in F^+$, $\delta_i = 0$ if $i \in F^0$, and $\delta_i = -1$ if $i \in F^-$. Hence, a face-defining cut $\bar{y}^T(x - \bar{x}) \geq 1$ such as an (RP-CGLP)$_k^p$ optimum can be described by some nonnegative combination of multipliers $\{\bar{\lambda}_i\}_{i \in F}$ in which $\bar{y} = \sum\limits_{i \in F} \bar{\lambda}_i \gamma^i$, $\bar{\lambda} \geq 0$, and $\sum\limits_{i \in F^+} \bar{\lambda}_i - \sum\limits_{i \in F^-} \bar{\lambda}_i = 1$ since $\sum\limits_{i \in F} \bar{\lambda}_i \delta_i = 1$.

Now we can characterize the cuts from (RP-CGLP)$_k^p$ through a result that resembles complementary slackness for the facets not separating $\bar{x}$:

Theorem 2.2. *For a cut $\bar{y}^T(x - \bar{x}) \geq 1$ derived from (RP-CGLP)$_k^p$ for $p \in \mathrm{P}^{\mathrm{K}}$, any combination $\bar{\lambda}$ is such that $\bar{\lambda}_i = 0$ or $(\gamma^i)^T(p - \bar{x}) - \delta_i = 0$ $\forall i \in F^0 \cup F^-$.*

Proof. Suppose not for a cut of the form $\bar{y}^T(x - \bar{x}) \geq 1$ with a combination $\bar{\lambda}$. Let $G^0 := \{i \in F^0 : \bar{\lambda}_i > 0 \text{ and } (\gamma^i)^T(p - \bar{x}) > 0\}$ and $G^- := \{i \in F^- : \bar{\lambda}_i > 0 \text{ and } (\gamma^i)^T(p - \bar{x}) + 1 > 0\}$, where $G^0 \cup G^- \neq \emptyset$. If $G^0 \neq \emptyset$, then starting with $\lambda' \leftarrow \bar{\lambda}$ and setting $\lambda'_i \leftarrow 0$ $\forall i \in G^0$ would yield a valid cut with objective value smaller by $\sum\limits_{i \in G^0} \bar{\lambda}_i[(\gamma^i)^T(p - \bar{x})] > 0$. If $G^- \neq \emptyset$, we similarly could start with $\lambda' \leftarrow \bar{\lambda}$, set $\lambda'_i \leftarrow 0$ $\forall i \in G^-$ and accordingly set $\lambda'_j \leftarrow \bar{\lambda}_j / \left(1 + \sum\limits_{i \in G^-} \bar{\lambda}_i\right)$ $\forall j \in F^+$ to keep $\sum\limits_{i \in F^+} \lambda'_i - \sum\limits_{i \in F^-} \lambda'_i = 1$. That reduces the CGLP objective by $\sum\limits_{i \in G^-} \bar{\lambda}_i[(\gamma^i)^T(p - \bar{x}) + 1] > 0$ and $\sum\limits_{j \in F^+} (\bar{\lambda}_j - \bar{\lambda}'_j)[(\gamma^j)^T(p - \bar{x}) - 1] \geq 0$, respectively. Hence, $\bar{y}^T(x - \bar{x}) \geq 1$ would not come from an (RP-CGLP)$_k^p$ optimum, a contradiction. □

Corollary 2.3. *For a cut $\bar{y}^T(x-\bar{x}) \geq 1$ with combination $\bar{\lambda}$ derived from (RP-CGLP)$_k^p$ for $p \in \mathrm{P}^{\mathrm{K}}$, we have $\bar{y}^T(p-\bar{x}) = (\gamma^i)^T(p-\bar{x})\ \forall i \in F^+ : \bar{\lambda}_i > 0$.*

Proof. From Theorem 2.2, we have $\bar{y}^T(p-\bar{x}) = \sum\limits_{i \in F^+} \bar{\lambda}_i[(\gamma^i)^T(p-\bar{x})]$. Furthermore, either $\bar{y}^T(p-\bar{x}) - 1 = 0$ or $\sum\limits_{i \in F^+} \bar{\lambda}_i = 1$, since otherwise $\sum\limits_{i \in F^-} \bar{\lambda}_i > 0$ and we could find a cut with better objective with multipliers λ' by starting with $\lambda' \leftarrow \bar{\lambda}$, setting $\lambda'_i \leftarrow 0\ \forall i \in F^-$, and then scaling down with $\lambda'_j \leftarrow \bar{\lambda}_j / \left(1 + \sum\limits_{i \in F^-} \bar{\lambda}_i\right)\ \forall j \in F^+$. Thus, if $\bar{y}^T(p-\bar{x}) > (\gamma^i)^T(p-\bar{x})$ for some $i \in F^+ : \bar{\lambda}_i > 0$, then $\exists j \in F^+ : \bar{\lambda}_j > 0$ such that $\bar{y}^T(p-\bar{x}) < (\gamma^j)^T(p-\bar{x})$ and vice-versa. If that was possible, however, then there would be a cut with strictly better objective by increasing $\bar{\lambda}_i$ while decreasing $\bar{\lambda}_j$ accordingly. □

Corollary 2.4. *For $p \in int(\mathrm{P}^{\mathrm{K}})$, a cut from (RP-CGLP)$_k^p$ is a combination $\bar{\lambda}$ of facets separating $\bar{x}$ that each correspond to some (RP-CGLP)$_k^p$ optimum.*

Proof. If p is not on any facet, Theorem 2.2 implies that $\bar{\lambda}_i = 0\ \forall i \in F^0 \cup F^-$, and by Corollary 2.3 the cut from each facet i with $\bar{\lambda}_i > 0$ defines an (RP-CGLP)$_k^p$ optimum. □

Thus, cuts from (RP-CGLP)$_k^p$ are a combination of (i) inequalities of $\mathcal{F}$ that are active at p, and (ii) inequalities of $\mathcal{F}$ that separate $\bar{x}$. The only set with non-zero objective value is the latter, which comprises inequalities indexed by F^+ with same evaluation for p if normalized by the same right-hand side, hence each associated with some (RP-CGLP)$_k^p$ optimum. We are now left with one question – what makes the cuts defining each of such facets optimal?

Lemma 2.5. *A cut $\alpha^T x \geq \beta$ from (RP-CGLP)$_k^p$ with objective value ζ is active at the point $p' := \bar{x} + \frac{1}{\zeta+1}(p-\bar{x})$, which lies on the ray from $\bar{x}$ to p.*

Proof. It suffices to check that $\alpha^T p' - \beta = 0$: $\alpha^T\left(\bar{x} + \frac{1}{\zeta+1}(p-\bar{x})\right) - \beta = (\alpha^T\bar{x} - \beta) + \left(\frac{1}{\zeta+1}(\alpha^T p - \alpha^T\bar{x})\right) = -1 + \left(\frac{1}{\zeta+1}(\alpha^T p - (\beta - 1))\right) = -1 + \left(\frac{1}{\zeta+1}(\zeta+1)\right) = 0$. □

Theorem 2.6. *A cut from (RP-CGLP)$_k^p$ is active at the first intersection of P^{K} with the ray from $\bar{x}$ to p, which corresponds to point p' from Lemma 2.5.*

Proof. A cut $\alpha^T x \geq \beta$ from (RP-CGLP)$_k^p$ is such that $\alpha^T\bar{x} - \beta = -1$ and $\alpha^T p - \beta \geq 0$, hence defining a monotonically increasing function for the slack along the ray. Since that slack is negative for any point before p', those are all separated by the cut and p' is the first intersection of the ray with P^{K}. □

In other words, cutting planes from (RP-CGLP)$_k^p$ optima are combinations of facets of P^{K} that are first intersected by a ray from $\bar{x}$ toward p. Note that the ray from $\bar{x}$ to p may intersect other facets separating $\bar{x}$ prior to p'. In fact, a cutting plane from (RP-CGLP)$_k^p$ only combines facets indexed by F_+ that are last intersected by the ray from $\bar{x}$ to p.

We can observe that in a different way. If we replace β according to the normalization, the objective can be restated as $\min \alpha^T(p - \bar{x}) - 1$. Consequently, all points defining the same ray with $\bar{x}$ as p yield the same cuts from (RP-CGLP)$_k^p$. If we choose a point p' along that ray for which the objective value is 0, it becomes clear that facets separating $\bar{x}$ should be active at p'. If p' is at the interior of a facet of P^K, then that facet is the unique cutting plane from (RP-CGLP)$_k^p$ optima, as implied by Corollary 2.3.

2.3 Equivalent and Related Work

For any CGLP formulation, the imposed normalization and its interplay with the objective function may result in a different ranking of the cuts. In order to compare (RP-CGLP)$_k^p$ with other recent formulations, we need to understand how cuts are truly evaluated.

Lemma 2.7. *Valid inequalities of* P^K *that separate* $\bar{x}$ *are compared by (RP-CGLP)$_k^p$ using*

$$\min \frac{\alpha^T p - \beta}{\beta - \alpha^T \bar{x}} \tag{2.1}$$

Proof. Let us denote cuts in the form $\mu^T x \geq \nu$, where $\|\mu\| = 1$. Hence, for a cut of the form $\alpha^T x \geq \beta$, we consider a correspondence of the form $(\alpha, \beta) = \theta(\mu, \nu)$ for some $\theta > 0$.

Normalization $\beta - \alpha^T \bar{x} = 1$ implies that $\theta = \frac{1}{\nu - \mu^T \bar{x}}$. The objective function is restated as $\min \alpha^T p - \beta = \min \theta(\mu^T p - \nu) = \min \frac{\mu^T p - \nu}{\nu - \mu^T \bar{x}}$. Therefore, the cuts separating $\bar{x}$ obtained with (RP-CGLP)$_k^p$ are those minimizing the ratio between the slack for p and the violation for $\bar{x}$. Note that the ratio does not depend on the algebraic representation of the cut. □

2.3.1 Equivalent Formulations

A similar reformulation is proposed by Balas and Perregaard [2002], as follows:

$$\begin{array}{ll} \min & \alpha^T \bar{x} - \beta \\ \text{s.t.} & (\overline{\mathrm{C}})_\mathrm{k} \\ & \alpha^T(p - \bar{x}) = 1 \end{array} \tag{BP-CGLP$_k^p$}$$

Balas and Perregaard [2002] have proven that (BP-CGLP)$_k^p$ has an optimum if, and only if, the line defined by $\bar{x}$ and p ever intersects P^K. If so, the resulting cut defines a supporting hyperplane, which contains the point of P^K that is the closest to $\bar{x}$ on the line between $\bar{x}$ and p. This normalization is used for multi-row cuts by Louveaux et al. [2015]. Formulation (BP-CGLP)$_k^p$ resembles (RP-CGLP)$_k^p$ when the objective of the latter is restated as $\min \alpha^T(p - \bar{x}) - 1$ by substituting β according to the normalization, except that switching the expressions for the objective function and the normalization constraint.

However, a key difference between $(\text{BP-CGLP})_k^p$ and $(\text{RP-CGLP})_k^p$ is that the feasible set of the latter does not depend on p. That allows to generate different cuts by just changing the objective function and reoptimizing, hence entailing more variability if we choose the next point in P^{K} to entirely change the set of CGLP optima. Thus, it generalizes the approach by Balas [1997] of using alternate CGLP optima to generate multiple cuts.

Corollary 2.8. *$(\text{BP-CGLP})_k^p$ and $(\text{RP-CGLP})_k^p$ derive the same cuts for $p \in \mathrm{P}^{\mathrm{K}}$.*

Proof. The normalization of $(\text{BP-CGLP})_k^p$ implies $\theta = \frac{1}{(\mu)^T(p-\bar{x})}$, and the objective becomes $\min \alpha\bar{x} - \beta = \max \beta - \alpha\bar{x} = \max \theta[\nu - \mu^T\bar{x}] = \max \frac{\nu-\mu^T\bar{x}}{\mu^T(p-\bar{x})} = \max \frac{\nu-\mu^T\bar{x}}{(\mu^T p-\nu)+(\nu-\mu^T\bar{x})}$. Since $(\text{BP-CGLP})_k^p$ optima yield cuts separating $\bar{x}$, we can divide numerator and denominator by the violation. The objective becomes $\max \frac{1}{\frac{\mu^T p-\nu}{\nu-\mu^T\bar{x}}+1}$, which is equivalent to $\min \frac{\mu^T p-\nu}{\nu-\mu^T\bar{x}} + 1$, which in turn matches that of $(\text{RP-CGLP})_k^p$ except for a constant term. □

Cadoux and Lemaréchal [2013] praise the boundedness from normalizing by an interior point, hence motivating what we denote as the Polar CGLP (P-CGLP):

$$\begin{array}{ll} \min & \alpha^T\bar{x} - \beta \\ \text{s.t.} & (\overline{\mathrm{C}})_{\mathrm{k}} \\ & \alpha^T p - \beta = 1 \end{array} \qquad (\text{P-CGLP})_k^p$$

We can similarly state $(\text{P-CGLP})_k^p$ as a CGLP for a problem on coordinates centered at p with right-hand side of -1, hence characterizing the α-projection as a subset of the polar $\left(\mathrm{P}^{\mathrm{K}} - p\right)^{\circ} := \{y : y^T(x-p) \leq 1 \ \forall x \in \mathrm{P}^{\mathrm{K}}\}$ when p is an interior point. Moreover, we can adapt the proof of Lemma 2.1 to show that cuts from $(\text{P-CGLP})_k^p$ define supporting hyperplanes. However, while normalization $\alpha^T p - \beta = 1$ defines a bounded feasible set if $p \in \text{int}(\mathrm{P}^{\mathrm{K}})$, a valid inequality might be active at p while separating $\bar{x}$ if $p \in \text{bd}(\mathrm{P}^{\mathrm{K}})$, in which case $(\text{P-CGLP})_k^p$ has no optimum. If P^{K} is not full-dimensional, there are no interior points.

While giving less importance to separating $\bar{x}$, Cadoux and Lemaréchal [2013] nevertheless regard that as a possible role for the objective function in the polar formulation. If separating $\bar{x}$ is of central importance, however, we show below that $(\text{RP-CGLP})_k^p$ is equivalent but more general than $(\text{P-CGLP})_k^p$ because it works with any $p \in \text{bd}(\mathrm{P}^{\mathrm{K}})$. When the happens, cuts from $(\text{P-CGLP})_k^p$ have to be derived from rays of the unbounded problem. Cadoux and Lemaréchal [2013] also present a concern with unbounded sets such as the reverse polar, which we address with the objective function evaluating a point in P^{K}.

Corollary 2.9. *If $(\text{P-CGLP})_k^p$ has an optimum, then $(\text{P-CGLP})_k^p$ and $(\text{RP-CGLP})_k^p$ derive the same cuts for $p \in \mathrm{P}^{\mathrm{K}}$.*

Proof. For $(\text{P-CGLP})^p_k$, the normalization implies $\theta = \frac{1}{\mu^T p - \nu}$ and the objective becomes $\min \alpha \bar{x} - \beta = \max \beta - \alpha \bar{x} = \max \theta[\nu - \mu^T \bar{x}] = \max \frac{\nu - \mu^T \bar{x}}{\mu^T p - \nu}$. Hence, unless p is active at some facet-defining inequality separating $\bar{x}$, we are maximizing a well-defined ratio. □

Finally, the work most closely related to ours is a parallel development by Conforti and Wolsey [2016]. After placing a point p from the relative interior at the origin, they maximize violation through a polar normalization with positive or zero slack. The latter is required when the disjunctive hull is not full-dimensional, hence generalizing $(\text{P-CGLP})^p_k$. In contrast, their work does not explore the ratios that define how cuts are ranked.

A poster with the results proven above was presented on May 2016 at the MIP Workshop, which was almost simultaneous with their presentation at the CORE@50 Conference.

2.3.2 Duality

The dual of a CGLP is regarded as the lift-and-project primal, where the solutions correspond to a convex combination of points on each term of the disjunction. The effect of each normalization constraint in the CGLP is to relax the lift-and-project in a different way to make $\bar{x}$ feasible, hence defining an infeasibility certificate. Ceria and Soares [1997] explore the interpretation of these duals in conventional CGLP formulations.

Among the formulations that we have proven equivalent above, the dual for $(\text{BP-CGLP})^p_k$ is particularly insightful because it yields the point alluded by Lemma 2.5 as $p' = x^0 + x^1$:

$$
\begin{array}{llrrcl}
\min & \omega & & & & \\
\text{s.t.} & & \tilde{A}x^0 & -\tilde{b}y_0 & \geq & 0 \\
 & & x^0_j & & \geq & 0 \\
 & & \tilde{A}x^1 & -\tilde{b}y_1 & \geq & 0 \\
 & & x^1_j & -y_1 & \geq & 0 \\
 & & & y_0 + y_1 & = & 1 \\
 & (\bar{x} - p)\omega & +x^0 + x^1 & & = & \bar{x}
\end{array}
\qquad (\text{BP-L\&P})^p_k
$$

2.3.3 Supporting Hyperplane Methods

There is a broader stream of literature dating back to Veinott [1967], which are often denoted as the supporting hyperplane methods. They are used for network design by Ben-Ameur and Neto [2007] and for mixed-integer nonlinear programs by Kronqvist et al. [2016], both considering a segment between an interior point p and $\bar{x}$ to obtain a boundary point p' and a cutting plane active at p'. Using a conventional CGLP, the in-out approach by Fischetti and Salvagnin [2010] separates another exterior point within that segment. For p sufficiently close to P^{K}, the latter approach is intuitively equivalent to ours.

The target cuts by Buchheim et al. [2008] exploit a projection where it is possible to enumerate all extreme points. Using a polar normalization and an objective maximizing violation, a linear program prevents inequalities from separating the extreme points and yields a facet-defining cut for the projection. This facet is active at the boundary point between p and $\bar{x}$. The method is later applied to solve robust network design [Buchheim et al., 2011] and quadratic integer programming [Buchheim et al., 2010]. The same idea is used by Tjandraatmadja and van Hoeve [2016] to generate facet-defining cuts with respect to the polytope associated with a decision diagram for a problem relaxation.

2.4 Cut Generation from the Simplex Tableau

When the CGLP is defined on a split disjunction with a single normalization constraint, Balas and Perregaard [2003] have shown that there is a correspondence between basic solutions of the CGLP defining a cut and those of the LP relaxation. Those results assume the restricted set of inequalities defined by $(\overline{\mathrm{C}})_{\mathrm{k}}$, where there are $2n+3$ basic variables in any basic CGLP solution defining a cut separating $\bar{x}$. The basic variables consist of α, β, u_0, v_0, and n multipliers among u and v. Furthermore, the basic variables among u and v correspond to linearly independent inequalities of the LP feasible set. From each of those basic multipliers, we infer that the slack of the corresponding inequality is non-basic at a basic LP solution where the same cut can be derived as a Gomory cut. These slacks correspond to the variables in x for the inequalities defining bounds.

In what follows, we show how to generate cuts from (RP-CGLP)$_k^p$ directly from the simplex tableau, hence using some results and proof steps from Balas and Perregaard [2003] when appropriate. Let $\bar{a}_{ij}$ denote the j-th column and $\bar{a}_{i0}$ the right-hand side of the i-th row of the simplex tableau for basic solution $\bar{x}$. The set $J = M_1 \cup M_2$ defines the index set of nonbasic variables of the LP for a given CGLP solution, where M_1 correspond to basic multipliers among u and M_2 to basic multipliers among v. Finally, let the slacks of the linear relaxation with respect to $\bar{x}$ and p as $\bar{s} = \tilde{A}\bar{x} - \tilde{b}$ and $\bar{\bar{s}} = \tilde{A}p - \tilde{b}$, respectively.

Theorem 2.10. *For a given basic solution of the LP relaxation, the reduced costs of non-basic multipliers u_i and v_i of (RP-CGLP)$_k^p$ for some row i for the corresponding given basic solution $(\bar{\alpha}, \bar{\beta}, \bar{u}, \bar{v})$ are*

$$r_{u_i} = \quad -\sigma \left[\sum_{j \in M_2} \bar{a}_{ij} \bar{s}_j - \bar{a}_{i0}(1 - \bar{x}_k) \right] - \left[\sum_{j \in M_2} \bar{a}_{ij} \bar{\bar{s}}_j - \bar{a}_{i0}(1 - p_k) \right] \tag{2.2}$$

and

$$r_{v_i} = -\sigma \left[\sum_{j \in M_1} \bar{a}_{ij} \bar{s}_j - \bar{a}_{i0} \bar{x}_k \right] - \left[\sum_{j \in M_1} \bar{a}_{ij} \bar{\bar{s}}_j - \bar{a}_{i0} p_k \right], \tag{2.3}$$

where the objective function value of the CGLP solution corresponds to

$$\sigma = -\frac{\sum_{j \in M_2} \bar{a}_{kj} \bar{\bar{s}}_j - \bar{a}_{k0}(1 - p_k)}{\sum_{j \in M_2} \bar{a}_{kj} \bar{s}_j - \bar{a}_{k0}(1 - \bar{x}_k)} \tag{2.4}$$

or

$$\sigma = -\frac{\sum_{j \in M_1} \bar{a}_{kj} \bar{\bar{s}}_j + (1 - \bar{a}_{k0}) p_k}{\sum_{j \in M_1} \bar{a}_{kj} \bar{s}_j + (1 - \bar{a}_{k0}) \bar{x}_k} \tag{2.5}$$

with $\bar{a}_{kj} \leq 0 \ \forall j \in M_1$ *and* $\bar{a}_{kj} \geq 0 \ \forall j \in M_2$*.*

Proof. By restricting to the basic multipliers in u and v along with one non-basic multiplier for each term, u_i and v_i, we have the following expressions for the cut coefficients:

$$\alpha = u_{M_1}^T \tilde{A}_{M_1} + u_i \tilde{A}_i - u_0 e_k = v_{M_2}^T \tilde{A}_{M_2} + v_i \tilde{A}_i + v_0 e_k \tag{2.6}$$

$$\beta = u_{M_1}^T \tilde{b}_{M_1} + u_i \tilde{b}_i = v_{M_2}^T \tilde{b}_{M_2} + v_i \tilde{b}_i + v_0 \tag{2.7}$$

After substitutions using Lemma 8 from Balas and Perregaard [2003], we have

$$u_j = -(u_0 + v_0)\bar{a}_{kj} + (u_i - v_i)\bar{a}_{ij} \qquad \forall j \in M_1 \tag{2.8}$$

$$v_j = (u_0 + v_0)\bar{a}_{kj} - (u_i - v_i)\bar{a}_{ij} \qquad \forall j \in M_2 \tag{2.9}$$

$$v_0 = (u_0 + v_0)\bar{a}_{k0} - (u_i - v_i)\bar{a}_{i0} \tag{2.10}$$

Since $u_0, v_0 > 0$, we show the partitioning among M_1 and M_2 by setting $u_i, v_i = 0$:

$$u_j = -(u_0 + v_0)\bar{a}_{kj} \qquad \rightarrow \bar{a}_{kj} \leq 0 \ \forall j \in M_1 \tag{2.11}$$

$$v_j = (u_0 + v_0)\bar{a}_{kj} \qquad \rightarrow \bar{a}_{kj} \geq 0 \ \forall j \in M_2 \tag{2.12}$$

Now we compute the slack of $\bar{x}$ with respect to the M_2 and also M_1:

$$\begin{aligned}
\alpha^T\bar{x} - \beta = & \quad v_{M_2}^T(\tilde{A}_{M_2}\bar{x} - \tilde{b}_{M_2}) + v_i(\tilde{A}_i\bar{x} - \tilde{b}_i) + v_0(e_k^T\bar{x} - 1) \\
= & \quad v_{M_2}^T\bar{s}_{M_2} + v_i\bar{s}_i + v_0(\bar{x}_k - 1) \\
= & \quad (u_0 + v_0)\sum_{j\in M_2}\bar{a}_{kj}\bar{s}_j - (u_i - v_i)\sum_{j\in M_2}\bar{a}_{ij}\bar{s}_j + v_i\bar{s}_i + [(u_0+v_0)\bar{a}_{k0} - (u_i - v_i)\bar{a}_{i0}](\bar{x}_k - 1) \\
= & \quad (u_0 + v_0)\left[\sum_{j\in M_2}\bar{a}_{kj}\bar{s}_j - \bar{a}_{k0}(1 - \bar{x}_k)\right] - (u_i - v_i)\left[\sum_{j\in M_2}\bar{a}_{ij}\bar{s}_j - \bar{a}_{i0}(1 - \bar{x}_k)\right] + v_i\bar{s}_i
\end{aligned} \tag{2.13}$$

$$\begin{aligned}
\alpha^T\bar{x} - \beta = & \quad u_{M_1}^T(\tilde{A}_{M_1}\bar{x} - \tilde{b}_{M_1}) + u_i(\tilde{A}_i\bar{x} - \tilde{b}_i) - u_0 e_k^T\bar{x} \\
= & \quad u_{M_1}^T\bar{s}_{M_1} + u_i\bar{s}_i - u_0\bar{x}_k \\
= & \quad -(u_0 + v_0)\sum_{j\in M_1}\bar{a}_{kj}\bar{s}_j + (u_i - v_i)\sum_{j\in M_1}\bar{a}_{ij}\bar{s}_j + u_i\bar{s}_i - [(u_0+v_0)(1 - \bar{a}_{k0}) + (u_i - v_i)\bar{a}_{i0}]\bar{x}_k \\
= & \quad -(u_0 + v_0)\left[\sum_{j\in M_1}\bar{a}_{kj}\bar{s}_j + (1 - \bar{a}_{k0})\bar{x}_k\right] + (u_i - v_i)\left[\sum_{j\in M_1}\bar{a}_{ij}\bar{s}_j - \bar{a}_{i0}\bar{x}_k\right] + u_i\bar{s}_i
\end{aligned} \tag{2.14}$$

We can obtain similar expressions with respect to p:

$$\alpha^T p - \beta = \quad (u_0 + v_0)\left[\sum_{j\in M_2}\bar{a}_{kj}\bar{\bar{s}}_j - \bar{a}_{k0}(1 - p_k)\right] - (u_i - v_i)\left[\sum_{j\in M_2}\bar{a}_{ij}\bar{\bar{s}}_j - \bar{a}_{i0}(1 - p_k)\right] + v_i\bar{\bar{s}}_i \tag{2.15}$$

$$\alpha^T p - \beta = \quad -(u_0 + v_0)\left[\sum_{j\in M_1}\bar{a}_{kj}\bar{\bar{s}}_j + (1 - \bar{a}_{k0})p_k\right] + (u_i - v_i)\left[\sum_{j\in M_1}\bar{a}_{ij}\bar{\bar{s}}_j - \bar{a}_{i0}p_k\right] + u_i\bar{\bar{s}}_i \tag{2.16}$$

This is the point where our proof differs from Balas and Perregaard [2003]. We use our normalization to determine the value of $u_0 + v_0$ and plug that in the objective function. We first use the expressions depending on M_2:

$$\alpha^T\bar{x} - \beta = \quad -1 \rightarrow \quad (u_0 + v_0) = \quad \frac{-1 + (u_i - v_i)\left[\sum_{j\in M_2}\bar{a}_{ij}\bar{s}_j - \bar{a}_{i0}(1 - \bar{x}_k)\right] - v_i\bar{s}_i}{\sum_{j\in M_2}\bar{a}_{kj}\bar{s}_j - \bar{a}_{k0}(1 - \bar{x}_k)} \tag{2.17}$$

$$\alpha^T p - \beta = \quad \frac{-1 + (u_i - v_i)\left[\sum_{j \in M_2} \bar{a}_{ij}\bar{s}_j - \bar{a}_{i0}(1-\bar{x}_k)\right] - v_i\bar{s}_i}{\sum_{j \in M_2} \bar{a}_{kj}\bar{s}_j - \bar{a}_{k0}(1-\bar{x}_k)} \left[\sum_{j \in M_2} \bar{a}_{kj}\bar{\bar{s}}_j - \bar{a}_{k0}(1-p_k)\right] - (u_i - v_i)\left[\sum_{j \in M_2} \bar{a}_{ij}\bar{\bar{s}}_j - \bar{a}_{i0}(1-p_k)\right] + v_i\bar{\bar{s}}_i \qquad (2.18)$$

Note that fixing $u_i, v_i = 0$ above yields the CGLP objective as in (2.4), whereas fixing only $v_i = 0$ and subtracting σ yields (2.2). Now we use the expressions depending on M_1:

$$\alpha^T \bar{x} - \beta = \quad -1 \rightarrow \quad (u_0 + v_0) = \quad \frac{1 + (u_i - v_i)\left[\sum_{j \in M_1} \bar{a}_{ij}\bar{s}_j - \bar{a}_{i0}\bar{x}_k\right] + u_i\bar{s}_i}{\left[\sum_{j \in M_1} \bar{a}_{kj}\bar{s}_j + (1-\bar{a}_{k0})\bar{x}_k\right]} \qquad (2.19)$$

$$\alpha^T p - \beta = \quad \frac{-1 - (u_i - v_i)\left[\sum_{j \in M_1} \bar{a}_{ij}\bar{s}_j - \bar{a}_{i0}\bar{x}_k\right] - u_i\bar{s}_i}{\left[\sum_{j \in M_1} \bar{a}_{kj}\bar{s}_j + (1-\bar{a}_{k0})\bar{x}_k\right]} \left[\sum_{j \in M_1} \bar{a}_{kj}\bar{\bar{s}}_j + (1-\bar{a}_{k0})p_k\right] + (u_i - v_i)\left[\sum_{j \in M_1} \bar{a}_{ij}\bar{\bar{s}}_j - \bar{a}_{i0}p_k\right] + u_i\bar{\bar{s}}_i \qquad (2.20)$$

Similarly, fixing $u_i, v_i = 0$ above yields the CGLP objective as in (2.5), whereas fixing $u_i = 0$ and subtracting σ yields (2.3). Note that it would also be possible to define r_{v_i} with respect to M_1 as well as r_{u_i} with respected to M_2, but both of these would leave the slack of a nonbasic multiplier in the expression, which does not need to be computed otherwise.

□

We can observe that σ resembles (2.1). In contrast, Balas and Perregaard [2003] showed that the tableau expression for (CGLP)$_k$ is $\sigma' = \dfrac{\sum_{j \in M_2} \bar{a}_{kj}\bar{s}_j - \bar{a}_{k0}(1-\bar{x}_k)}{1 + \sum_{j \in J} |\bar{a}_{kj}|}$, where the denominator evidences the effect of coefficient scale pointed out by Fischetti et al. [2011].

Note that choosing which variable enters the CGLP basic solution corresponds to choosing which variable leaves the LP basic solution: solving the CGLP through the LP implies dualizing the solution method. However, LP solutions do not need to be feasible. In fact, it is very common for a CGLP solution to be associated with an infeasible LP solution.

If we decide to put u_i or v_i into the CGLP basis, we need to pivot out x_i from the LP basis and replace it by some other variable x_l. Consequently, we are changing the coefficients of the non-basic variables in the line defining x_k and thus the cut that we obtain.

To simplify notation, we assume x_k is defined by row k, x_i by row i, and s_j denotes a nonbasic variable in the LP that is a basic multiplier in the CGLP. Hence, the rows associated with x_k and

x_i in the LP relaxation can be denoted as

$$x_k + \sum_{j \in J} \bar{a}_{kj} s_j = \bar{a}_{k0} \tag{2.21}$$

$$x_i + \sum_{j \in J} \bar{a}_{ij} s_j = \bar{a}_{i0} \tag{2.22}$$

Corollary 2.11. *Given an LP basis where $\lfloor \bar{x}_k \rfloor < x_k < \lceil \bar{x}_k \rceil$ and variable x_i leaves the basis, pivoting a non-basic variable x_l preserves $\lfloor \bar{x}_k \rfloor < x_k < \lceil \bar{x}_k \rceil$ if $\frac{\lfloor \bar{x}_k \rfloor - \bar{a}_{k0}}{\bar{a}_{i0}} < \gamma_l < \frac{\lceil \bar{x}_k \rceil - \bar{a}_{k0}}{\bar{a}_{i0}}$, where $\gamma_l = -\frac{\bar{a}_{kl}}{\bar{a}_{il}}$. If $\gamma_l > 0$, the corresponding improvement in the objective function of (RP-CGLP)$_k^p$ is given by*

$$f^+(\gamma) := \qquad -\frac{\sum_{j \in J} \left(\min\{0, \bar{a}_{kj} + \gamma \bar{a}_{ij}\}\right) \bar{\bar{s}}_j + (1 - \bar{a}_{k0} - \gamma \bar{a}_{ij}) p_k}{\sum_{j \in J} \left(\min\{0, \bar{a}_{kj} + \gamma \bar{a}_{ij}\}\right) \bar{s}_j + (1 - \bar{a}_{k0} - \gamma \bar{a}_{ij}) \bar{x}_k} - \sigma.$$

Otherwise, if $\gamma_l < 0$,

$$f^-(\gamma) := \qquad -\frac{\sum_{j \in J} \left(\max\{0, \bar{a}_{kj} + \gamma \bar{a}_{ij}\}\right) \bar{\bar{s}}_j - (\bar{a}_{k0} + \gamma \bar{a}_{i0})(1 - p_k)}{\sum_{j \in J} \left(\max\{0, \bar{a}_{kj} + \gamma \bar{a}_{ij}\}\right) \bar{s}_j - (\bar{a}_{k0} + \gamma \bar{a}_{i0})(1 - \bar{x}_k)} - \sigma.$$

The pivot operation yields no improvement if $\gamma_l = 0$.

Proof. If we add row i multiplied by some $\gamma > 0$ to row k, we obtain

$$x_k + \gamma x_i + \sum_{j \in J} (\bar{a}_{kj} + \gamma \bar{a}_{ij}) s_j = \bar{a}_{k0} + \gamma \bar{a}_{i0}$$

and the right-hand side remains in the range $(\lfloor \bar{x}_k \rfloor, \lceil \bar{x}_k \rceil)$ if $\frac{\lfloor \bar{x}_k \rfloor - \bar{a}_{k0}}{\bar{a}_{i0}} < \gamma < \frac{\lceil \bar{x}_k \rceil - \bar{a}_{k0}}{\bar{a}_{i0}}$.

If x_i is pivoted out and replaced by the variable in the l-th column, then setting the coefficient of that variable to 0 in the k-th row requires $\gamma = -\frac{\bar{a}_{kl}}{\bar{a}_{il}} = \gamma_l$.

The impact of such pivot on (RP-CGLP)$_k^p$ depends on γ_l being positive or negative. If $\gamma_l > 0$, column i joins M_2. In such case, M_1 remains a subset of the non-basic variables from the previous basis, which corresponds to those variables with nonpositive coefficients in the k-th row. Hence, the objective function of (RP-CGLP)$_k^p$ after the pivot is given by

$$-\frac{\sum_{j \in J} \left(\min\{0, \bar{a}_{kj} + \gamma \bar{a}_{ij}\}\right) \bar{\bar{s}}_j + (1 - \bar{a}_{k0} - \gamma \bar{a}_{ij}) p_k}{\sum_{j \in J} \left(\min\{0, \bar{a}_{kj} + \gamma \bar{a}_{ij}\}\right) \bar{s}_j + (1 - \bar{a}_{k0} - \gamma \bar{a}_{ij}) \bar{x}_k}$$

Otherwise, if $\gamma_l < 0$, column i joins M_1 and the objective with respect to M_2 becomes

$$-\frac{\sum_{j \in J} \left(\max\{0, \bar{a}_{kj} + \gamma \bar{a}_{ij}\}\right) \bar{\bar{s}}_j - (\bar{a}_{k0} + \gamma \bar{a}_{i0})(1 - p_k)}{\sum_{j \in J} \left(\max\{0, \bar{a}_{kj} + \gamma \bar{a}_{ij}\}\right) \bar{s}_j - (\bar{a}_{k0} + \gamma \bar{a}_{i0})(1 - \bar{x}_k)}$$

Note that there is no change in σ if $\gamma_l = 0$ because the k-th row remains the same. □

Finally, we observe that there are LP basic solutions corresponding to solutions with negative objective for $(\text{RP-CGLP})_k^p$, which are those yielding inequalities that do not separate $\bar{x}$. These solutions have no correspondence in the CGLP because they are removed by the normalization constraint. When using the LP relaxation to generate the cut, one should not pivot to such bases. If we keep using $p \in \mathrm{P}^{\mathrm{K}}$, they are easily spotted through the objective function of the CGLP.

2.5 Computational Experiments

This section compares the cuts generated by solving $(\text{CGLP})_k$ and $(\text{RP-CGLP})_k^p$ through the tableau of the linear relaxation. We use the implementation for $(\text{CGLP})_k$ described in [Balas and Bonami, 2009] and adapt it to also generate cuts using $(\text{RP-CGLP})_k^p$ with two different methods to generate the point p. In the first, we take a linear combination of points maximizing the minimum slack on each term of the disjunction, using a point along any unbonded direction in the case that all slacks can be made arbitarily large. In the second, we restrict the objective to the slacks of constraints containing x_k and normalize them by the coefficient of x_k in the constraint.

In the experiments, we solve the linear relaxation of instances from the MIPLIB benchmarks[1] and then generate a lift-and-project cut for each fractional variable x_k using either $(\text{CGLP})_k$ or $(\text{RP-CGLP})_k^p$, which is then strengthened, up to a limit of 300 distinct cuts. We circumvent most of the numerical issues that arise when the objective function gets too close to zero by reverting to the conventional formulation in those cases. As we generate the cuts for each instance using each formulation, we measure the total gap closed when adding all the cuts generated and resolving the linear relaxation, the average Euclidean distance of each cutting plane to $\bar{x}$, the number of cuts discarded by the generator. The gap closed here refers to the optimal value of the linear relaxation after adding the cuts in comparison with the optimal value of the linear relaxation with no cuts and the known optimal value of each instance. Finally, we try solving the problem on CPLEX with the cuts from each method added and automatic cut generation disabled and a time limit of one hour, from which we report either the solving time or else the remaining gap after one hour.

All code is written in C++ (gcc version 4.8.24) and ran in Ubuntu 14.04.2 LTS on a machine with an Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz processor and 128 GB of RAM.

Table 2.1 summarizes the results, where conventional refers to using $(\text{CGLP})_k$, reformulation 1 to using $(\text{RP-CGLP})_k^p$ with the first method to generate p, and reformulation 2 to using (RP-

[1] miplib.zib.de

CGLP)$_k^p$ with the second method. On the one hand, we note that the gap closed and the average Euclidean distance of the cuts with the reformulation are often smaller. The reformulation with the first method closes a larger gap in 26 instances and a smaller one in 36 instances, while with the second it closes a larger gap in 26 instances and a smaller one in 45. With respect to the average Euclіean distance, the first method leads to better results in 38 instances against 51, and the second is better in 29 instances against 65. In addition, more cuts are discarded for the sake of numerical safety: from 258 with the convetional formulation, we observe a total of 292 with the first method and 320 with the second method. On the other hand, we note a different outcome when comparing the performance of the formulation with the cuts on CPLEX: while the first method performs better in 45 cases against 61, the second performs better in 64 cases against 42.

2.6 Conclusion

This chapter introduced the Reverse Polar Cut Generating Linear Program (RP-CGLP), which is parameterized by a point $\bar{x}$ that we want to separate and a point p that we cannot. We have shown that these lift-and-project cuts define supporting hyperplanes of the immediate closure. When that closure if full-dimensional, the cutting plane is a combination of facets that are active at the point that a ray from $\bar{x}$ to p first intersects the closure, with each facet separating $\bar{x}$ also corresponding to an optimal solution to RP-CGLP. Finally, we proved that RP-CGLP is equivalent to some other CGLPs and presented some computational results from a tableau implementation.

While we switch the roles of normalization and objective in comparison to other CGLP formulations, we nevertheless observe that a distortion in how the cuts are compared with respect to the objective is unavoidable. We fix violation to guarantee separability and then choose to minimize the slack for p to ensure boundedness. That intuitively favors cuts that are farther away from $\bar{x}$ and closer to p. In fact, we show that RP-CGLP actually minimizes the ratio between slack for p and violation for $\bar{x}$ across all valid cuts, consequently proving the equivalence between RP-CGLP and other recent CGLP formulations. Previously, the CGLPs for which an explicit representation of the objective was known were those fixing a norm of α. Moreover, in comparison to the equivalent CGLP formulations that have been recently proposed, RP-CGLP has the benefit of preserving the feasible set for different choices of p, whereas changing p affects the left-hand side of those formulations. Hence, RP-CGLP may facilitate generating multiple cuts because we only need to reoptimize after changing p. In addition, we could possibly use sensitivity analysis on the objective of RP-CGLP to look for points in the disjunctive hull yielding a disjoint set of cuts. However, finding a good choice for such point p is a topic that deserves further attention.

The experimental results have shown that there is some potential for RP-CGLP, but also room for improvement. The total gap closed and the average distance of the cutting planes is smaller when p is generated with either method proposed. This is intuitive given that these cutting planes are only a combination of facets separating $\bar{x}$ along the ray towards p instead of a valid inequality

Table 2.1: Comparison of total gap closed (%), average Euclidean distance of the cuts, number of discarded cuts, and runtime in seconds to solve the problem after adding the generated cuts.

	Conventional				Reformulation 1				Reformulation 2			
Instance	Gap	Avg. dist.	Lost	CPLEX	Gap	Avg. dist.	Lost	CPLEX	Gap	Avg. dist.	Lost	CPLEX
10teams	57.1 %	0.021	0	40.1 s	57.1 %	0.021	0	58.1 s	57.1 %	0.021	0	71.1 s
a1c1s1	22.1 %	0.911	1	30.1 %	19.1 %	0.851	7	31.1 %	19.1 %	0.851	7	31.1 %
aflow30a	20.1 %	0.061	0	48.1 s	20.1 %	0.061	0	42.1 s	18.1 %	0.041	0	30.1 s
aflow40b	12.1 %	0.021	0	1821.1 s	12.1 %	0.021	0	1687.1 s	11.1 %	0.021	0	872.1 s
air01	100.1 %	0.031	0	0.1 s	100.1 %	0.031	0	0.1 s	100.1 %	0.031	0	0.1 s
air02	75.1 %	0.011	0	3.1 s	75.1 %	0.011	0	0.1 s	75.1 %	0.011	0	1.1 s
air03	100.1 %	0.011	0	2.1 s	100.1 %	0.011	0	3.1 s	100.1 %	0.011	0	2.1 s
air04	11.1 %	0.001	26	328.1 s	12.1 %	0.001	26	470.1 s	12.1 %	0.001	26	212.1 s
air05	5.1 %	0.001	0	84.1 s	5.1 %	0.001	0	60.1 s	5.1 %	0.001	0	26.1 s
air06	44.1 %	0.001	0	25.1 s	45.1 %	0.001	0	48.1 s	45.1 %	0.001	0	39.1 s
arki001	28.1 %	0.171	7	32.1 s	28.1 %	0.171	7	49.1 s	29.1 %	0.161	18	51.1 s
atlanta-ip	1.1 %	0.021	14	3401.1 s	1.1 %	0.021	12	1.1 %	1.1 %	0.021	21	3.1 %
bell3a	52.1 %	0.541	0	3.1 s	52.1 %	0.581	1	10.1 s	51.1 %	0.531	0	7.1 s
bell3b	47.1 %	0.541	0	8.1 s	47.1 %	0.551	0	2.1 s	47.1 %	0.551	0	1.1 s
bell4	29.1 %	0.561	0	6.1 s	29.1 %	0.561	0	7.1 s	29.1 %	0.561	0	2.1 s
bell5	85.1 %	0.481	2	24.1 s	85.1 %	0.481	2	18.1 s	85.1 %	0.481	1	19.1 s
blend2	16.1 %	0.111	0	16.1 s	16.1 %	0.111	0	13.1 s	16.1 %	0.111	0	37.1 s
bm23	17.1 %	0.121	0	13.1 s	13.1 %	0.131	0	9.1 s	8.1 %	0.061	0	4.1 s
cap6000	32.1 %	0.011	0	110.1 s	37.1 %	0.011	0	53.1 s	42.1 %	0.011	0	59.1 s
cracpb1	0.1 %	0.051	0	1.1 s	0.1 %	0.021	0	1.1 s	0.1 %	0.021	0	0.1 s
danoint	2.1 %	0.111	26	681.1 s	2.1 %	0.071	29	598.1 s	2.1 %	0.011	31	559.1 s
dcmulti	38.1 %	1.521	10	11.1 s	38.1 %	1.241	10	12.1 s	38.1 %	1.221	10	9.1 s
ds	1.1 %	0.001	1	74.1 %	1.1 %	0.001	1	76.1 %	1.1 %	0.001	1	77.1 %
dsbmip	0.1 %	0.471	12	6.1 s	0.1 %	0.481	12	8.1 s	0.1 %	0.501	15	1.1 s
egout	55.1 %	0.881	0	12.1 s	55.1 %	0.901	0	10.1 s	55.1 %	0.901	0	5.1 s
fast0507	2.1 %	0.001	0	1640.1 s	3.1 %	0.001	0	682.1 s	1.1 %	0.001	0	599.1 s
fast0507	2.1 %	0.001	0	479.1 s	1.1 %	0.001	0	896.1 s	3.1 %	0.001	0	736.1 s
fiber	39.1 %	0.051	0	48.1 s	39.1 %	0.051	0	49.1 s	38.1 %	0.051	0	24.1 s
fixnet3	75.1 %	0.731	0	25.1 s	74.1 %	0.731	0	27.1 s	74.1 %	0.731	0	19.1 s
fixnet4	28.1 %	0.731	0	21.1 s	26.1 %	0.731	0	25.1 s	26.1 %	0.731	0	11.1 s
fixnet6	11.1 %	0.731	0	27.1 s	11.1 %	0.731	0	47.1 s	11.1 %	0.731	0	34.1 s
flugpl	11.1 %	0.351	2	5.1 s	11.1 %	0.351	2	6.1 s	11.1 %	0.351	2	6.1 s
gen	69.1 %	0.051	0	5.1 s	69.1 %	0.051	0	12.1 s	69.1 %	0.051	0	10.1 s
gesa2	28.1 %	0.161	4	56.1 s	25.1 %	0.221	20	45.1 s	25.1 %	0.221	20	99.1 s
gesa2_o	30.1 %	0.211	0	40.1 s	30.1 %	0.201	0	50.1 s	30.1 %	0.201	0	29.1 s
gesa3	61.1 %	0.091	21	4.1 s	28.1 %	0.081	28	13.1 s	28.1 %	0.081	30	17.1 s

Continuation of Table 2.1

Instance	Conventional				Reformulation 1				Reformulation 2			
	Gap	Avg. dist.	Lost	CPLEX	Gap	Avg. dist.	Lost	CPLEX	Gap	Avg. dist.	Lost	CPLEX
gesa3_o	73.1 %	0.131	4	9.1 s	52.1 %	0.121	4	3.1 s	52.1 %	0.121	5	6.1 s
glass4	0.1 %	0.961	0	124.1 s	0.1 %	0.961	0	130.1 s	0.1 %	0.961	0	102.1 s
gt2	92.1 %	0.581	0	14.1 s	92.1 %	0.581	0	13.1 s	92.1 %	0.581	0	7.1 s
harp2	26.1 %	0.031	0	129.1 s	23.1 %	0.021	0	158.1 s	23.1 %	0.021	0	106.1 s
khb05250	75.1 %	0.641	0	26.1 s	75.1 %	0.641	0	23.1 s	75.1 %	0.641	0	19.1 s
l152lav	3.1 %	0.011	0	28.1 s	5.1 %	0.011	0	7.1 s	3.1 %	0.011	0	27.1 s
lp4l	51.1 %	0.021	0	7.1 s	88.1 %	0.021	0	0.1 s	94.1 %	0.021	0	0.1 s
lseu	56.1 %	0.111	0	8.1 s	56.1 %	0.071	0	16.1 s	56.1 %	0.091	0	7.1 s
manna81	100.1 %	0.291	0	0.1 s	100.1 %	0.291	0	0.1 s	100.1 %	0.291	0	0.1 s
markshare1	0.1 %	0.031	0	100.1 %	0.1 %	0.031	0	100.1 %	0.1 %	0.031	0	100.1 %
markshare2	0.1 %	0.031	0	100.1 %	0.1 %	0.031	0	100.1 %	0.1 %	0.031	0	100.1 %
mas74	7.1 %	0.011	0	573.1 s	6.1 %	0.011	0	686.1 s	5.1 %	0.011	0	516.1 s
mas76	6.1 %	0.011	0	43.1 s	4.1 %	0.021	0	83.1 s	5.1 %	0.011	0	53.1 s
misc01	3.1 %	0.141	0	5.1 s	3.1 %	0.141	0	6.1 s	3.1 %	0.121	0	3.1 s
misc02	5.1 %	0.141	0	5.1 s	5.1 %	0.131	0	5.1 s	5.1 %	0.141	0	1.1 s
misc03	9.1 %	0.071	0	6.1 s	9.1 %	0.071	0	16.1 s	9.1 %	0.061	0	10.1 s
misc04	19.1 %	0.121	3	4.1 s	16.1 %	0.081	4	4.1 s	16.1 %	0.081	4	3.1 s
misc05	24.1 %	0.091	0	8.1 s	24.1 %	0.091	0	10.1 s	24.1 %	0.091	0	2.1 s
misc06	30.1 %	0.031	3	6.1 s	30.1 %	0.031	3	7.1 s	30.1 %	0.031	5	14.1 s
misc07	0.1 %	0.051	0	18.1 s	0.1 %	0.051	0	35.1 s	0.1 %	0.041	0	17.1 s
mitre	85.1 %	0.041	1	0.1 s	85.1 %	0.041	1	0.1 s	85.1 %	0.041	1	0.1 s
mkc	15.1 %	0.081	0	0.1 %	13.1 %	0.051	0	0.1 %	11.1 %	0.051	0	0.1 %
mkc	15.1 %	0.081	0	0.1 %	13.1 %	0.061	0	0.1 %	11.1 %	0.051	0	0.1 %
mod008	20.1 %	0.021	0	39.1 s	4.1 %	0.021	0	15.1 s	4.1 %	0.021	0	12.1 s
mod010	6.1 %	0.021	0	4.1 s	70.1 %	0.011	0	0.1 s	70.1 %	0.021	0	1.1 s
mod011	19.1 %	0.591	0	31.1 s	32.1 %	0.321	0	25.1 s	32.1 %	0.301	0	31.1 s
mod013	4.1 %	0.481	0	18.1 s	4.1 %	0.501	0	17.1 s	4.1 %	0.491	0	8.1 s
modglob	15.1 %	0.541	0	81.1 s	17.1 %	0.561	0	95.1 s	19.1 %	0.561	0	41.1 s
momentum1	45.1 %	0.151	67	4.1 %	45.1 %	0.161	67	0.1 %	65.1 %	0.141	67	0.1 %
momentum2	43.1 %	0.181	9	356.1 s	43.1 %	0.171	9	668.1 s	43.1 %	0.181	9	2764.1 s
msc98-ip	9.1 %	0.061	1	1818.1 s	8.1 %	0.061	1	1995.1 s	12.1 %	0.041	1	9.1 %
mzzv11	29.1 %	0.031	1	74.1 s	28.1 %	0.031	1	99.1 s	32.1 %	0.021	1	160.1 s
mzzv42z	22.1 %	0.021	1	49.1 s	22.1 %	0.021	1	57.1 s	21.1 %	0.021	1	40.1 s
net12	9.1 %	0.141	1	790.1 s	9.1 %	0.141	1	586.1 s	7.1 %	0.071	0	429.1 s
noswot	0.1 %	0.141	3	36.1 s	0.1 %	0.141	3	26.1 s	0.1 %	0.141	3	16.1 s
nsrand-ipx	35.1 %	0.011	0	2662.1 s	32.1 %	0.001	0	0.1 %	27.1 %	0.001	0	0.1 %

Continuation of Table 2.1

	Conventional				Reformulation 1				Reformulation 2			
Instance	Gap	Avg. dist.	Lost	CPLEX	Gap	Avg. dist.	Lost	CPLEX	Gap	Avg. dist.	Lost	CPLEX
nw04	62.1 %	0.001	0	18.1 s	62.1 %	0.001	0	17.1 s	62.1 %	0.001	0	19.1 s
opt1217	5.1 %	0.071	0	24.1 %	0.1 %	0.071	0	25.1 %	26.1 %	0.081	0	18.1 %
p0033	57.1 %	0.371	0	4.1 s	57.1 %	0.371	0	4.1 s	13.1 %	0.341	0	3.1 s
p0040	100.1 %	0.101	0	1.1 s	100.1 %	0.101	0	0.1 s	100.1 %	0.101	0	0.1 s
p0201	41.1 %	0.041	0	4.1 s	41.1 %	0.041	0	5.1 s	41.1 %	0.041	0	3.1 s
p0282	4.1 %	0.151	0	8.1 s	4.1 %	0.151	0	8.1 s	8.1 %	0.141	0	13.1 s
p0291	41.1 %	0.241	0	11.1 s	28.1 %	0.181	0	10.1 s	6.1 %	0.161	0	3.1 s
p0548	41.1 %	0.601	0	10.1 s	41.1 %	0.601	0	13.1 s	41.1 %	0.591	0	8.1 s
p2756	1.1 %	0.431	0	68.1 s	1.1 %	0.431	0	95.1 s	1.1 %	0.431	0	41.1 s
p6000	32.1 %	0.011	0	94.1 s	37.1 %	0.011	0	60.1 s	42.1 %	0.011	0	15.1 s
pipex	28.1 %	0.151	0	9.1 s	28.1 %	0.151	0	9.1 s	28.1 %	0.151	0	2.1 s
pk1	0.1 %	0.011	0	47.1 s	0.1 %	0.011	0	65.1 s	0.1 %	0.011	0	66.1 s
pp08a	51.1 %	0.811	0	550.1 s	51.1 %	0.811	0	418.1 s	42.1 %	0.801	0	1277.1 s
pp08aCUTS	40.1 %	0.281	1	9.1 s	30.1 %	0.271	2	13.1 s	28.1 %	0.231	2	12.1 s
protfold	31.1 %	0.051	1	45.1 %	6.1 %	0.031	1	38.1 %	6.1 %	0.021	1	40.1 %
qnet1	22.1 %	0.041	0	15.1 s	15.1 %	0.011	0	15.1 s	13.1 %	0.021	0	20.1 s
qnet1_o	50.1 %	0.191	0	18.1 s	46.1 %	0.141	0	6.1 s	47.1 %	0.141	0	26.1 s
rd-rplusc-21	0.1 %	0.171	29	764.1 s	0.1 %	0.151	30	944.1 s	0.1 %	0.141	30	360.1 s
rentacar	24.1 %	0.971	6	15.1 s	24.1 %	0.971	6	11.1 s	24.1 %	0.991	7	15.1 s
rgn	5.1 %	0.091	0	7.1 s	10.1 %	0.121	0	11.1 s	12.1 %	0.131	0	6.1 s
roll3000	28.1 %	0.081	0	3.1 %	28.1 %	0.081	0	2.1 %	24.1 %	0.071	0	3.1 %
rout	3.1 %	0.081	0	87.1 s	3.1 %	0.071	0	53.1 s	2.1 %	0.071	0	80.1 s
sample2	6.1 %	0.591	0	11.1 s	6.1 %	0.591	0	10.1 s	6.1 %	0.591	0	3.1 s
sentoy	19.1 %	0.031	0	5.1 s	20.1 %	0.051	0	10.1 s	16.1 %	0.041	0	5.1 s
set1ch	38.1 %	0.821	0	14.1 %	38.1 %	0.811	0	15.1 %	36.1 %	0.781	0	15.1 %
seymour	13.1 %	0.071	1	2.1 %	13.1 %	0.071	1	2.1 %	11.1 %	0.061	1	2.1 %
sp97ar	16.1 %	0.001	0	1.1 %	16.1 %	0.001	0	1.1 %	14.1 %	0.001	0	1.1 %
stein9	0.1 %	0.211	0	2.1 s	0.1 %	0.221	0	3.1 s	0.1 %	0.191	0	1.1 s
stein15	0.1 %	0.251	0	2.1 s	0.1 %	0.261	0	5.1 s	0.1 %	0.251	0	2.1 s
stein27	0.1 %	0.241	0	5.1 s	0.1 %	0.241	0	6.1 s	0.1 %	0.241	0	16.1 s
stein45	0.1 %	0.301	0	33.1 s	0.1 %	0.181	0	45.1 s	0.1 %	0.221	0	44.1 s
swath	28.1 %	0.071	0	18.1 %	13.1 %	0.031	0	19.1 %	13.1 %	0.041	0	18.1 %
timtab1	24.1 %	0.301	0	15.1 %	24.1 %	0.301	0	14.1 %	24.1 %	0.301	0	22.1 %
timtab2	18.1 %	0.231	0	51.1 %	18.1 %	0.231	0	50.1 %	18.1 %	0.231	0	55.1 %
vpm1	10.1 %	0.141	0	25.1 s	10.1 %	0.131	0	25.1 s	10.1 %	0.131	0	28.1 s
vpm2	19.1 %	0.071	0	18.1 s	15.1 %	0.061	0	22.1 s	14.1 %	0.051	0	27.1 s

that maximizes violation for some scale of the constraints. However, one could argue that there is a way to choose p in each case for which the same cuts from the conventional formulation are obtained, or else strictly better ones are found in the case that they do not define supporting hyperplanes. In addition, we also observe that more cuts are discarded with the reformulation. Finally, there is a noticeable difference in the results according to the method used to generate p, especially for the second experiment of testing the performance on CPLEX with those cuts and automatic cut generation disabled. While in both cases the outcome is more favorable than that observed with respect to the total gap or the average distance, the conventional formulation is only beat by this second method that maximizes the scaled slacks of constraints containing the variable defining the disjunction. A better performance in this case is also intuitive, given that the points generated with respect to each disjunction are expected to be more distinct from one another.

Ultimately, one could argue that the reformulation shifts where the numerical issues are. While the concept of a most violated cut depends on an adequate scale of the constraints in the conventional formulation, the family of equivalent in-out formulations to which RP-CGLP belongs depends on a careful choice of point or ray parameterizing direction of separation. Therefore, an important milestone for these approaches is finding points in the disjunctive hull that yield strictly better cuts. One would expect the ideal point to define a CGLP optima with unique (α, β)-projection, which in turn derives a cut $\alpha^T x \geq \beta$ defining a facet of the disjunctive hull. When the disjunctive hull is not full-dimensional and the point is inevitably at the boundary, further restricting the CGLP by facial reduction [Borwein and Wolkowicz, 1981] may possibly attenuate numerical issues associated with small perturbations. In a sense, while we are able to obtain cuts defining supporting hyperplanes, their quality ultimately depends on how p is chosen and the space is restricted for each disjunction.

Chapter 3

Checking the Regularity of Lift-and-Project Cuts

This chapter is based on the manuscript "When Lift-and-Project Cuts are Different" [Balas and Serra, 2018], which is under preparation for submission.

3.1 Introduction

Many techniques to generate cutting planes for a Mixed-Integer Linear Program (MILP) are equivalent to one another under certain conditions. Since some are more general and usually more expensive computationally, it is important to determine if and when they generate cuts that others cannot. In this chapter, we introduce a technique to verify if a lift-and-project cut [Balas et al., 1993, 1996] from an arbitrary disjunction corresponds to a standard intersection cut [Balas, 1971] and analyze computational results on several instances for insights. This study is particularly relevant due to the recent research activity around cuts from multiple rows of the simplex tableau since their introduction by Andersen et al. [2007b], which in the case of q rows are equivalent to disjunctive cuts from a 2^q-term disjunction [Balas and Qualizza, 2013]. More specifically, it helps us find out about the convese: how often lift-and-project cuts from multi-term disjunctions cannot be directly obtained from the simplex tableau.

We can obtain a lift-and-project cut by solving a Cut Generating Linear Program (CGLP), which defines valid inequalities for an MILP relaxation consisting of a Disjunctive Program (DP) [Balas, 1979, 1998]. Such DPs are often unions of disjoint polyhedra that exclude the region around a particular solution $x = \bar{x}$ of the Linear Program (LP) relaxation. The most common DPs consist of intersecting the LP relaxation with a *split disjunction* of the form $\{x : \pi x \leq \pi_0\} \cup \{x : \pi x \geq \pi_0 + 1\}$ when $\pi_0 < \pi\bar{x} < \pi_0 + 1$ and no MILP solution is removed by the disjunction. For that case, Balas and Perregaard [2003] have shown that there is a correspondence between basic CGLP solutions and intersection cuts from basic solutions of the LP relaxation, feasible or not. That entails a more efficient procedure to implicitly solve CGLPs from split disjunctions by pivoting among LP bases, which has been implemented in a number solvers including CglLandP [Balas and Bonami, 2009] in COIN-OR [COIN].

The equivalence identified by Balas and Perregaard [2003] is particularly useful because lift-and-project cuts from a *simple* split disjunction of the form $\{x : x_k \leq 0\} \cup \{x : x_k \geq 1\}$ map to Gomory fractional cuts [Gomory, 1958] from the row of the tableau defining the value of x_k. Similarly, strengthening those lift-and-project cuts by changing the coefficients associated with integer nonbasic variables [Balas and Jeroslow, 1980, Balas et al., 1993] makes them equivalent to Gomory Mixed-Integer (GMI) cuts [Gomory, 1960] from the corresponding row of the tableau.

More recently, Balas and Kis [2016] have shown that the correspondence between lift-and-project cuts and intersection cuts may also hold for some lift-and-project cuts from arbitrary disjunctions. More specifically, they have proven that it holds if, and only if, there is a basic CGLP solution associated with the cut that maps to an LP basis where it corresponds to a standard intersection cut. These basic CGLP solutions mapping to LP bases are denoted *regular*. A lift-and-project cut is then denoted *regular* if there exists a corresponding regular basic CGLP solution and *irregular* otherwise.

The elegance and convenience of generating cuts from the simplex tableau has motivated a recent stream of theoretical work on generating cuts from two rows of the simplex tableau [Andersen et al., 2007b, Cornuéjols and Margot, 2008] and subsequently more rows [Borozan and Cornuéjols, 2009, Basu et al., 2010], on how these cuts can be strengthened when nonbasic variables are integer [Dey and Wolsey, 2010, Conforti et al., 2011a, Basu et al., 2013, Fukasawa et al., 2016], and several other variants. The reader is referred to Conforti et al. [2011b] and Basu et al. [2015] for a broader review of this line of work, which has been accompanied by extensive computational experimentation [Espinoza, 2010, Basu et al., 2011, Dey et al., 2014, Louveaux et al., 2015].

However, there are other ways in which one can exploit that more than one integer variable is fractional in $\bar{x}$. For example, we can generate lift-and-project cuts using disjunctions with more than two terms, and those may yield irregular cuts instead. A natural generalization of the commonly used split disjunction is defined by Li and Richard [2008] as the *t-branch split disjunction*

$$\bigcup_{S \subseteq \{1,2,\ldots,t\}} \left\{x : \pi^i x \leq \pi_0^i, \text{ if } i \in S;\ \pi^i x \geq \pi_0^i + 1, \text{ if } i \notin S\right\},$$

of which the cross disjunction [Dash et al., 2012] corresponds to the special case where $t = 2$. Dash et al. [2014] observed that the resulting cuts close a substantially larger gap in comparison to split cuts, thus making it important to identify when they are likely to be distinct from regular cuts. Furthermore, Andersen et al. [2005] as well as Kis [2014] have shown examples of disjunctive cuts that do not correspond to intersection cuts.

In this work, we focus on the equivalence with respect to lift-and-project cuts without strengthening, which is determined by solving an MILP formulation based on the CGLP. We note that mixed-integer formulations have already been used for similar purposes, such as generating Chvátal-Gomory cuts [Fischetti and Lodi, 2007].

In this work, we focus on the equivalence with respect to lift-and-project cuts without strengthening, which is determined by solving an MILP formulation based on the CGLP. Mixed-integer formulations have already been used for similar purposes, such as generating Chvátal-Gomory cuts [Fischetti and Lodi, 2007], and may help us understand the structure of families of instances.

This chapter presents the following contributions. First, we state a result that simplifies the verification of regularity for basic CGLP solutions from Balas and Kis [2016] and show that it can also be applied to CGLP solutions that are not basic in Section 3.3. Second, we introduce and prove the validity of an MILP that checks whether there is a regular CGLP solution for a given cut in Section 3.4. Third, we describe a numerical procedure based on such MILP that verifies if a lift-and-project cut is regular or not in Section 3.5. Finally, we present computational results from 74 benchmark instances in Section 3.6, and we use these results to analyze what factors may lead to a higher incidence of irregular cuts and how these cuts compare with regular ones in Section 3.7.

3.2 Preliminaries

Let us consider a mixed $0-1$ linear program with rational coefficients

$$\min \Big\{ cx \ : \ Ax \geq b, x \geq 0, \ x_j \in \{0,1\}, \ j = 1, \dots, p \Big\}, \tag{$\mathcal{P}$}$$

where A is an $m \times n$ matrix. Let $\tilde{A}x \geq \tilde{b}$ denote the feasible set of the LP relaxation $P := \{x : Ax \geq b, x \geq 0, x_j \leq 1, j = 1, \dots, p\}$ and let $P_I := \{x : x \in P, x_j \in \{0,1\}, j = 1, \dots, p\}$ denote the feasible set of $(\mathcal{P})$. Hence, $\tilde{A}$ is a $q \times n$ matrix, where $q = m + n + p$. Let $Q := \{1, \dots, q\}$.

Furthermore, let $\bar{x}$ be a basic optimal solution of the LP relaxation of $(\mathcal{P})$, i.e., $\bar{x} = \arg\min\{cx : \tilde{A}x \geq \tilde{b}\}$. If $\bar{x}$ is not feasible for $(\mathcal{P})$, we can define a disjunction $\cup_{t \in T} D^t x \geq d_0^t$ that contains the feasible set of $(\mathcal{P})$ and not $\bar{x}$. For example, if there is a nonempty set $K \subseteq \{1, \dots, p\}$ for which $0 < \bar{x}_k < 1$ for every $k \in K$, then we can define a disjunction of the form

$$\bigcup_{K' \subseteq K} \begin{pmatrix} x_k \geq 1, & k \in K' \\ x_k \leq 0, & k \in K \setminus K' \end{pmatrix}, \tag{3.1}$$

which we denote as a *simple* t-branch split disjunction[1], where $t = |K|$. In its general form $\cup_{t \in T} D^t x \geq d_0^t$, we can parameterize a family of polyhedra with P_I-free interior $S\left(\{v^t\}_{t \in T}\right) := \{x : (v^t D^t) \leq v^t d_0^t\}$, where $v^t \geq 0$ and $v^t \neq 0$, and for each such polyhedron containing $\bar{x}$ as an interior point it is possible to derive an intersection cut separating $\bar{x}$.

[1] Unless noted otherwise, we will use t to index terms of a DP in general form instead of t-branch split disjunctions.

We can generate lift-and-project cuts by intersecting sets such as (3.1) with the linear relaxation of $(\mathcal{P})$. More generally, we have a DP of the form

$$\bigcup_{t \in T} \left(\begin{array}{c} \tilde{A}x \geq \tilde{b} \\ D^t x \geq d_0^t \end{array} \right). \tag{3.2}$$

The classic formulation for the Cut Generating Linear Program (CGLP) used to find a lift-and-project cut from (3.2) separating $\bar{x}$, which excludes some dominated inequalities, is as follows:

$$\begin{array}{llll} \min \ \alpha\bar{x} - \beta & & & \qquad (3.3) \\ \alpha & -u^t\tilde{A} - v^t D^t & = 0, & t \in T \qquad (3.4) \\ \beta & -u^t\tilde{b} - v^t d_0^T & = 0, & t \in T \qquad (3.5) \\ & \displaystyle\sum_{t \in T} u^t e + \sum_{t \in T} v^t e & = 1 & \qquad (3.6) \\ & u^t, v^t & \geq 0, & t \in T \qquad (3.7) \end{array}$$

If we assume, without loss of generality, that the upper bounds on the binary variables are contained in $Ax \geq b$, then we extend x with surplus variables of the form $x_{n+i} = \sum_{j=1}^{n} a_{ij}x_j - b_i$ for $i = 1, \ldots, m$ and define the following LP cone $C(J)$ from each basic solution $x(J)$ of P:

$$x_i = \bar{a}_{i0} - \sum_{j \in J} \bar{a}_{ij}x_j, \qquad i \in I \tag{3.8}$$

$$x_i \geq 0, \qquad i \in J \tag{3.9}$$

where I is the index set of basic variables and J of the nonbasic variables. Cone $C(J)$ has an extreme ray $r^j(J)$ corresponding to each nonbasic variable $x_j, j \in J$, with $r^j_j(J) = 1$, $r^j_i(J) = -\bar{a}_{ij}$ for $i \in I$, and $r^j_i(J) = 0$ for $i \in J \setminus \{j\}$. If there is a convex set S containing $x(J)$ but no feasible point in its interior, which we denote as P_I-free, we can define the intersection cut [Balas, 1971]

$$\sum_{j \in J} \frac{1}{\lambda_j^*} x_j \geq 1 \tag{3.10}$$

separating $x(J)$ from P_I, where λ_j^* parameterizes the point at which each ray $x(J) + \lambda_j r_j(J)$ intersects with the boundary of S. If some ray never intersects the boundary, then the corresponding coefficient of the intersection cut is zero instead of λ_j^* inversed.

3.3 Regularity of CGLP Solutions

Let $\bar{w} = \left(\bar{\alpha}, \bar{\beta}, \{\bar{u}^t, \bar{v}^t\}_{t\in T}\right)$ be a basic feasible CGLP solution. The inequality $\bar{\alpha}x \geq \bar{\beta}$ cuts off part of P only if $\bar{v}^t e > 0$ for each $t \in T$, since otherwise $\bar{\alpha}x \geq \bar{\beta}$ is implied by $\tilde{A}x \geq \tilde{b}$, as shown in Lemma 1 of Balas and Perregaard [2003].

From Theorem 10 of Balas and Kis [2016], a CGLP solution $\bar{w}$ such that $\bar{v}^t e > 0$ is regular if $\tilde{A}$ has an $n \times n$ nonsingular submatrix $\tilde{A}_J$ such that $\bar{u}^t_j = 0$ for all $j \notin J, t \in T$, in which case $\bar{\alpha}x \geq \bar{\beta}$ is equivalent to the intersection cut from the LP cone associated with the cobasis indexed by J and the P_I-free convex set defined by $\{x : (\bar{v}^t D^t)x \leq \bar{v}^t d^t_0, t \in T\}$. More specifically, a positive multiplier for some row of $Ax \geq b$ maps the corresponding surplus variable as nonbasic, a positive multiplier for a bound $x_j \geq 0$ maps x_j as nonbasic at the lower bound, and a positive multiplier for the row corresponding to bound $x_j \leq 1$ maps x_j as nonbasic at the upper bound. From Theorem 12 of Balas and Kis [2016], the sufficient condition for the regularity of $\bar{w}$ given in Theorem 10 is also necessary, i.e., if the condition is not met, then $\bar{w}$ is irregular.

The next Theorem gives a simple criterion for deciding whether a basic CGLP solution $\bar{w}$ is regular or not.

Theorem 3.1. *For a basic CGLP solution $\bar{w} = (\bar{\alpha}, \bar{\beta}, \{\bar{u}^t, \bar{v}^t\}_{t\in T})$, let $\tilde{A}_N$ be the $|N| \times n$ submatrix of $\tilde{A}$ whose rows are indexed by $N(\bar{u}) := \{j \in Q : \bar{u}^t_j > 0$ for some $t \in T\}$. Then $\bar{w}$ is a regular solution if, and only if, $\tilde{A}_N$ is of full row rank.*

Proof. Sufficiency. Assume $\text{rank}(\tilde{A}_N) = |N|$. Then $|N| \leq n$. We show that in this situation $\bar{w}$ is regular.

Case 1: $|N| = n$. Then $\tilde{A}_N$ is an $n \times n$ nonsigular submatrix of $\tilde{A}$ such that $u^t_j = 0$ for all $j \notin N$ and all $t \in T$, i.e., $\bar{w}$ satisfies the condition of Theorem 10 of Balas and Kis [2016].

Case 2: $|N| < n$. Then $\tilde{A}$ has $n - |N|$ rows $\tilde{A}_j$ with $u^t_j = 0$ which can be added to $\tilde{A}_N$ in order to form an $n \times n$ nonsingular matrix $\tilde{A}_{N'}$ since $\tilde{A}$ contains I_n. Substituting $\tilde{A}_{N'}$ for $\tilde{A}_N$ then reduces this case to Case 1.

Necessity. Assume $\text{rank}(\tilde{A}_N) < |N|$. We show that in this case $\bar{w}$ is irregular. In particular, any $n \times n$ nonsingular submatrix $\tilde{A}_J$ of $\tilde{A}$ has among its rows at most $\text{rank}(\tilde{A}_N)$ rows of $\tilde{A}_N$, thus leaving $|N| - \text{rank}(\tilde{A}_N)$ rows j such that $\bar{u}^t_j > 0$ for some $t \in T$ outside of $\tilde{A}_J$. Therefore no such $\tilde{A}_J$ meets the condition of Theorem 10 of Balas and Kis [2016], hence $\bar{w}$ is irregular. □

Corollary 3.2. *If a CGLP solution $\bar{w} = (\bar{\alpha}, \bar{\beta}, \left\{\bar{u}^t, \bar{v}^t\right\}_{t\in T})$ is not basic but satisfies the other conditions of Theorem 3.1 for regularity, then the cut is valid for the closure of regular cuts.*

Proof. If $\bar{w}$ is not basic, then we can describe it as a proper convex combination of a set of basic CGLP solutions. Let these solutions be indexed by a given set $\mathcal{B}$, so that $(\bar{\alpha}, \bar{\beta}, \left\{\bar{u}^t, \bar{v}^t\right\}_{t\in T}) = \sum_{b\in\mathcal{B}} \lambda_b \left(\tilde{\alpha}^b, \tilde{\beta}^b, \left\{\tilde{u}^{t^b}, \bar{v}^{t^b}\right\}_{t\in T}\right)$, $\lambda > 0$, and $\sum_{b\in\mathcal{B}} \lambda_b = 1$.

Note that $\tilde{u}_j^{t^b} > 0$ implies that $\bar{u}_j^t > 0$, and thus $N(\tilde{u}^b) \subseteq N(\bar{u})$ for all $b \in \mathcal{B}$. Since the submatrix of $\tilde{A}$ on the rows of $N(\bar{u})$ is of full row rank, then there exists a set N such that $|N| = n$ and $N(\bar{u}) \subseteq N$ for which $\text{rank}(\tilde{A}_N) = n$, and thus $N(\tilde{u}^b) \subseteq N$ for all $b \in \mathcal{B}$. That implies that all cuts of the form $\tilde{\alpha}^b x \geq \tilde{\beta}^b$ for each $b \in \mathcal{B}$ define intersection cuts from a same basis. □

Thus deciding whether a basic CGLP solution $\bar{w}$ is regular or not is straightforward. However, if $\bar{w}$ is irregular, then according to Theorem 12 of Balas and Kis [2016] the cut $\bar{\alpha}x \geq \bar{\beta}$ is irregular only if there exists no regular basic feasible solution $\tilde{w} = (\tilde{\alpha}, \tilde{\beta}, \{\tilde{u}^t, \tilde{v}^t\}_{t \in T})$ of the CGLP with $(\tilde{a}, \tilde{b}) = \theta(\bar{\alpha}, \bar{\beta})$ for some $\theta > 0$. With Corollary 3.2, any regular CGLP solution suffices to prove that a given cut is regular. Next we examine how to use that for determining cut regularity.

3.4 Regularity of Cuts from CGLP Solutions

Given an irregular CGLP solution $\bar{w} = (\bar{\alpha}, \bar{\beta}, \{\bar{u}^t, \bar{v}^t\}_{t \in T})$, we define a mixed-integer program based on the CGLP to establish whether there is a regular CGLP solution $\tilde{w} = (\tilde{\alpha}, \tilde{\beta}, \{\tilde{u}^t, \tilde{v}^t\}_{t \in T})$ such that $(\tilde{\alpha}, \tilde{\beta}) = \theta(\bar{\alpha}, \bar{\beta})$ for some $\theta > 0$. In comparison to the CGLP formulation, we add a variable $\theta \in [0, 1]$ and restrict the value of (α, β) to $\theta(\bar{\alpha}, \bar{\beta})$. Furthermore, we remove the normalization constraint (3.6) and introduce a binary upper bounding variable δ_j for each u_j in order to model the set N of indices $j \in Q$ such that $u_j^t > 0$ for some $t \in T$. We embed Theorem 1 by restricting the size of such set to at most n and requiring that the submatrix $\tilde{A}_N$ to be of full row rank. We denote the resulting formulation as the Regular Cut Verifier MILP (RCV-MILP):

$$
\begin{array}{llllll}
\max & \theta & & & & (3.11)\\
& \theta\bar{\alpha} & -u^t\tilde{A} - v^t D^t & = 0, & t \in T & (3.12)\\
& \theta\bar{\beta} & -u^t\tilde{b} - v^t d_0^t & = 0, & t \in T & (3.13)\\
& \delta_j & -u_j^t & \geq 0, & j \in Q,\, t \in T & (3.14)\\
& \displaystyle\sum_{j \in Q} \delta_j & & \leq n & & (3.15)\\
& \displaystyle\sum_{j \in N} \delta_j & & \leq \text{rank}(\tilde{A}_N), & N \subseteq Q & (3.16)\\
& & u^t, v^t & \geq 0, & t \in T & (3.17)\\
& & \delta_j & \in \{0,1\}, & j \in Q & (3.18)\\
& & \theta & \in [0,1] & & (3.19)
\end{array}
$$

Some comments regarding RCV-MILP are in order. First, constraints (3.14) and (3.18) define an implicit normalization constraint $\|u\|_\infty \leq 1$, which may prevent us from finding the cut in the same scale. Hence, variable θ is necessary even though normalization (3.6) is removed in comparison to the CGLP. Second, the equivalence between a RCV-MILP solution $(\breve{\theta}, \breve{\delta}, \{\breve{u}^t, \breve{v}^t\}_{t\in T})$ and a regular CGLP solution denoting a cut equivalent to $\bar{\alpha}x \geq \bar{\beta}$ is not direct. Instead of simply stating a corresponding CGLP solution $(\breve{\theta}\bar{\alpha}, \breve{\theta}\bar{\beta}, \{\breve{u}^t, \breve{v}^t\}_{t\in T})$, we may need to first scale the RCV-MILP solution if $\sum\limits_{t\in T} \breve{u}^t e + \sum\limits_{t\in T} \breve{v}^t e \neq 1$ in order to satisfy normalization (3.6). Such scaling is done a number of times in the next proof. Last, the number of subsets of Q, and consequently the number of rows due to constraint (3.16), can be very large. In Section 3.5, we address that by iteratively adding only the relevant subsets of Q to the formulation.

The next result proves the validity of RCV-MILP. In what follows, we keep denoting the set of rows with positive multipliers as $N(\bar{u})$. Furthermore, let $\delta(\bar{u})$ be a vector in which $\delta_i = 1$ if $\bar{u}_i^t > 0$ for some $t \in T$ and $\delta_i = 0$ otherwise.

Theorem 3.3. *Let $\bar{w} = (\bar{\alpha}, \bar{\beta}, \{\bar{u}'\bar{v}^t\}_{t\in T})$ be a basic optimal solution of CGLP and let $(\breve{\theta}, \breve{\delta}, \{\breve{u}^t, \breve{v}^t\}_{t\in T})$ be an optimal solution of RCV-MILP for cut $\bar{\alpha}x \geq \bar{\beta}$. Then the cut is regular if, and only if, $\breve{\theta} > 0$. Furthermore, if $\bar{\alpha}x \geq \bar{\beta}$ is regular then there is a cobasis indexed by J, $J \supseteq N(\bar{u})$, such that the cut is equivalent to the intersection cut from such basis and the P_I-free polyhedron $S(\bar{v})$.*

Proof. If $\bar{w}$ is a regular CGLP solution, then $(1, \delta(\bar{u}), \{\bar{u}^t, \bar{v}^t\}_{t\in T})$ is an optimal solution of RCV-MILP. First, $\theta = 1$ implies that constraints (3.4) and (3.5) are equivalent to (3.12) and (3.13), whereas constraints (3.7) and (3.17) are the same. Second, normalization (3.6) implies that $\bar{u} \leq 1$ and thus $\delta(\bar{u})$ satisfies constraints (3.14) and (3.18). Last, constraints (3.15) and (3.16) are satisfied since $\bar{w}$ is regular.

For the rest of the proof, we assume that the CGLP solution $\bar{w}$ is irregular.

Suppose that the cut is regular, and thus there is a regular CGLP solution $\tilde{w} = (\tilde{\alpha}, \tilde{\beta}, \{\tilde{u}^t, \tilde{v}^t\}_{t\in T})$ for which $(\bar{\alpha}, \bar{\beta}) = \lambda(\tilde{\alpha}, \tilde{\beta})$ for some $\lambda > 0$. If $\lambda \leq 1$, then $(\lambda, \delta(\tilde{u}), \{\tilde{u}^t, \tilde{v}^t\}_{t\in T})$ is feasible for RCV-MILP and thus $\breve{\theta} \geq \lambda > 0$. If $\lambda > 1$, we can divide the same values by λ to obtain another feasible CGLP solution $\hat{w} = \left(\frac{1}{\lambda}\tilde{\alpha}, \frac{1}{\lambda}\tilde{\beta}, \left\{\frac{1}{\lambda}\tilde{u}^t, \frac{1}{\lambda}\tilde{v}^t\right\}_{t\in T}\right) = \left(\bar{\alpha}, \bar{\beta}, \left\{\frac{1}{\lambda}\tilde{u}^t, \frac{1}{\lambda}\tilde{v}^t\right\}_{t\in T}\right)$. The construction of a RCV-MILP solution when $\lambda \leq 1$ applies to $\hat{w}$, thereby implying that $\left(1, \delta(\tilde{u}), \left\{\frac{1}{\lambda}\tilde{u}^t, \frac{1}{\lambda}\tilde{v}^t\right\}_{t\in T}\right)$ is optimal for RCV-MILP and thus $\breve{\theta} = 1$.

Finally, suppose that the cut is irregular, and thus there is no regular CGLP solution corresponding to cut $\bar{\alpha}x \geq \bar{\beta}$. Let us suppose, for contradiction, that there is a RCV-MILP solution $(\breve{\theta}, \breve{\delta}, \{\breve{u}^t, \breve{v}^t\}_{t\in T})$ with $\breve{\theta} > 0$. Let $\sigma := \sum\limits_{t\in T} \breve{u}^t e + \sum\limits_{t\in T} \breve{v}^t e$. Since the cut separates $\bar{x}$ and $\breve{\theta} > 0$, then $(\bar{\alpha}, \bar{\beta}) \neq 0$ and thus $\sigma > 0$. Hence, if we divide the multipliers $\{\breve{u}^t, \breve{v}^t\}_{t\in T}$ by σ, then normalization (3.6) is satisfied and $\left(\frac{\breve{\theta}}{\sigma}\bar{\alpha}, \frac{\breve{\theta}}{\sigma}\bar{\beta}, \left\{\frac{1}{\sigma}\breve{u}^t, \frac{1}{\sigma}\breve{v}^t\right\}_{t\in T}\right)$ is a feasible regular CGLP solution: a contradiction. $\square$

3.5 Numerical Procedure

Next we describe a numerical procedure to identify irregular lift-and-project cuts, which addresses two issues with using RCV-MILP directly.

First, finite numerical precision may lead to rounding errors and cause false negatives, thus overcounting the number of irregular cuts. We address that by adding a relative tolerance parameter ε on the coefficients of the cut. For example, if we want to determine if a cut $2x_1 + 0.3x_2 \geq 10$ is regular for $\varepsilon = 0.0001$, then we look for valid inequalities on each term where $\alpha_1 \in \theta[1.9998, 2.0002]$, $\alpha_2 \in \theta[0.29997, 0.30003]$, and $\beta \in \theta[9.999, 10.001]$. Thus we avoid type II errors at the price of tolerating type I errors. This choice is intentional, since we are mainly interested in knowing which lift-and-project cuts are not regular. Furthermore, if ε is sufficiently small, misclassifications are very unlikely.

Second, the number of subsets of Q can be very large and many of those subsets might be irrelevant. We can address that by defining a set $\mathcal{Q}$ of subsets of Q and then adding elements to this set as needed.

Hence, we define the Iterative RCV-MILP (IRCV-MILP):

$$\begin{aligned}
\max \theta & & \\
& -\theta\varepsilon \leq \theta\bar{\alpha} - u^t\tilde{A} - v^tD^t \leq \theta\varepsilon, & t \in T \\
& -\theta\varepsilon \leq \theta\bar{\beta} - u^t\tilde{b} - v^td_0^t \leq \theta\varepsilon, & t \in T \\
& \delta_j - u_j^t \geq 0, & j \in Q, t \in T \\
& \sum_{j \in Q} \delta_j \leq n & \\
& \sum_{j \in N} \delta_j \leq \operatorname{rank}(\tilde{A}_N), & N \subseteq \mathcal{Q} \\
& u^t, v^t \geq 0, & t \in T \\
& \delta_j \in \{0, 1\}, & j \in Q \\
& \theta \in [0, 1] &
\end{aligned}$$

Algorithm 9 uses IRCV-MILP to determine if a cut is regular, subject to false positives only. It finishes in finite time since each repetition of the loop corresponds to adding a different subset N to $\mathcal{Q}$. In the unlikely worst case, Algorithm 9 terminates when all subsets of Q have been added to $\mathcal{Q}$.

We can reduce the number of loop repetitions by preventing combinations of inequalities corresponding to parallel hyperplanes across different terms of the disjunction. For example, if rows j_1 and j_2 correspond to $x_i \geq 0$ and $x_i \leq 1$, then we can add the following inequality to IRCV-CGLP:

Algorithm 9 Checks if there is a regular CGLP solution for a given cut

```
function IsCutRegular($\bar{\alpha}, \bar{\beta}, \{\bar{u}^t, \bar{v}^t\}_{t \in T},\ \varepsilon$)
    $N \leftarrow N(\bar{u})$
    if rank($\tilde{A}_N$) = $|N|$ then
        return True                                          ▷ Original CGLP solution is regular
    else
        $\mathcal{Q} \leftarrow \emptyset$
        loop
            Get optimal solution ($\check{\theta}, \check{\delta}, \{\check{u}^t, \check{v}^t\}_{t \in T}$) of IRCV-MILP
            $N \leftarrow N(\check{u})$
            if $\bar{\theta} = 0$ then
                return False                                 ▷ There is no regular CGLP solution
            else if rank($\tilde{A}_N$) < $|N|$ then
                $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{N\}$                      ▷ Loop has to be repeated
            else
                return True                                  ▷ Found regular CGLP solution
            end if
        end loop
    end if
end function
```

$$\delta_{j_1} + \delta_{j_2} \leq 1 \tag{3.20}$$

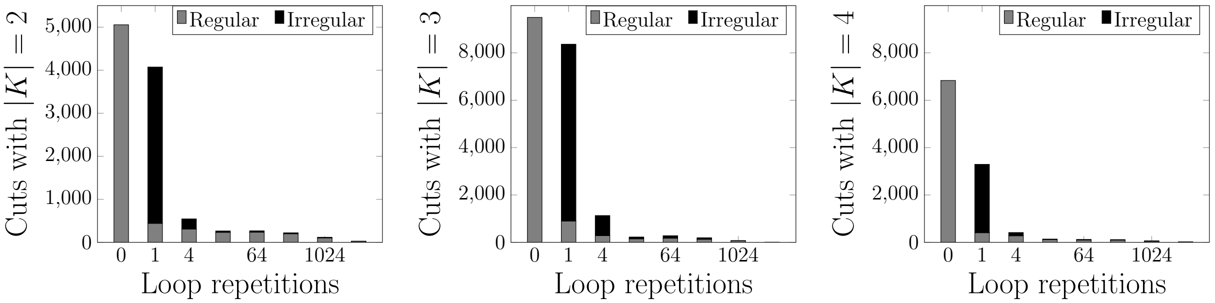
3.6 Computational Experiments

We have run experiments to find irregular lift-and-project cuts among the first round of cuts generated for 74 instances from the MIPLIB 2, 3, and 2003 benchmarks [Bixby et al., 1992, 1998, Achterberg et al., 2006]. For each of those instances, we found an optimal solution $\bar{x}$ of the LP relaxation and generated a cut using the CGLP from each disjunction of the form (3.1) with $|K| = 2$ for all instances as well as $|K| = 3$ and $|K| = 4$ for smaller ones. Since the verification can be computationally expensive, the experiments were restricted to instances with at most 10,000 nonzeroes and each verification was interrupted if inconclusive after a predefined number of steps or time, which are both detailed later. All code is written in C++ (gcc 4.8.2) and ran in Ubuntu 14.04.1 LTS on a machine with 48 Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz processors and 32 GB of RAM. No more than two copies were run in parallel and each was restricted to 10 GB of RAM. The formulations were solved with CPLEX version 12.6.3 and matrix ranks were computed with Eigen[2]. Finally, we have run Algorithm 9 to determine if the cut is irregular with $\varepsilon = 0.0001$ and constraints of the form (3.20) for the bounds of each variable $x_i,\ i = 1, \ldots, p$.

The most extensive experiments were run on 45 instances, where we counted the number of times that the loop of Algorithm 9 is repeated and set a time limit of 1 hour to interrupt the verification for each cut. We have chosen the subset of instances with at most 2,000 nonzeroes, 150

[2] Available at `http://eigen.tuxfamily.org`

Figure 3.1: Incidence of cuts assessed by Algorithm 9 by the order of repetitions of the loop upon termination.



integer variables, or 50 rows; and we have also included larger instances from families of instances that nearly made the cut, namely `p0291`, `misc03`, `misc07`, and `pp08a`. For instances with at most 20 fractional values in $\bar{x}$ for integer variables, we generated and tested cuts with $|K| = 2$ to $|K| = 4$. For instances with at most 30 fractional values, we generated and tested cuts with $|K| = 2$ and $|K| = 3$. For other instances with at most 50 fractional values and `pp08a`, we only generated and tested cuts with $|K| = 2$. Some instances were discarded or only the results for less disjunctions were reported in cases where too many verifications timed out or the program run out of resources.

The results for 36 instances having cuts with $|K| = 2$ and $|K| = 3$ are found in Table 3.1. Additional results for the 22 instances having cuts with $|K| = 4$ are found in Table 3.2. We also compare the incidence of irregular CGLP solutions and cuts with $|K| = 2$ to $|K| = 4$ for those 22 instances in Table 3.3. The results for the remaining 9 instances having cuts with $|K| = 2$ are found in Table 3.4. Since the number of cuts for each instance varies, we summarize a per-instance average of the percentages for each metric, which weights the results of each instance for equal contribution.

Figure 3.1 shows the number of cuts identified as regular and irregular by Algorithm 9 according to the order of magnitude of repetitions of the loop upon termination. Figure 3.2 compares the regularity of cuts from each K of size 3 and 4 with that of cuts from subsets of K of size 2 among the 97% of the cases where no cut for a subset timed out. Figure 3.3 compares the average gap closed and the average Euclidean distance of the cuts generated with respect to the fractional solution $\bar{x}$ on all disjunction sizes where both types of cuts are observed on each of the 45 instances.

For the remaining 29 instances reported in Table 3.5, we repeat the loop in Algorithm 9 once. This set includes some instances previously excluded.

Table 3.1: Results for instances where lift-and-project cuts with |K|=2 and |K|=3 are generated tested. For each instance and size of K, we report the regularity of the CGLP solution and the regularity of the cut by running Algorithm 9 with time limit of one hour.

		Lift-and-project cuts with $\lvert K\rvert = 2$					Lift-and-project cuts with $\lvert K\rvert = 3$				
	Frac.	CGLP basis		Cut			CGLP basis		Cut		
Inst.	Vars.	Regular	Irregular	Regular	Irregular	Unknown	Regular	Irregular	Regular	Irregular	Unknown
air01	5	4	6	4	6	0	1	9	1	5	4
bell5	25	240	60	280	9	11	1787	513	2090	63	147
blend2	6	4	11	9	5	1	8	12	15	4	1
bm23	6	0	15	0	15	0	0	20	0	20	0
enigma	8	4	24	25	0	3	4	52	45	0	11
flugpl	10	16	29	16	29	0	33	87	35	85	0
gt2	11	55	0	55	0	0	153	12	164	0	1
khb05250	19	10	161	11	160	0	0	969	6	957	6
lseu	11	43	12	51	4	0	92	73	150	15	0
markshare1	6	0	15	0	15	0	0	20	0	20	0
markshare2	7	10	11	10	11	0	9	26	10	25	0
mas74	12	65	1	66	0	0	203	17	220	0	0
mas76	11	55	0	55	0	0	165	0	165	0	0
misc01	12	64	2	64	0	2	166	54	175	13	32
misc02	8	26	2	26	2	0	32	24	32	14	10
misc03	14	87	4	87	1	3	334	30	334	0	30
misc05	11	17	38	21	22	12	18	147	27	64	74
misc07	16	115	5	115	0	5	502	58	502	0	58
mod008	5	0	10	4	6	0	0	10	3	7	0
mod013	5	4	6	6	4	0	2	8	6	3	1
modglob	30	2	433	32	403	0	0	4060	75	3983	2
p0033	6	13	2	15	0	0	19	1	20	0	0
p0040	4	5	1	6	0	0	3	1	4	0	0
p0201	20	189	1	190	0	0	1124	16	1140	0	0
p0282	26	241	84	255	48	22	1585	1015	1703	471	426
p0291	10	30	15	34	7	4	70	50	76	26	18
pipex	6	3	12	8	7	0	0	20	8	7	5
pk1	15	1	104	9	96	0	0	455	4	451	0
rgn	14	40	51	65	25	1	76	288	190	152	22
sample2	12	18	48	21	44	1	28	192	31	175	14
sentoy	8	1	27	11	17	0	0	56	28	28	0
stein9	6	9	6	13	2	0	2	18	12	8	0
stein15	12	9	57	13	53	0	0	220	0	220	0
stein27	21	42	168	45	164	1	8	1322	14	1316	0
vpm1	15	100	5	102	3	0	441	14	450	5	0
vpm2	30	314	121	390	38	7	2632	1428	3441	461	158
Inst. average		51.3%	48.7%	62.9%	35.0%	2.0%	40.8%	59.2%	54.5%	38.5%	7.0%

Table 3.2: Additional results for instances where lift-and-project cuts with |K|=4 are also generated and tested.

		Lift-and-project cuts with $|K| = 4$				
	Frac.	CGLP basis		Cut		
Inst.	Vars.	Regular	Irregular	Regular	Irregular	Unknown
blend2	6	4	11	6	9	0
bm23	6	0	15	0	15	0
enigma	8	4	66	68	1	1
flugpl	10	52	158	71	139	0
gt2	11	179	151	238	1	91
markshare1	6	0	15	0	15	0
markshare2	7	1	34	1	34	0
misc03	14	850	151	856	0	145
mod008	5	0	5	1	4	0
mod013	5	2	3	3	2	0
p0033	6	15	0	15	0	0
p0040	4	0	1	1	0	0
p0201	20	4191	654	4753	0	92
p0291	10	106	104	109	34	67
pipex	6	0	15	3	5	7
pk1	15	0	1365	0	1365	0
rgn	14	74	927	317	539	145
sample2	12	17	478	21	430	44
sentoy	8	0	70	31	39	0
stein9	6	0	15	12	3	0
stein15	12	0	495	0	495	0
vpm1	15	1339	26	1347	17	1
Inst. average		26.6%	73.4%	47.3%	46%	6.7%

Figure 3.2: Incidence of regular and irregular cuts from disjunctions with |K| = 3 and |K| = 4 according to the incidence of irregular cuts across all disjunctions from 2-subsets of K, i.e., all K' ⊂ K such that |K'|=2.

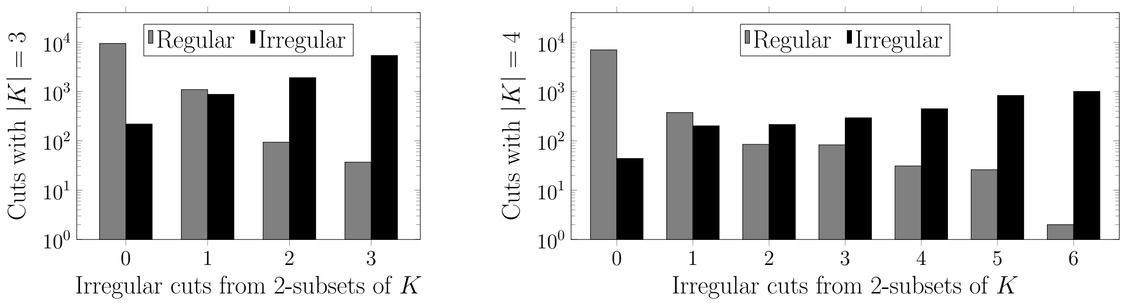


Table 3.3: Frequency of irregular bases and cuts with $|K| = 2$, 3, and 4.

	$\|K\| = 2$		$\|K\| = 3$		$\|K\| = 4$	
Inst.	Basis	Cut	Basis	Cut	Basis	Cut
blend2	73.3%	33.3%	60%	20%	73.3%	60%
bm23	100%	100%	100%	100%	100%	100%
enigma	85.7%	0%	92.9%	0%	94.3%	1.4%
flugpl	64.4%	64.4%	72.5%	70.8%	75.2%	66.2%
gt2	0%	0%	7.3%	0%	45.8%	0.3%
markshare1	100%	100%	100%	100%	100%	100%
markshare2	52.4%	52.4%	74.3%	71.4%	97.1%	97.1%
misc03	4.4%	1.1%	8.2%	0%	15.1%	0%
mod008	100%	60%	100%	70%	100%	80%
mod013	60%	40%	80%	30%	60%	40%
p0033	13.3%	0%	5%	0%	0%	0%
p0040	16.7%	0%	25%	0%	100%	0%
p0201	0.5%	0%	1.4%	0%	13.5%	0%
p0291	33.3%	15.6%	41.7%	21.7%	49.5%	16.2%
pipex	80%	46.7%	100%	35%	100%	33.3%
pk1	99%	91.4%	100%	99.1%	100%	100%
rgn	56%	27.5%	79.1%	41.8%	92.6%	53.8%
sample2	72.7%	66.7%	87.3%	79.5%	96.6%	86.9%
sentoy	96.4%	60.7%	100%	50%	100%	55.7%
stein9	40%	13.3%	90%	40%	100%	20%
stein15	86.4%	80.3%	100%	100%	100%	100%
vpm1	4.8%	2.9%	3.1%	1.1%	1.9%	1.2%
Inst. average	56.3%	38.9%	64.9%	42.3%	73.4%	46%

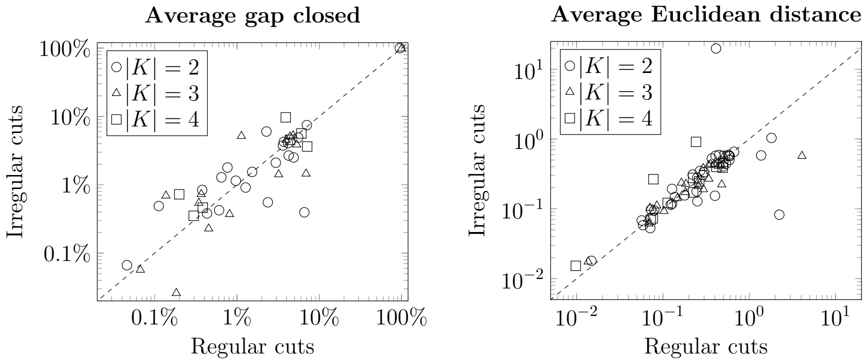
Table 3.4: Results for instances where only lift-and-project cuts with $|K| = 2$ are generated and tested.

		Lift-and-project cuts with $\|K\| = 2$				
	Frac.	CGLP basis		Cut		
Inst.	Vars.	Regular	Irregular	Regular	Irregular	Unknown
bell3a	32	404	92	458	12	26
bell3b	36	442	188	568	33	29
bell4	46	887	148	966	18	51
dcmulti	49	503	673	526	631	19
egout	40	16	764	435	320	25
noswot	22	189	42	199	0	32
p0548	48	313	815	648	138	342
pp08a	53	35	1343	35	1343	0
pp08aCUTS	46	430	605	636	295	104
Inst. average		48.4%	51.6%	64.9%	26.9%	8.2%

Table 3.5: Remaining results for larger instances where lift-and-project cuts with $|K| = 2$ are generated and tested, but the loop of Algorithm 9 is only repeated once.

		Lift-and-project cuts with $|K| = 2$				
		CGLP basis		Cut		
Inst.	Vars.	Regular	Irregular	Regular	Irregular	Unknown
aflow30a	31	155	310	157	296	12
aflow40b	38	347	356	349	240	114
cracpb1	40	389	391	408	259	113
dsbmip	48	464	664	871	36	221
fiber	42	687	174	713	42	106
fixnet3	69	198	2148	228	0	2118
fixnet4	67	180	2031	196	0	2015
fixnet6	60	139	1631	153	0	1617
gen	41	505	315	511	58	251
gesa2	58	704	949	749	7	897
gesa2_o	73	1356	1272	1383	763	482
gesa3	85	1785	1785	1820	589	1161
gesa3_o	100	2175	2775	2266	1555	1129
glass4	72	2556	0	2556	0	0
harp2	30	435	0	435	0	0
l152lav	51	254	1021	292	904	79
lp4l	23	50	203	73	121	59
opt1217	27	74	277	88	106	157
p2756	77	1521	1405	2190	282	454
qiu	36	474	156	474	16	140
qnet1	47	578	503	645	12	424
qnet1_o	11	20	35	32	0	23
rout	35	304	291	308	220	67
set1al	218	239	23414	276	20940	2437
set1ch	138	122	9331	150	8786	517
set1cl	220	180	23910	225	21393	2472
timtab1	134	2829	6082	2942	5935	34
timtab2	236	8929	18801	9172	18027	531
tr12-30	348	47	60331	605	59046	727
Inst. average		38.7%	61.3%	43.0%	31.8%	25.2%

Figure 3.3: Comparison of regular vs. irregular cuts on instances and sizes of K where both types are found: the average gap closed is larger in 22 cases for irregular cuts vs. 24 cases for regular cuts; and the average Euclidean distance of the cut is larger in 37 cases for irregular cuts vs. 34 for regular cuts.



3.7 Discussion

The experiments above evidence some trends on irregular CGLP bases and cuts, from which we can draw five observations about their incidence.

First, there is a substantial difference between the number of irregular CGLP bases and that of irregular cuts. In all tables, the per-instance averages differ in at least 10% with respect to the total number of cuts. Therefore, an irregular CGLP basis often does not guarantee that the corresponding cut is irregular.

Second, most regular and irregular cuts can nevertheless be identified with the first test applied on them. Such paradox is in part explained by the fact that irregular cuts are less frequent than regular cuts in our experiments. Figure 3.1 shows that most regular cuts are identified immediately because they come from regular CGLP solutions, whereas most irregular cuts are identified in the first repetition of the loop of Algorithm 9. The same is true for the larger problems reported in Table 3.5: with a few exceptions, the majority of the cuts is identified as regular or irregular even though the loop is never repeated. In other words, restricting the number of nonzero multipliers of the linear relaxation to n across all terms of the CGLP formulation is an effective way of determining if a cut is irregular. Conversely, most irregular cuts are those that can only be derived by using more than n rows of the linear relaxation. Hence, the linear dependence among such rows is a secondary factor for cut irregularity in practice.

Third, simple t-branch split disjunctions with more terms yield more irregular cuts and even more irregular CGLP bases. We can observe that first from the difference in per-instance averages of the corresponding columns for $|K| = 2$ and $|K| = 3$ in Table 3.1 and from $|K| = 2$ to $|K| = 4$ in Table 3.3. In addition, the frequency of irregular CGLP bases and cuts for each instance often increases with the size of K, the former more than the latter. For the 36 instances reported in Table 3.1, the frequency of irregular CGLP bases for $|K| = 3$ is higher in 29 instances and lower in 3 when compared to the case of $|K| = 2$, whereas the frequency of irregular cuts is higher in 17 and lower in 9. For the 22 instances reported in Table 3.3, there are proportionally more irregular CGLP bases for $|K| = 4$ in 15 instances and less in 2 instances when compared to $|K| = 2$, while the ratio of irregular cuts is higher in 12 instances and lower in 4 instances. Hence, a disjunction with more terms seems more likely to yield lift-and-project cuts that are not regular.

Fourth, some disjunctions are more likely to yield irregular cuts than others. For simple t-branch split disjunctions, we observe in Figure 3.2 that an irregular cut is often obtained from a disjuction where K augments the set K' of indices of a disjunction with less terms, for which an irregular cut is obtained. For example, a disjunction on variables x_A, x_B, and x_C yields an irregular cut with increasing probability as more irregular cuts are obtained among disjuctions on x_A and x_B, x_A and x_C, and x_B and x_C. The probability of an irregular cut is already close to 50% from $|K'| = 2$ to $|K| = 3$ and $|K| = 4$ if one disjunction yields an irregular cut. A simpler hypothesis that could be ventured is that the presence of certain variables in the disjunction make it more likely to yield irregular cuts, although it is yet to be determined what makes disjunctions on particular variables more prone to yield irregular cuts. Nevertheless, our results indicate that one could use information about cuts from disjunctions where $|K| = 2$ to augment and combine those sets of indices yielding irregular cuts in the hope of obtaining irregular cuts on disjunctions with more terms.

Finally, the frequency of irregular cuts depends on the structure of the problem, it can be higher in larger instances but lower when more inequalities are added. This comes from observing the results for the many families of instances in the experiments. For families with similar size, we observe that some have little irregularity in the results (`mas74` and `mas76`, all `misc` instances except `misc05`, `bell` instances, and `fixnet` instances), some have a moderate level (`markshare1` and `markshare2`, `mod008` and `mod013`, and `aflow30a` and `aflow40b`), and some are highly irregular (`set1` instances and `timtab` instances). Notably, instances in the latter sets are often larger. We also observe instances yielding more irregular cuts according to their size for `p` instances, `stein9` to `stein27`, and `vpm1` and `vpm2`. Curiously, however, we observe a reduction in the number of irregular cuts when additional valid inequalities are added to some problems. There are pairs of problems that only differ in that, for example from `pp08a` to `pp08aCUTS`, from `gesa2_o` to `gesa2`, and from `gesa3_o` to `gesa3`. In all thoses cases, the number of irregular cuts reduced. One exception is from `qnet1_o` to `qnet1`, but in this case the number of fractional variables is substantially smaller and that may have affected the results. We could hypothesize that these additional constraints extend the reach of regular cuts, hence making irregular cuts less relevant than before. Overall, these results indicate that cuts from disjunctions with more terms would yield more irregular cuts at the root node and that they are more likely to do so in larger problems.

On the other hand, there is no clear picture about the impact of regular vs. irregular cuts. When comparing the average gap closed or the average Euclidean distance to the fractional solution separated, there are about the same number of cases favoring either type of cut. From Figure 3.3, we can nevertheless note that the results for the average Euclidean distance fluctuate at a much smaller scale around the identity line, possibly indicating that outliers to either side could be a starting point to refine the analysis in future work.

3.8 Conclusion

We have used a mixed-integer formulation to determine the equivalence between lift-and-project cuts from arbitrary disjunctions and intersection cuts. This method is conveniently used to evaluate the extent to which t-branch split cuts differ from multi-row cuts, two types of cuts that have been intensely studied for the past decade. When there is no equivalence, the cut is said to be irregular and can only be obtained from irregular CGLP solutions. On the one hand, we have found that the incidence of irregular cuts varies across different families of instances and that many irregular CGLP solutions nevertheless correspond to regular cuts. On the other hand, the incidence of irregular cuts often increases as the problems get larger, the linear relaxation is weaker, and the disjunction has more terms. Furthermore, we have observed that 3-branch and 4-branch split disjunctions yield irregular cuts more often when they augment multiple 2-branch split disjunctions that also yield irregular cuts, a result that could be used to judiciously choose disjunctions with more terms from which to generate additional irregular cuts.

Chapter 4

The Integrated Last-Mile Transportation Problem

This chapter is based on the paper "The Integrated Last-Mile Transportation Problem (ILMTP)" [Raghunathan et al., 2018], which is in the proceedings of the upcoming ICAPS 2018 conference.

4.1 Introduction

Last-mile transportation (LMT) is defined as the service that delivers people from the hub of a mass transportation (MT) service to each passenger's final destination. The MT service can be one of air, boat, bus, or train. The LMT service can be facilitated by bike [Liu et al., 2012], car [Shaheen, 2004, Thien, 2013], autonomous pods [Shen et al., 2017], or personal rapid transit systems. Though the term LMT has also been used for the movement of goods in supply chains, home-delivery systems, and telecommunications, we will restrict our attention in this chapter exclusively to the transportation of people. A LMT service expands the access of MT to an area wider than that defined as "walking distance" of a transportation hub. Interest in the design and operation of LMT has grown tremendously in the past decade. This has been driven primarily by three factors [Wang, 2017]: (i) governmental push to reduce congestion and air pollution; (ii) increasing aging population in cities; and (iii) providing mobility for the differently abled and school children.

We consider the following typical scenario for the operation of the MT service in conjunction with the LMT. All passengers of the LMT service start their journey on the MT from one of the stations served by a train, and request automated transportation to the buildings within a time-window. The buildings B1-B10 can represent offices for different companies that are collocated in an industrial park, or residential buildings in a neighborhood. The buildings can be accessed by paths that are shared by both pedestrians and LMT vehicles. For convenience, we will refer to the vehicle providing LMT service as a *commuter vehicle* (CV). The CVs are typically parked at a terminal (T0) at which passengers arrive from trains and proceed to their respective destination buildings by sharing a ride in a CV. The LMT service may represent the morning commute to the office or the evening commute back to residences. Once all the passengers are delivered to their destinations, the CVs return back to the terminal for subsequent trips.

We envision an operational scenario in which the passengers indicate their origin station, destination building, and the desired time-window of arrival at the destination. This information is assumed to be available to the scheduler well in advance of scheduling decisions. For instance, consider the situation where the buildings represent offices and the passengers enter requests through a smartphone app on the evening of the previous day. Once all requests have been received, the scheduler determines for each user: (i) the train to board at their origin station; (ii) the CV to board at `T0`, (iii) the time the CV will depart from `T0`, and (iv) the time of arrival at the destination building. The scheduler also communicates to the different CVs the routes, start times, and list of passengers. The choice of route determines the times of arrival of passengers at their destination and the total time spent by the passengers in the CV.

Note that the first-mile operation, wherein the passengers first ride on CVs to reach a hub of a MT service, can be easily accommodated in an analogous manner. Furthermore, the requests need not be provided in advance and can instead be communicated to the scheduler in a *just-in-time* manner, but we do not address such online variants in this work.

In this work, we introduce the Integrated Last-Mile Transportation Problem (ILMTP). The ILMTP is defined as the problem of scheduling passengers jointly on the MT and LMT services so that the passengers reach their destinations within specified time-windows and the total transit time for all passengers is minimized. We assume that the information about passengers is available *in advance*. The transit time includes the time spent traveling in the transportation vehicles in addition to any time spent waiting for the LMT service. In determining the schedules on the LMT service, the problem also determines the set of groups that share a ride in a CV. Thus, the time spent by the passengers in the CV depends on the co-passengers. To the best of our knowledge, this general version has not been addressed in the literature.

We introduce a constructive heuristic and a local search method to group passengers for LMT trips, which based on this grouping are then optimally scheduled in a few seconds by an Integer Linear Programming (ILP) solver. These groups preserve the invariant that each group has a single destination and that the deadlines of the passengers across groups going to the same destination can be sorted in nondecreasing order. We show that there is always an optimal solution with such a structure if all passengers have time windows of the same size, all MT trips serve all stations with uniform trip times, and each LMT trip has a single destination. We present computational results restricted to solutions of this form and compare them with a general lower bound.

4.2 Related Work

The ILMTP can be broadly viewed as an instance of routing and scheduling with time-windows. We survey the relevant literature and describe the key differences with the problem here studied.

4.2.1 Last-Mile Transportation Problem

The literature on last-mile transportation has been mostly focused on the LMT service, without much consideration to the MT system. Seminal work in this area dates back to the 1960s and has focused mostly on freight transportation (see Wang (2017) for a discussion). For passenger transportation, a number of case studies have analyzed the last-mile problem in different contexts, such as a bicycle-sharing program in Beijing [Liu et al., 2012]. Wang (2017) is the first work to consider routing and scheduling in the LMT service. The paper also considered the minimization of total travel time and proposed a heuristic approach for constructing solutions. More recently, Mahéo et al. (2018) consider the design of a public transit system that includes multiple modes of transportation, however the authors did not considered scheduling aspects. The ILMTP is a strict generalization of Wang (2017), in that we consider time-windows for arrival and scheduling on the MT service. Furthermore, it also complements the work of Mahéo et al. (2018) by focusing exclusively on the scheduling aspects of multi-modal transportation.

4.2.2 Personal Rapid Transit

Personal Rapid Transit (PRT) has similarities to the last-mile problem and has attracted significant attention in the past decade. Research has been conducted on PRT system control frameworks [Anderson, 1998], financial assessments [Bly and Teychenne, 2005, Berger et al., 2011], performance approximations [Lees-Miller et al., 2009, 2010] and case studies [Mueller and Sgouridis, 2011]. However, none of these papers have addressed last-mile operational issues.

4.2.3 Demand Responsive Transit

A large body of research has been devoted to demand responsive transit (DRT), which is another type of on-demand service. Some papers focus on DRT concept discussions, practical implementation, and assessment of simulations in case studies [Brake et al., 2004, Horn, 2002a, Mageean and Nelson, 2003, Palmer et al., 2004, Quadrifoglio et al., 2008]. Models have been developed to assist in system design and regulation (e.g., Daganzo 1978, Diana et al. 2006, Wilson and Hendrickson 1980). Routing options in specific contexts have also been considered [Chevrier et al., 2012, Horn, 2002b]. The LMT can be viewed as a specific variant of a broadly defined DRT concept—namely, a demand responsive transportation system that addresses last-mile service requests with batch passenger demand and a shared passenger origin. Unlike most papers in the DRT literature, we also focus on routing and scheduling on the MT and LMT from an optimization and operational perspective.

4.2.4 Vehicle Routing & Dial-a-Ride Problems

Vehicle routing problems have long been studied, and they comprise a large body of literature. The vehicle routing problem with time windows (VRPTW) has been the subject of intensive study, with many heuristic and exact optimization approaches suggested in the literature. A thorough review of the VRPTW literature can be found in Toth and Vigo [2014]. The dial-a-ride problem (DARP) and related variations have also been extensively investigated [Cordeau and Laporte, 2007, Jaw et al., 1986, Lei et al., 2012]. As argued by Wang [2017], the VRPTW focuses on reducing operating costs while ILMTP aims to improve the level-of-service by minimizing total passenger transit time. The limited capacity of CVs may also comparatively facilitate approaching the ILMTP .

In summary, ILMTP has the following features that distinguishes it from previous studies in the literature: (1) joint scheduling of passengers on the MT and LMT services; (2) time-windows on arrival at destination; (3) common last-mile origin (which is also the vehicle depot); and (4) minimization of the total passenger transit time.

4.3 Problem Formulation

In this section we introduce notation relevant to ILMTP and present an optimization model. For ease of exposition, we will assume for the rest of the chapter that the MT corresponds to trains that arrive at T0. The problem data, summarized in Table 4.1, can be divided into those associated with: (i) the MT service, (ii) the LMT service, and (iii) the passengers. The set $\mathcal{U}(s)$ represents the set of all possible MT service options that are available for passengers from station s to the *terminal* T0. The MT services that are identical in travel time to reach T0 but leave at different times are treated as distinct MT service options in $\mathcal{U}$. The set $\mathcal{U}(s)$ includes the possibility of passengers transferring between trains in order to travel from s to T0. A user that leaves from s on a MT service $u \in \mathcal{U}(s)$ has a travel time of $(t^{\mathtt{T0}}(s) - t^{\text{start}}(u, s))$ on the MT service to reach T0. If a user does not leave on the CV until time t, then the user spends $(t - t^{\mathtt{T0}}(u, s))$ time waiting at T0. For commuting in the CVs, the travel time for the passengers depends on: (i) the route choice on the CV, and (ii) the number of stops on the route prior to their destination. Without loss of generality, we abstract loading and unloading times as part of the pre-computed routes, since routes with stops that are not actually used are suboptimal.

The decision variables in the formulation are:

- $y_{p,u} \in \{0, 1\}$: is 1 if passenger p is assigned to train service $u \in \mathcal{U}(s(p))$;
- $z_{p,r,t} \in \{0, 1\}$: is 1 if passenger p is assigned to CV route $r \in \mathcal{R}$ (with $b(p) \in \mathcal{B}(r)$) and departs at time $t \in \mathcal{T}$;
- $n_{\mathtt{T0},t}$: number of CVs at T0 at time $t \in \mathcal{T}$;

$\mathcal{T}$	planning horizon for the problem
$\mathcal{S}$	set of boarding stations for passengers
$\texttt{TO}$	the alighting station for passengers
$\mathcal{U}(s)$	the set of all MT services that serve s
$t^{\text{start}}(u, s)$	time that MT service $u \in \mathcal{U}(s)$ leaves s
$t^{\texttt{TO}}(u)$	time that MT service $u \in \mathcal{U}(s)$ reaches $\texttt{TO}$
$\mathcal{P}$	set of passengers
$s(p)$	origin station of $p \in \mathcal{P}$
$b(p)$	destination of $p \in \mathcal{P}$
$\texttt{rel}(p)$	earliest time of arrival for $p \in \mathcal{P}$
$\texttt{ded}(p)$	latest time of arrival for $p \in \mathcal{P}$
$\mathcal{C}$	set of commuter vehicles
$CV^{\max}$	capacity of CV
$\mathcal{R}$	the set of routes that the CV can take
$\mathcal{B}(r)$	the set of stops served on LMT route $r \in \mathcal{R}$
$\tau^{\text{travel}}(r, b)$	elapsed time on CV to reach $b \in \mathcal{B}(r)$ starting from $\texttt{TO}$ on route r
$\tau^{\texttt{trip}}(r)$	elapsed time for a CV to return to $\texttt{TO}$ on route r

Table 4.1: List of symbols for the ILMTP.

- $n_{r,t} \in \{1, \dots, |\mathcal{C}|\}$: number of CVs that are assigned to route r and depart at time $t \in \mathcal{T}$; and
- τ_p: transit time for passenger $p \in \mathcal{P}$.

We first describe the constraints in the problem that model operational requirements. The total transit time for each passenger $p \in \mathcal{P}$ is

$$\tau_p = \sum_{r \in \mathcal{R}} \sum_{t \in \mathcal{T}} (t + \tau^{\text{travel}}(r, b(p))) \cdot z_{p,r,t} - \sum_{u \in \mathcal{U}(s(p))} t^{\text{start}}(u) \cdot y_{p,u}. \tag{4.1a}$$

Next we model that each passenger $p \in \mathcal{P}$ is assigned to exactly one u on the MT service and has an unique CV route r and start time t:

$$\begin{aligned} \sum_{u \in \mathcal{U}(s(p))} y_{p,u} &= 1, \\ \sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}} z_{p,r,t} &= 1. \end{aligned} \tag{4.1b}$$

The following constraints ensure that each passenger starts on a CV trip after arriving to TO:

$$\sum_{t \le t^{\texttt{TO}}(u)} z_{p,r,t} \le 1 - y_{p,u} \, \forall \, p \in \mathcal{P}, u \in \mathcal{U}(s(p)). \tag{4.1c}$$

The time-windows on the arrivals are imposed using

$$\texttt{rel}(p) \leq \sum_{r\in\mathcal{R}}\sum_{t\in\mathcal{T}}(t+\tau^{\text{travel}}(r,b(p)))\cdot z_{p,r,t} \leq \texttt{ded}(p). \tag{4.1d}$$

The availability of CVs at TO for transporting passengers are modeled using time-difference equations in (4.1e). This formulation was presented by Wang (2017) and is frequently used in modeling cumulative scheduling problems.

$$\begin{aligned} n_{\texttt{TO},t} =& n_{\texttt{TO},t-1} + \sum_{r\in\mathcal{R}} n_{r,t-\tau^{\texttt{trip}}(r)} - \sum_{r\in\mathcal{R}} n_{r,t} \,\forall\, t\in\mathcal{T} \\ n_{\texttt{TO},0} =& |\mathcal{C}|. \end{aligned} \tag{4.1e}$$

Constraint (4.1e) states that the number of CVs in TO at t is the sum of three components: number of CVs in TO at time $t-1$; number of CVs returning to TO upon completion of trips started at time $t-\tau^{\texttt{trip}}(r)$ for each route r; and the negative of the number of CVs leaving TO on trips at time t.

Finally, the capacity of the CVs are modeled using

$$\left.\begin{aligned} \sum_{p\in\mathcal{P}} z_{p,r,t} \leq& CV^{\max}\cdot n_{r,t} \\ \sum_{p\in\mathcal{P}} z_{p,r,t} \geq& CV^{\max}\cdot (n_{r,t}-1) \end{aligned}\right\} \forall\, r\in\mathcal{R}, t\in\mathcal{T}. \tag{4.1f}$$

An optimization formulation modeling ILMTP is thus

$$\min \sum_{p\in\mathcal{P}} \tau_p \text{ s.t. } (4.1\text{a}) - (4.1\text{f}). \tag{4.2}$$

To the best of our knowledge, this is the first optimization formulation for the ILMTP. The ILMTP is a generalization of the problem considered in Wang [2017], which was shown to be NP-Hard. Hence, the ILMTP is NP-Hard.

4.4 Theoretical Results

In this section, we analyze the structure of optimal solutions to the ILMTP under the following assumptions:

Assumption 4.1. *The MT services in $u\in\mathcal{U}$ are identical and serve all boarding stations; i.e. $\mathcal{U}(s)=\mathcal{U}$ for all $s\in\mathcal{S}$ and $t^{TO}(u)-t^{\text{start}}(u,s) = t^{TO}(u')-t^{\text{start}}(u',s)$ for all $u,u'\in\mathcal{U}(s)$ and $s\in\mathcal{S}$.*

Assumption 4.2. *Each LMT trip serves a single destination.*

Assumption 4.3. *For all pairs of passengers $p_1, p_2 \in \mathcal{P}$, $\texttt{ded}(p_1) - \texttt{rel}(p_1) = \texttt{ded}(p_2) - \texttt{rel}(p_2)$.*

Although Assumptions 4.1-4.3 are restrictive, the analysis in this section shows that there exists an optimal solution satisfying a particular ordering property. We exploit this in devising an heuristic, which we show through experimental evaluation is effective at identifying high-quality solutions.

Prior to presenting the technical results, additional notation is in order. We define a *group* to be a set of passengers that ride together in a CV on their LMT trip. Thus, a group consists of passengers having: the same start time on a common CV; cardinality less than the capacity of the CV; and a common destination (by Assumption 4.2). For any solution, let $G_1, \dots, G_{\mathcal{L}}$ denote the partitioning of passengers into groups. Let $G(p_j)$ be the group that contains passengers p_j. We assume, without loss of generality, that the passengers are ordered $p_1, \dots, p_{|\mathcal{P}|}$ so that $i < j \iff \texttt{rel}(p_i) \leq \texttt{rel}(p_j)$. Denote by $\text{CVT}(G_\ell)$ the time that the CV for group G_ℓ reaches the destination building.

We begin the analysis with some preliminary results.

Lemma 4.4. *All passengers in a group have the same wait time between services in an optimal solution to the ILMTP.*

Proof. Due to Assumption 4.1, in any optimal solution, all passengers arrive with the latest MT service that arrives at `T0` prior to their departure time on the CV. Hence, their wait times are the same. □

Following Lemma 4.4, let $\text{TT0}(G_\ell)$ denote the time that G_ℓ reaches `T0` on the MT in an optimal solution to ILMTP.

Lemma 4.5. *Let $G_1, \dots, G_{\mathcal{L}}$ be the groupings in an optimal solution to an ILMTP instance. If passengers p_1, p_2 are such that $b(p_1) = b(p_2)$ and $G(p_1) \neq G(p_2)$, then exchanging passengers p_1 and p_2 between groups $G(p_1)$ and $G(p_2)$ (holding all else equal) does not affect the solution value.*

Proof. For each passenger p, let τ_p be the original total transit time and τ'_p be the resulting total transit time for that passenger. The only change to the objective function is the change in the total transit times for these two passengers. In particular, the resulting transit times for the passengers are $\tau'_{p_1} = \tau_{p_1} + \Delta_2 - \Delta_1$ and $\tau'_{p_2} = \tau_{p_2} + \Delta_1 - \Delta_2$ where $\Delta_i = \text{CVT}(G(p_i)) - \text{TT0}(G(p_i))$. The net change in the objective function value is $\left(\tau'_{p_1} + \tau'_{p_2}\right) - (\tau_{p_1} + \tau_{p_2}) = 0$. □

Lemma 4.5 states that we can exchange passengers between groups that are going to the same destination without affecting the objective. Note that Lemma 4.5 does not make any claim on the feasibility of the groupings after the exchange.

Lemma 4.6. *For any optimal solution and any group G_ℓ, let $p_{i_1}, p_{i_2} \in G_\ell$, for $1 \leq i_1 < i_2 \leq |\mathcal{P}|$. If $|G_\ell| < CV^{\max}$, then for any i with $i_1 < i < i_2$, moving passenger p_i into group G_ℓ preserves feasibility.*

Proof. We need only show that $\texttt{rel}(p_i) \leq \mathrm{CVT}(G_\ell) \leq \texttt{ded}(p_i)$. The first inequality follows because $\texttt{rel}(p_i) \leq \texttt{rel}(p_{i_2})$, by the assumption on the ordering of the passengers, and $\texttt{rel}(p_{i_2}) \leq \mathrm{CVT}(G_\ell)$, by the feasibility of the original solution. The second inequality follows because $\mathrm{CVT}(G_\ell) \leq \texttt{ded}(p_{i_1})$, by the feasibility of the solution, and $\texttt{ded}(p_{i_1}) \leq \texttt{ded}(p_i)$, by the assumption on the ordering of the passengers, together with Assumption 4.3. □

We now state and prove, in Theorem 4.7, the main result of the section, which shows that an optimal solution exists where the passengers in a group have consecutive deadlines.

Theorem 4.7. *Suppose the passengers $\{p_1, \ldots, p_n\}$ have identical destinations and are ordered so that $i \leq j \iff \texttt{rel}(p_i) \leq \texttt{rel}(p_j)$. For any ILMTP instance there is an optimal solution with groupings $G_1, \ldots, G_{\mathcal{L}}$ for which if $p_j, p_{j+k} \in G_\ell$, for some $j \in \{1, \ldots, n-2\}, k \geq 2$ with $j+k \leq n$,b and $\ell \in \{1, \ldots, \mathcal{L}\}$, then $p_{j+1} \in G_\ell$.*

Proof. By way of contradiction, suppose there exists an instance for which there is no optimal solution satisfying the condition of the theorem. Consider the optimal solution for which the smallest index j that violates this condition is maximized. Let j^* be the smallest index in this solution for which there exists a k with $G(p_{j^*}) = G(p_{j^*+k})$ and $G(p_{j^*+1}) \neq G(p_{j^*})$. Let k^* be such an index k, $G_\ell = G(p_{j^*})$ and $G_{\ell'} = G(p_{j^*+1})$. We first show that $j \geq j^* + 1$ for all $\{j \,|\, p_j \in G_{\ell'}\}$. Suppose not; let $\hat{j} = \arg\max\{j \,|\, p_j \in G_{\ell'}, p_{j+1} \notin G_{\ell'}, j < j^* + 1\}$ (which will be non-empty by assumption). Then passenger indices j', $j' + k'$ with $j' = \hat{j}, k' = j^* + 1 - \hat{j}$ satisfy $p_{j'}, p_{j'+k'} \in G_{\ell'}$ and $p_{j'+1} \notin G_{\ell'}$ are a set of indices violating the claim of the theorem, contradicting the minimality of j^*. Hence, $(j^* + 1)$ is the minimum index among all $p_j \in G_{\ell'}$.

In the remainder of the proof, we construct another solution in which the index j^* does not violate the claims of the theorem. This contradicts the maximality of j^* among all optimal solutions, thereby establishing the result.

Conditioning on the relative values of the departure times of the CVs for G_ℓ and $G_{\ell'}$, first consider the case where $\mathrm{CVT}(G_\ell) \leq \mathrm{CVT}(G_{\ell'})$. We claim that exchanging the group assignment of p_{j^*+1} and $p_{j^*+k^*}$ and holding all else equal results in another feasible solution. We need only show that (a) $\texttt{rel}(p_{j^*+1}) \leq \mathrm{CVT}(G_\ell) \leq \texttt{ded}(p_{j^*+1})$ and (b) $\texttt{rel}(p_{j^*+k^*}) \leq \mathrm{CVT}(G_{\ell'}) \leq \texttt{ded}(p_{j^*+k^*})$. (a) follows directly from Lemma 4.6 with $i_1 = j^*, i_2 = j^* + k^*$ and $i = j^* + 1$. The first inequality in (b) follows because $\texttt{rel}(p_{j^*+k^*}) \leq \mathrm{CVT}(G_\ell)$, by the feasibility of the original solution, and $\mathrm{CVT}(G_\ell) \leq \mathrm{CVT}(G_{\ell'})$, by assumption. The second inequality in (b) holds because $\texttt{ded}(p_{j^*+1}) \leq \mathrm{CVT}(G_{\ell'})$, by the feasibility of the original solution and $\texttt{ded}(p_{j^*+1}) \leq \texttt{ded}(p_{j^*+k^*})$, by the ordering of passengers. Furthermore, by Lemma 4.5 the objective function remains unchanged by this exchange, and is therefore optimal. If the resulting solution satisfies the claim of this theorem, then the claim holds. If not, then the claim of this theorem is violated for another $j > j^*$; contradicting the maximality of j^*.

We now consider the alternative case where $\mathrm{CVT}(G_\ell) > \mathrm{CVT}(G_{\ell'})$. The exchange from the previous case may not work because putting passenger $p_{j^*+k^*}$ into group $G_{\ell'}$ may not be feasible. Additionally, if there exist $k' > 0$ passengers in G_ℓ with indices lower than j^* then these passengers must be $p_{j^*-k'}, \ldots, p_{j^*-1}$. If not, it would contradict the assumption of j^* as the smallest index in the optimal solution violating the claim of this theorem. Define k' so that $p_{j^*-k'}$ is the minimum indexed passenger in group G_ℓ, which is 0 if p_{j^*} is the minimum indexed passenger.

Consider the following two-step exchange—for $i = 0, \ldots, k'$, move each passenger p_{j^*-i} from G_ℓ into $G_{\ell'}$. Then, move the $k' + 1$ passengers with the highest indices in the resulting $G_{\ell'}$ into G_ℓ. The resulting groups have the same cardinality as they originally had, and so by Lemma 4.5 the objective values remain the same.

We now show that the resulting solution is valid and then show that the choice of optimal solution contradicts the maximality assumption on the selection of j^*, which concludes the proof. Any passenger $p_i \in G_\ell$ with index $i \leq j^*$ can be moved to $G_{\ell'}$ without violating p_i's arrival time window because $\texttt{rel}(p_i) \leq \texttt{rel}(p_{j^*+1}) \leq \text{CVT}(G_{\ell'}) < \text{CVT}(G_\ell)$ and $\texttt{ded}(p_i) \geq \text{CVT}(G_\ell) > \text{CVT}(G_{\ell'})$. Additionally, any passenger $p \in G_{\ell'}$ can be moved to G_ℓ without violating the arrival time windows because $\texttt{rel}(p) \leq \text{CVT}(G_{\ell'}) < \text{CVT}(G_\ell)$ and $\texttt{ded}(p) \geq \texttt{ded}(p_{j^*}) \geq \text{CVT}(G_\ell)$. Hence, the resulting solution is also optimal. Finally, if the resulting solution violates the claim of this theorem, then the smallest index must be larger than j^*. This again contradicts the maximality of j^* among all optimal solutions, as assumed. □

Note that we can independently reorganize the set of passengers for each destination to be sorted as indicated in Theorem 4.7. The result can thus be extended to instances with multiple destinations, with the additional assumption that each CV is restricted to carry passengers going to a common building (Assumption 4.2).

4.5 Algorithm

We describe an algorithmic framework for constructing a heuristic solution and improving it with local search.

The heuristic construction comprises three phases: (i) sorting the passengers; (ii) grouping them; and (iii) assigning groups to vehicles and scheduling the LMT trips.

4.5.1 Sorting the Passengers

We define a bijection $\texttt{ordp} : \{1, \ldots, |\mathcal{P}|\} \to \mathcal{P}$ such that

$$\texttt{ded}\big(\texttt{ordp}(i)\big) \leq \texttt{ded}\big(\texttt{ordp}(j)\big), 1 \leq i < j \leq |\mathcal{P}|.$$

The intuition is that passengers with similar arrival deadlines can be grouped and served together.

Algorithm 10 Groups the passengers by destination

```
 1: function GroupPassengersByDestination($\mathcal{P}$, ordp)
 2:     $\mathcal{G} \leftarrow \emptyset$
 3:     for $i = 1 \rightarrow |\mathcal{P}|$ do
 4:         $\text{Grouped}_i \leftarrow$ False
 5:     end for
 6:     for $i = 1 \rightarrow |\mathcal{P}|$ do
 7:         if not $\text{Grouped}_i$ then
 8:             $p \leftarrow \texttt{ordp}(i)$
 9:             $g \leftarrow \{p\}$
10:             $j \leftarrow i + 1$
11:             while $j \leq |\mathcal{P}|$ and $|g| < CV^{\max}$ do
12:                 if not $\text{Grouped}_j$ then
13:                     $q \leftarrow \texttt{ordp}(j)$
14:                     if $\texttt{rel}(q) < \texttt{ded}(p)$ and $\text{b(p)} = \text{b(q)}$ then
15:                         $g \leftarrow g \cup \{q\}$
16:                         $\text{Grouped}_j \leftarrow$ True
17:                     end if
18:                 end if
19:             end while
20:             $\mathcal{G} \leftarrow \mathcal{G} \cup \{g\}$
21:         end if
22:     end for
23:     return $\mathcal{G}$
24: end function
```

4.5.2 Grouping the Passengers

Algorithm 10 defines groups of passengers with common destination by traversing the set of passengers as sorted by `ordp`. For each passenger $p = \texttt{ordp}(i)$ that does not have a group yet, i.e., $\text{Grouped}_i = \texttt{False}$, the loop starting at line 6 creates a new group g. Then the loop starting at line 11 verifies if each of the next ungrouped passengers $q = \texttt{ordp}(j)$ for $j > i$ can be added to group g. Since the passengers are added as ordered by `ordp`, their deadlines are nondecreasing and the verification that the earliest arrival time for q is before the latest time for p in line 14 suffices to determine if the resulting group is feasible for a single destination.

4.5.3 Optimal Scheduling of the Groups

We now present an ILP formulation to obtain an optimal schedule of groups on MT and LMT for a given grouping of passengers for the LMT trips. For ease of exposition, we begin by describing some sets that can be pre-computed for a fixed grouping.

The set of feasible start times on CVs for a group $g \in \mathcal{G}$ using route r can be computed as

$$\mathcal{T}(g, r) := \left\{ t \;\middle|\; \begin{array}{r} (t + \tau^{\text{travel}}(r, b(p))) \in [\texttt{rel}(p), \texttt{ded}(p)] \\ \forall\, p \in g \end{array} \right\}.$$

We use binary variables $x_{g,r,t} \in \{0, 1\}$ to represent the start time t for group g on a CV route

r, i.e.

$$x_{g,r,t} = \begin{cases} 1 \text{ if group } g \text{ uses route } r \text{ and starts at } t \\ 0 \text{ otherwise} \end{cases}$$
$$\forall\, g \in \mathcal{G}, r \in \mathcal{R}(g), t \in \mathcal{T}(g,r),$$

where $\mathcal{R}(g)$ is the set of CV routes that stop at destinations of passengers in g; i.e.

$$\mathcal{R}(g) := \{r \in \mathcal{R} \mid b(p) \in \mathcal{B}(r) \,\forall\, p \in g\}\,.$$

To model the use of CVs by different groups, we compute a set defining the groups, routes, and start times directly affecting the number of CVs used at each time instant t as

$$\mathcal{V}(t) := \left\{(g,r,t') \,|\, t \in [t', t' + \tau^{\texttt{trip}}(g,r)]\right\}.$$

Each of these is a subset of the indices of decision variables and its use will be clear in the upcoming formulation.

The exact ILP modeling the optimal scheduling of passengers on CVs for LMT service is as follows:

$$\min \sum_{g\in\mathcal{G}} \sum_{r\in\mathcal{R}(g)} \sum_{t\in\mathcal{T}(g,r)} \alpha_{g,r,t} \cdot x_{g,r,t} \tag{4.3a}$$

$$\text{s.t.} \sum_{r\in\mathcal{R}(g)} \sum_{t\in\mathcal{T}(g,r)} x_{g,r,t} = 1 \,\forall\, g \in \mathcal{G} \tag{4.3b}$$

$$\sum_{(g,r,t')\in\mathcal{V}(t)} x_{g,r,t'} \leq |\mathcal{C}| \,\forall\, t \in \mathcal{T}. \tag{4.3c}$$

The objective coefficient $\alpha_{g,r,t}$ is defined as

$$\alpha_{g,r,t} := \sum_{p\in g} \left(t + \tau^{\text{travel}}(r, b(p)) - t^{\text{start}}(u^{min}(t, s(p)))\right),$$
$$\text{where } u^{\text{min}}(t, s(p)) := \arg \min_{u\in\mathcal{U}(s(p)):t^{\texttt{T0}}(u)\leq t} (t - t^{\text{start}}(u)).$$

Constraint (4.3b) imposes that each group is assigned to exactly one CV route and start time. Constraint (4.3c) ensures that the number of groups that are on the LMT service at any time is less than or equal to the total number of CVs.

For a given optimal solution $x^*_{g,r,t}$ to (4.3), we can obtain an optimal assignment of routes $r^* : \mathcal{G} \to \mathcal{R}$ and start times $t^* : \mathcal{G} \to \mathcal{T}$ for groups on CVs as follows:

$$r^*(g) = r \iff x^*_{g,r,t} = 1 \text{ and } t^*(g) = t \iff x^*_{g,r,t} = 1.$$

The optimal MT services $u^* : \mathcal{P} \to \mathcal{U}$ are then obtained as

$$u^*(p) = u^{\text{min}}(t^*(g), s(p),) \text{ where } g : p \in g.$$

Algorithm 11 Assigns vehicles to scheduled groups

```
function AssignVehiclesToGroups($\mathcal{C}$, $\texttt{ordp}$, $r^*(.)$)
    $\texttt{veh\_queue} \leftarrow \emptyset$
    for $v \in \mathcal{C}$ do
        $\texttt{veh\_queue}.\text{push}\big((v, 0)\big)$
    end for
    for $i \leftarrow 1, \dots, |\mathcal{G}|$ do
        $(v, t) \leftarrow \texttt{veh\_queue}.\text{pop}()$
        $g \leftarrow \texttt{ord}(i)$
        $\texttt{veh}(g) \leftarrow v$
        $\texttt{veh\_queue}.\text{push}\big((v, t + \tau^{\texttt{trip}}(g, r^*(g)) + 1)\big)$
    end for
    return $\texttt{veh}$
end function
```

Formulation (4.3) does not explicitly assign a CV to any of the groups. The CV assignments can be determined once the solution is given by Algorithm 11. The algorithm makes use of a priority queue $\texttt{veh\_queue}$ consisting of (v, t) pairs, where v is index of the CV and t is time that the CV is available at $\texttt{T0}$ for servicing a group. The pairs with lower t have higher priority in the queue. Let $\texttt{ord} : \mathcal{G} \to \{1, \dots, |\mathcal{G}|\}$ be a mapping that sorts $\mathcal{G}$ in ascending order of $t^*(g)$, i.e.

$$\texttt{ord}(g) \leq \texttt{ord}(g') \iff t^*(g) \leq t^*(g').$$

Algorithm 11 returns a mapping $\texttt{veh}\colon \mathcal{G} \to \mathcal{C}$ such that $\texttt{veh}(g)$ is the vehicle assigned to service the group g.

Some comments regarding formulation (4.3) are in order. In contrast to formulation (4.2), the time indices in formulation (4.3) for each group are restricted to the intersection of possible departures for all passengers in the group, hence making the formulation more tractable. Furthermore, formulation (4.3) also avoids the combinatorial explosion in the number of routes, which takes all possible subsets of $CV^{\max}$ stops. It is possible to define smaller formulations in both cases by modeling time with continuous variables instead of indexing binary variables, as discussed by Floudas and Lin (2004). However, Balas (1985) has shown that the linear relaxation of such formulations is the weakest possible, thereby pushing the solution process towards a time-consuming enumeration of alternatives that does not truly benefit from the ILP techniques that mathematical optimization solvers exploit.

4.5.4 Lower Bounds

This section describes lower bounds on MT and LMT trip times per passenger as well as on the sum of wait and LMT trip times per passenger and per group of passengers. We use these bounds to evaluate the quality of solutions and also to direct the local search methods as described next.

The minimum MT trip time per passenger is defined as

$$\mathrm{MMT}(p) := \min\Big\{t^{\mathtt{TO}}(u) - t^{\mathrm{start}}\big(u, s(p)\big) \mid u \in \mathcal{U}\big(s(p)\big)\Big\}. \tag{4.4}$$

In turn, the minimum LMT trip time is given by

$$\mathrm{MLM}(p) := \min\Big\{\tau^{\mathrm{travel}}\big(r, b(p)\big) \mid r \in \mathcal{R},\ b(p) \in \mathcal{B}(r)\Big\}. \tag{4.5}$$

For convenience, we define a function corresponding to the minimum waiting time if a passenger coming from station s leaves the terminal in the time window $[t_a, t_b]$:

$$\mathrm{MWT}(s, t_a, t_b) := \min\left\{t_a - t^{\mathtt{TO}}(u) \,|\, u \in \mathcal{U}(s), t^{\mathtt{TO}}(u) \leq t_b\right\}. \tag{4.6}$$

This waiting time is implied by the set of MT trips that can bring a passenger from s to the terminal within $[t_a, t_b]$. In turn, the time window may be restricted according to the passengers that are grouped together.

For each passenger p, we can compute the minimum waiting time if the passenger takes the shortest travel time as

$$\mathrm{MPWT}(p) := \mathrm{MWT}\left(s(p), \mathtt{rel}(p) - \mathrm{MLM}(p), \mathtt{ded}(p) - \mathrm{MLM}(p)\right). \tag{4.7}$$

Note that this is not a bound on the waiting time, since passenger p could wait less if taking a longer LMT trip. However, trading waiting for trip time would never lead to a smaller sum of both, and thus the combined bound is valid. We can therefore obtain a lower bound **LB** that combines the trip times and the wait times for each passenger as

$$\mathbf{LB} := \sum_{p \in \mathcal{P}} \mathrm{MMT}(p) + \mathrm{MLM}(p) + \mathrm{MPWT}(p). \tag{4.8}$$

If the number of CVs is large enough to define groups where no passenger delays another and all groups can be scheduled at the most convenient time for its passengers, then **LB** corresponds to the value of the optimal solution. When this is not the case, we need to account for the tighter departure time windows defined by each group.

For a group of passengers g, the minimum wait time of each passenger $p \in g$ if taking a trip of shortest time is

$$\mathrm{MGWT}(g, p) := \mathrm{MWT}\left(s(p), \max_{q \in g}(\mathtt{rel}(q) - \mathrm{MMT}(q)), \min_{q \in g}(\mathtt{ded}(q) - \mathrm{MMT}(q))\right). \tag{4.9}$$

Whenever $\mathrm{MGWT}(g, p) > \mathrm{MPWT}(p)$ for some $p \in g$, then some passenger in $g \setminus \{p\}$ is delaying p. While the former expressions define a lower bound per passenger, the latter is used to decide which groups to modify by local search.

4.5.5 Break-and-Shift Local Search

Algorithm 10 aims to create as few groups as possible, each with many passengers. This can increase total waiting time because the CV fleet may be underutilized. Hence, we define a local search method to iteratively modify these groups by breaking them and swapping passengers.

We define a local search to address this, which loops between two operations: (i) breaking groups where one passenger delays another; and (ii) shifting passengers to groups with earlier departure times when this is feasible and no delay is implied. These operations are performed on sets of groups that arrive through MT trips at the same time to $\mathtt{TO}$ according to an optimal solution of (4.3). If the LMT trips take less than the inter-arrival time of MT, operation (i) can result in as many groups as the number of CVs ($|\mathcal{C}|$). The fragmented groups resulting from operation (i) can be reorganized differently by operation (ii), whereby some groups might vanish. In such a case, we are able to repeat a loop with both operations once more. Hence, we define a set of MT arrival times as

$$T := \left\{ t^{\mathtt{TO}}(u) \mid u \in \mathcal{U}(s), s \in \mathcal{S} \right\} \tag{4.10}$$

and the groups corresponding to each $t \in T$ is

$$G_t := \Big\{ g \in \mathcal{G} \mid t = \max\left\{ t^{\mathtt{TO}}(u) \mid u \in \mathcal{U}, t^{\mathtt{TO}}(u) \le t^*(g) \right\} \Big\}. \tag{4.11}$$

For convenience, we denote the set of passengers on each group g as $g := \big\{g(1), \dots, g(|g|)\big\}$, with $\mathtt{ded}\big(g(i)\big) \le \mathtt{ded}\big(g(j)\big)$ for $i \le j$. Furthermore, let G^b denote the set of groups with all passengers heading to b, and let $G^b := \Big\{ g^b_1, \dots, g^b_{|G^b|} \Big\}$ with $t^*(g^b_i) \le t^*(g^b_j)$ for $i \le j$. Algorithm 12 describes operation (i), which breaks down some groups in G_t. Staring with a set H_t corresponding to G_t, the procedure loops over each group $g \in G_t$, attempting to break them to increase the objective value of the solution. If the total number of groups is still smaller than the number of CVs and the earliest passenger $g(1)$ has to wait more with the group than alone, the group is broken before the first passenger $g(i)$ that causes the delay, thereby replacing g with two groups in H_t. Algorithm 13 describes operation (ii), which moves passengers between consecutive groups. In this case, H^b begins empty, we refer to the next group to be added to H^b as h, and we loop on G^b to construct this set H^b. We define the first group in G^b as the initial incumbent group h and, while possible, we keep adding passengers to it by looping on the subsequent groups. We either empty these groups on h and proceed to the next one, or else fill h to capacity. In the latter case, we add h to H^b and reassign h as h', which consists of the remaining passengers from the last group of G^b that we iterated on.

In order to avoid infeasibility when breaking and reorganizing the groups, one can solve (4.3) after each modification. For efficiency, we have found that it was sufficient to apply both operators

Algorithm 12 Breaks groups with co-passenger delays

```
function BreakGroups($G_t$)
    $H_t \leftarrow G_t$
    for $g \in G_t$ do
        if $|H_t| < |\mathcal{C}|$ and $\text{MGWT}\big(g, g(1)\big) > \text{MPWT}\big(g(1)\big)$ then
            for $i \leftarrow 2, \ldots, |g|$ do
                if $\text{MPWT}\big(g(i)\big) > \text{MPWT}\big(g(1)\big)$ then
                    $g' \leftarrow \{g(1), \ldots, g(i-1)\}$
                    $g'' \leftarrow \{g(i), \ldots, g(|g|)\}$
                    $H_t \leftarrow \big(H_t \setminus \{g\}\big) \cup \{g', g''\}$
                    break
                end if
            end for
            if $|H_t| = |\mathcal{C}|$ then
                break
            end if
        end if
    end for
    return $H_t$
end function
```

until either there was no change in the groups or the groups resulted in an infeasible schedule.

4.6 Experiments

We present numerical experiments in the setting of Figure **??**. The passengers originate from a set of 4 stations and desire to reach one of the buildings in B1-B10 within a specified time-window. $\mathcal{U}$ consists of MT services that reach T0 every 15 minutes and the travel time between stations is 5 minutes. We assume that the CVs are all parked at T0 and return back to the T0 after dropping off all passengers. The CVs take 1 minute to go between T0-B1, T0-B10, and all pairs of buildings that are adjacent on the shaded track, except for B5-B6 which takes 2 minutes. The CVs are restricted to move along the shaded region shown in Figure **??**. We also assume that a CV spends 0.5 minutes at a building where they drop passengers. As a result, the time that a passenger reaches the destination depends on co-passengers in the CV that have prior destinations. The modeling of drop-off time is important in applications where the capacity of $CV^{\max}$ is small, typically ≤ 5. Inspired by the application to corporate-campus settings, we use a small number of destinations. This may not be the case in all applications, where one might expect the number of destinations for the LMT service to be in the same order as the number of passengers, thus making the problem harder to solve. In those cases, however, we believe that the suggestion by Mahéo et al. (2018) to treat last-mile stops as aggregations of several passenger destinations, such as bus stops, is a reasonable compromise. Hence, the chosen number of destinations is of minor importance.

In the experiments performed, we restrict the passengers to be grouped according to their destination. The set of possible CV round-trip movements considered are (in the sequence of

Algorithm 13 Reorganizes passenger groups

```
function SHIFTPASSENGERS($G^b$)
    $H^b \leftarrow \emptyset$
    $h \leftarrow g_1^b$
    for $i \leftarrow 2, \dots, |G^b|$ do
        if $|h| = CV^{\max}$ then
        end if
        if $|h| < CV^{\max}$ then
            for $j \leftarrow 1, \dots, \min\{CV^{\max} - |h|, |g_i^b|\}$ do State $p \leftarrow g_i^b(j)$
                if $\texttt{rel}(p) \leq \texttt{ded}\big(h(1)\big)$ and $\mathrm{MGWT}\big(h \cup \{p\}, h(1)\big) = \mathrm{MGWT}\big(h, h(1)\big)$ then
                    $h \leftarrow h \cup \{p\}$
                end if
            end for
            $h' \leftarrow g_i^b \setminus h$
            if $|h'| > 0$ then
                $H^b \leftarrow H^b \cup \{h\}$
                $h \leftarrow h'$
            end if
        end if
    end for
    $H^b \leftarrow H^b \cup \{h\}$
    return $H^b$
end function
```

building visits)

$$\{\texttt{T0}, B1, B2, \dots, B9, B10, \texttt{T0}\}, \{\texttt{T0}, B10, B9, \dots, B2, B1, \texttt{T0}\}$$

$$\{\texttt{T0}, B1, B2, B3, B8, B9, B10, \texttt{T0}\}, \{\texttt{T0}, B10, B9, B8, B3, B2, B1, \texttt{T0}\}$$

$$\{\texttt{T0}, B1, \dots, B3, B8, B7, \dots, B3, B8, \dots, B10, \texttt{T0}\}$$

$$\{\texttt{T0}, B10, \dots, B8, B3, B4, \dots, B8, B3, \dots, B1, \texttt{T0}\}.$$

On each of the CV round-trip movements, a stop at the first occurrence of the building is considered a route r for the CV. Thus, $\mathcal{R}$ consists of 52 (= 10 + 10 + 6 + 6 + 10 + 10) routes. From an optimality perspective, we expect the CVs to typically use only the shortest route to the destination. However, in a few instances the passengers with earliest deadlines are assigned to longer routes so that they arrive to their destinations at the start of the time-windows. Removing this option can result in infeasibility.

To test the algorithms described, we generated 10 scenarios consisting of 600 passengers. The desired earliest time ($\texttt{rel}(p)$) of arrival for the passengers is assumed to be uniformly distributed over 1 hour, in increments of 30 seconds. For each passenger, the origin and destination station are drawn uniformly and independently at random from the four stations and 10 buildings, as shown in Figure **??**. We assume that the length of time windows ($\texttt{ded}(p) - \texttt{rel}(p)$) are identical for all the passengers; the values are set by assigning, if t' is the requested arrival time for a passenger, $\texttt{rel}(p) = t' - K/2$ and $\texttt{ded}(p) = t' + K/2$, for a fixed $K > 0$. We test the impact of K and the number of CVs by varying $K \in \{5, 10, 20\}$ and $|\mathcal{C}| \in \{30, 40, 50, 60\}$, resulting in 12 different configurations, and thus 120 different instances. All experiments were run on a machine with an

Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz and 32 GB RAM. All algorithms were implemented in `Python 2.7.6` and the ILPs are solved using `Gurobi 7.5.1`.

Table 4.2 presents a summary of the results. The first two columns report the size of the time windows and the number of CVs, respectively. Column $\text{UB}^{\text{red}} = (UB^0 - UB^H)/(UB^0 - LB^*) \times 100$ is the percent decrease in the optimality gap between the initial solution obtained using the heuristic (UB^0) and the final (UB^H) solution resulting from the local search procedure, over the best known lower bound (LB^*) obtained by `Gurobi` solving model (4.2) with a time limit of 10 minutes, averaged over all instances in that configuration. A 100% reduction means the entire optimality gap is closed. It is clear that the heuristic is able to obtain optimal solutions on problems where the number of CVs and time windows are not very constrained.

The fifth and sixth columns in Table 4.2 show the average percent optimality gap at termination over the instances where solutions are found when solving model (4.2) using `Gurobi`: (i) without the heuristic solution (MIP) and (ii) with the heuristic solution (MIP+H) as an initial solution using the `MIPStart` feature. In both approaches, the time limit is set to 10 minutes. The optimality gap is computed as $(UB^* - LB^*)/LB^* \times 100$ where UB^* is the best feasible solution and LB^* is the best lower bound obtained by the approach (MIP or MIP+H). In all cases, the gaps are computed after subtracting the constant MT travel times from each station of origin to `T0`. In the cases that not all runs found feasible solutions, next to each average there is the number of cases where at least one feasible solution is found. The last two columns in Table 4.2 show the number of instances that were solved to optimality. From the results in Table 4.2, it is clear that MIP+H outperforms MIP in terms of the number of problems solved to optimality and average optimality gap closed.

Table 4.2: Summary of results averaging 10 instances per configuration of time window length K and flee size $|\mathcal{C}|$

				Avg. Gap (%)			Solved	
K	$\lvert\mathcal{C}\rvert$	UB^{red} (%)	t^{H}	MIP		MIP+H	MIP	MIP+H
5	30	2.9	0.99	0.0	(1)	0.1	1	4
5	40	14.3	3.73	0.0	(7)	0.0	7	10
5	50	31.4	6.38	0.0	(8)	0.0	8	10
5	60	60.2	15.25	0.0		0.0	10	10
10	30	6.5	3.65	–	(0)	2.0	0	0
10	40	34.0	12.31	0.3	(6)	0.2	2	4
10	50	76.1	25.97	0.0		0.0	10	10
10	60	96.0	32.23	0.0		0.0	10	10
20	30	62.4	7.99	0.0		0.0	10	10
20	40	100.0	8.7	0.0		0.0	10	10
20	50	100.0	7.62	0.0		0.0	10	10
20	60	100.0	7.54	0.0		0.0	10	10

We also investigate the effect on the solution quality due to the imposition of single destination per CV in Assumption 4.2. An upper bound to the optimal gap with respect to the general case is computed as $\dfrac{(UB(MIP+H) - \mathbf{LB})}{\mathbf{LB}}$, where $UB(MIP+H)$ is the best solution obtained from our algorithm and $\mathbf{LB}$ is the lower bound in (4.8). Note that the computation of $\mathbf{LB}$ in (4.8)

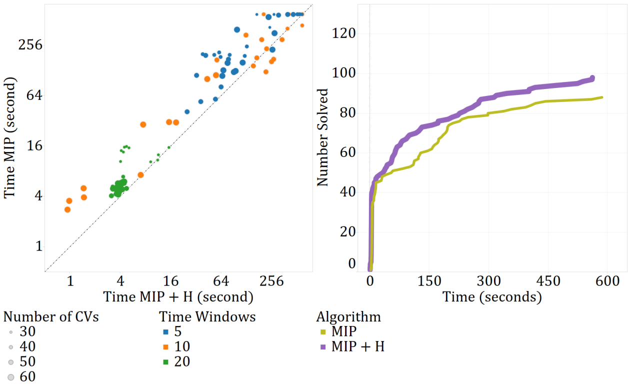


Figure 4.1: Comparison of model 4.2 with and without the heuristic. (left) A scatter plot with a point were run. (right) A cumulative distribution plot of performance indicating the number of instances solved by second.

is independent of the number of stops per CV. Table 4.3 summarizes the gaps for the solution obtained using MIP+H. These results indicate that Assumption 4.2 is not very restrictive on the instances tested.

Table 4.3: Upper bound on the optimal gap of the solutions obtained under Assumption 4.2 with respect to the general case.

K	$\|\mathcal{C}\|$	Avg. Gap %	Min. Gap %	Max. Gap %
5	30	6.2	5.6	6.7
5	40	2.7	2.3	3.2
5	50	1.3	1.0	1.6
5	60	0.5	0.4	0.8
10	30	12.3	10.7	14.32
10	40	2.9	2.5	3.5
10	50	0.7	0.4	0.9
10	60	0.1	0.0	0.2
20	30	0.0	0.0	0.0
20	40	0.0	0.0	0.0
20	50	0.0	0.0	0.0
20	60	0.0	0.0	0.0

Figure 4.1 depicts (left) a scatter plot with a point per instance and (right) a cumulative distribution plot of performance, comparing the solution times for MIP and MIP+H. The scatter plot contains a point per instance where the coordinates of the point correspond to the time taken to solve the instance with and without the heuristic. The plots are sized according to $|\mathcal{C}|$ and colored according to K. The cumulative distribution plot of performance provides another depiction of the data, showing the number of instances that are solved at each point of time in the solution horizon for both algorithmic configurations.

4.7 Conclusions and Future Work

This chapter addressed the ILMTP, focusing on a special case of this problem where each CV makes a stop at just one building per trip. Based on this setting, we prove a structured result regarding the form of optimal solutions that results in insights used to efficiently solve the problem. Our experiments indicate that the optimal solutions to this case are not far from a lower bound for the general problem, suggesting that (a) starting from this solution for a future study is promising, and (b) real-world systems implementing this solution may already achieve near optimal schedules.

We generalize and make improvements to a recent MIP formulation for the general problem suggested by Wang (2017), and use a constructive heuristic and local search procedure for identifying an initial high-quality solution that, when used as the starting solution for a commercial MIP solver, often closing the entire optimality gap. The results obtained with and without the heuristic also indicate a significant improvement in solution times. The instances appear to increase in difficulty as K and $|\mathcal{C}|$ decrease, which can be attributed to the problem becoming more constrained.

As the design of automated transportation systems becomes a reality, it is critical that models and algorithms such as the ones developed in this work are developed to ensure the system used are run efficiently. One critical extension will be to the case when CVs can stop at more than one building. The local search algorithm developed can be modified to accept solutions of this form, which may reduce the total waiting times of customers. Additionally, we plan to design problem-specific solution algorithms that can exploit results like the one proven in Theorem 4.7.

Another avenue for extending this work would be to exploit its complementarity with the study by Mahéo et al. (2018), where more consideration is given to designing the MT service. We hope that a unified framework can be developed in future work.

Conclusion

This thesis tackled a variety of topics in integer and mixed-integer linear programming. In a nutshell, we have explored how to operationalize the use of decision diagrams for postoptimality analysis of integer linear programs, we have analyzed and evaluated lift-and-project cuts for mixed-integer linear programs, and we have accelerated the solution of a novel scheduling application.

On postoptimality, our goal is to facilitate answering what-if questions on problems that are relatively easier to solve by storing the solutions for these problems in a compact but transparent data structure. In that context, decision diagrams can be regarded as a comparatively better data structure to store near-optimal solutions since, in contrast to using the tree that naturally arises as these solutions are obtained through a branch-and-bound method, there is an additional benefit from compressing nodes with same completions. Furthermore, decision diagrams can be efficiently queried for tasks such as determining the domain of discrete variables in near-optimal solutions, or the best possible solution upon fixing the value of certain variables. However, these diagrams can still be very large. Sound decision diagrams, which innocuously admit worse solutions, are particularly useful in this case because they preserve the equivalence between near-optimal solutions and near-optimal paths for a chosen optimality gap while exploiting to the maximum extent that decision diagrams with more solutions can be smaller. Our contributions here are the following:

- We have shown that an ordered tree or decision diagram of near optimal solutions can be efficiently compressed into a sound decision diagram of minimum size through the application, in any order, of a node merging operation that we denote as sound-reduction.

- We have shown that, unlike conventional decision diagrams that are compressed through reduction, the smallest sound decision diagrams are not necessarily unique.

- We have shown that extending the concept of sound decision diagrams to admit solutions that are either better or worse than a range of values defines a computationally hard problem.

- We have observed in computational experiments that the speed-up strategy of merging nodes representing equivalent subproblems of an integer linear program while branching naturally yields a sound decision diagram that in some cases is smaller than a conventional diagram.

On lift-and-project, our goal is to understand the cuts that can be obtained from optimal solutions of the Cut Generating Linear Program (CGLP). On the one hand, a CGLP for a split disjunction can be indirectly solved by pivoting among feasible and infeasible bases of the linear relaxation of the problem to which the cuts are generated, but there are different forms in which these cuts can be compared with each other according to the normalization and its interplay with the objective function in the CGLP. It has been observed by Fischetti et al. [2011] that this may result in cuts that are strictly dominated. Our contributions for this line of work are the following:

- We have proposed a new formulation that switches the roles previously played by normalization and objective function, the Reverse Polar CGLP (RP-CGLP), where the feasible set is an extended formulation of the reverse polar set from $\bar{x}$ and the objective function minimizes the evaluation of the cut for a point p in the disjunctive hull; and we have shown that cuts from RP-CGLP optima always define supporting hyperplanes of the disjunctive hull.

- We have shown that, if p is an interior point of the disjunctive hull, then the resulting cut is a combination of facet-defining cuts separating $\bar{x}$, all of which optimizing $\min \dfrac{\alpha p - \beta}{\beta - \alpha \bar{x}}$. In fact, the same is true of other formulations such as the BP-CGLP introduced by Balas and Perregaard [2002] and the P-CGLP suggested by Cadoux and Lemaréchal [2013] and analyzed by Conforti and Wolsey [2016], and they relate to a broader literature on in-out methods.

On the other hand, the correspondence between lift-and-project cuts and intersection cuts from bases on the linear relaxation does not necessarily hold in the case of non-split disjunctions [Andersen et al., 2005, Kis, 2014], and thus one may wonder if mimicking the CGLP on the tableau of the original problem may lead to the same cut that would have been obtained by solving the actual CGLP. Our contributions for this question are the following:

- We have simplified the necessary and sufficient condition shown by Balas and Kis [2016] for the correspondence between lift-and-project cuts and intersection cuts, which characterizes regular basic CGLP solutions and cuts, and shown that it applies to any CGLP solution.

- Using the extension above, we have introduced a mixed-integer formulation based on the CGLP and shown that it can be used to determine if a given cut corresponds to an intersection cut for the same problem, in which case we can find a corresponding basis and P_I-free set.

- We have used this formulation to evaluate lift-and-project cuts obtained from simple t-branch split disjunctions, and thus tested the extent to which these cuts are equivalent to the corresponding multi-row cuts. While families of problems differ on the incidence of regular lift-and-project cuts, we have found that the cuts tend to be irregular as the problems get larger, the linear relaxation is weaker, and the number of terms of the disjunction increases.

- We have also observed a convenient relationship among disjunctions leading to irregular cuts. Namely, the incidence of irregular cuts in optimal solutions of CGLPs from simple 3-branch or 4-branch split disjunction relates to the number of irregular cuts from simple 2-branch split disjunctions on the same variables, hence entailing a method to choose disjunctions with more terms that are more likely to yield irregular cuts.

On scheduling, our goal is to solve to optimality a last-mile passenger scheduling problem. While a convenient restriction of this problem can be loaded by a modern commercial solver with a time-indexed mixed-integer formulation, which is stronger than continuous time formulations, feasible solutions are often not found in reasonable time. Our contributions are as follows:

- We have shown that, subject to mild assumptions, there are optimal solutions to this problem that have a particular structure by which requests can be grouped by sorted deadlines.
- We have developed constructive and local search heuristics that preserve this structure while making a good use of the resources available to find a good scheduling solution.
- We have observed that using these heuristics to warm-start the mixed-integer solver lead to a significant impact in the runtime and the quality of the solutions obtained.

We believe that these contributions may be leveraged for future work, from developing more sophisticated forms of postoptimality analysis to finding efficient methods to generate irregular cuts and solving other scheduling problems through a mix of theory and computational experimentation.

Bibliography

T. Achterberg, T. Koch, and A. Martin. MIPLIB 2003. *Operations Research Letters*, 34(4):361–372, 2006.

T. Achterberg, S. Heinz, and T. Koch. Counting solutions of integer programs using unrestricted subtree detection. In L. Perron and M. A. Trick, editors, *Proceedings of CPAIOR*, pages 278–282. Springer, 2008.

S. B. Akers. Binary decision diagrams. *IEEE Transactions on Computers*, C-27:509–516, 1978.

H. R. Andersen, T. Hadžić, J. N. Hooker, and P. Tiedemann. A constraint store based on multivalued decision diagrams. In C. Bessiere, editor, *Principles and Practice of Constraint Programming (CP 2007)*, volume 4741 of *Lecture Notes in Computer Science*, pages 118–132. Springer, 2007a.

K. Andersen, G. Cornuéjols, and Y. Li. Split closure and intersection cuts. *Mathematical Programming*, 102(3):457–493, 2005.

K. Andersen, Q. Louveaux, R. Weismantel, and L. A. Wolsey. Inequalities from two rows of a simplex tableau. In M. Fischetti and D. P. Williamson, editors, *Integer Programming and Combinatorial Optimization*, pages 1–15. Springer Berlin Heidelberg, 2007b.

J. E. Anderson. Control of personal rapid transit systems. *J. Adv. Transportation*, 32(1):57–74, 1998.

D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study.* Princeton University Press, 2^{nd} edition, 2006.

J. A. Arthur, M. Hachey, K. Sahr, M. Huso, and A. R. Kiester. Finding all optimal solutions to the reserve site selection problem: Formulation and computational analysis. *Environmental and Ecological Statistics*, 4:153–165, 1997.

E. Balas. Intersection cuts – a new type of cutting planes for integer programming. *Oper. Res.*, 19: 19–39, 1971.

E. Balas. Disjunctive programming. *Annals of Discrete Mathematics*, (5):3–51, 1979.

E. Balas. On the facial structure of scheduling polyhedra. *Mathematical Programming Study*, 24: 179–218, 1985.

E. Balas. A modified lift-and-project procedure. *Mathematical Programming*, 79(1-3):19–31, 1997.

E. Balas. Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics*, 89(13):3 – 44, 1998.

E. Balas and P. Bonami. Generating lift-and-project cuts from the lp simplex tableau: open source implementation and testing of new variants. *Mathematical Programming Computation*, 1(2-3): 165–199, 2009.

E. Balas and R. G. Jeroslow. Strengthening cuts for mixed integer programs. *European Journal of Operational Research*, 4(4):224 – 234, 1980.

E. Balas and T. Kis. On the relationship between standard intersection cuts, lift-and-project cuts, and generalized intersection cuts. *Math. Program.*, 160:85–114, 2016.

E. Balas and M. Perregaard. Lift-and-project for mixed 0–1 programming: recent progress. *Discrete Applied Mathematics*, 123:129–154, 2002.

E. Balas and M. Perregaard. A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer gomory cuts for 0–1 programming. *Math. Program., Ser. B*, 94: 221–245, 2003.

E. Balas and A. Qualizza. Intersection cuts from multiple rows: A disjunctive programming approach. *EURO J. Comput. Optim.*, 1:3–49, 2013.

E. Balas and T. Serra. When lift-and-project cuts are different. Technical report, 2018.

E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Math. Program.*, 58:295–324, 1993.

E. Balas, S. Ceria, and G. Cornuéjols. Mixed 0–1 programming by lift-and-project in a branch-and-cut framework. *Manage. Sci.*, 42:1229–1246, 1996.

A. Basu, M. Conforti, G. Cornuéjols, and G. Zambelli. Maximal lattice-free convex sets in linear subspaces. *Mathematics of Operations Research*, 35(3):704–720, 2010.

A. Basu, P. Bonami, G. Cornujols, and F. Margot. Experiments with two-row cuts from degenerate tableaux. *INFORMS Journal on Computing*, 23(4):578–590, 2011.

A. Basu, M. Campêlo, M. Conforti, G. Cornuéjols, and G. Zambelli. Unique lifting of integer variables in minimal inequalities. *Mathematical Programming*, 141(1):561–576, 2013.

A. Basu, M. Conforti, and M. Di Summa. A geometric approach to cut-generating functions. *Mathematical Programming*, 151(1):153–189, 2015.

W. Ben-Ameur and J. Neto. Acceleration of cutting-plane and column generation algorithms:Applications to network design. *Networks*, 49(1):3–17, 2007.

T. Berger, Y. Sallez, S. Raileanu, C. Tahon, D. Trentesaux, and T. Borangiu. Personal rapid transit in an open-control framework. *Comput. Indust. Engrg.*, 61(2):300–312, 2011.

D. Bergman, A. A. Cire, W.-J. van Hoeve, and J. Hooker. *Decision Diagrams for Optimization.* Springer, 2016a.

D. Bergman, A. A. Ciré, W.-J. van Hoeve, and J. N. Hooker. Discrete optimization with binary decision diagrams. *INFORMS Journal on Computing*, 28:47–66, 2016b.

R. E. Bixby, E. A. Boyd, and R. R. Indovina. MIPLIB: A test set of mixed integer programming problems. *SIAM News*, 25:16, 1992.

R. E. Bixby, S. Ceria, C. M. McZeal, and M. W. P Savelsbergh. An updated mixed integer programming library: MIPLIB 3.0. *Optima*, (58):12–15, 1998.

P. H. Bly and R. Teychenne. Three financial and socio-economic assessments of a personal rapid transit system. In *Proc. 10th Internat. Conf. Automated People Movers*, pages 1–16, 2005.

P. Bonami. On optimizing over lift-and-project closures. *Mathematical Programming Computation*, 4:151–179, 2012.

V. Borozan and G. Cornuéjols. Minimal valid inequalities for integer constraints. *Mathematics of Operations Research*, 34(3):538–546, 2009.

J. Borwein and H. Wolkowicz. Regularizing the abstract convex program. *Journal of Mathematical Analysis and Applications*, 83(2):495 – 530, 1981.

J. Brake, J. D. Nelson, and S. Wright. Demand responsive transport: Towards the emergence of a new market segment. *J. Transport Geography*, 12(4):323–337, 2004.

R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35:677–691, 1986a.

R.E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986b.

C. Buchheim, F. Liers, and M. Oswald. Local cuts revisited. *Operations Research Letters*, 36: 430–433, 2008.

C. Buchheim, F. Liers, and M. Oswald. Speeding up IP-based algorithms for constrained quadratic 0-1 optimization. *Math. Program., Ser. B*, 124:513–535, 2010.

C. Buchheim, F. Liers, and L. Sanità. An exact algorithm for robust network design. In *Proceedings of INOC 2011*, pages 7–17. 2011.

F. Cadoux. Computing deep facet-defining disjunctive cuts for mixed-integer programming. *Mathematical Programming*, 122(2):197–223, 2010.

F. Cadoux and C. Lemaréchal. Reflections on generating (disjunctive) cuts. *EURO J. Comput. Optim.*, 1(1-2):51–69, 2013.

J. D. Camm. ASP, the art and science of practice: A (very) short course in suboptimization. *Interfaces*, 44(4):428–431, 2014.

S. Ceria and J. Soares. Disjunctive cuts for mixed $0-1$ programming: duality and lifting. *GSB, Columbia University*, 1997.

R. Chevrier, A. Liefooghe L. Jourdan, and C. Dhaenens. Solving a dial-a-ride problem with a hybrid evolutionary multi-objective approach: Application to demand responsive transport. *Appl. Soft Comput.*, 12(4):1247–1258, 2012.

COIN. COIN-OR (Common Infrastructure for Operations Research).
Available at `http://www.coin-or.org`.

M. Conforti and L. A. Wolsey. "Facet" separation with one linear program. *CORE DISCUSSION PAPER*, (2016/16), 2016.

M. Conforti, G. Cornuéjols, and G. Zambelli. A geometric perspective on lifting. *Operations Research*, 59(3):569–577, 2011a.

M. Conforti, G. Cornuéjols, and G. Zambelli. Corner polyhedron and intersection cuts. *Surveys in Operations Research and Management Science*, 16(2):105 – 120, 2011b.

J. F. Cordeau and G. Laporte. The dial-a-ride problem: Models and algorithms. *Ann. Oper. Res.*, 153(1):29–46, 2007.

G. Cornuéjols and F. Margot. On the facets of mixed integer programs with two integer variables and two constraints. *Mathematical Programming*, 120(2):429–456, 2008.

C. F. Daganzo. An approximate analytic model of many-to-many demand responsive transportation systems. *Transportation Res.*, 12(5):325–333, 1978.

E. Danna, M. Fenelon, Z. Gu, and R. Wunderling. Generating multiple solutions for mixed integer programming problems. In M. Fischetti and D. P. Williamson, editors, *Proceedings of IPCO*, pages 280–294. Springer, 2007.

S. Dash, S. S. Dey, and O. Günlük. Two dimensional lattice-free cuts and asymmetric disjunctions for mixed-integer polyhedra. *Mathematical Programming*, 135(1):221–254, 2012.

S. Dash, O. Gnlk, and J. P. Vielma. Computational experiments with cross and crooked cross cuts. *INFORMS Journal on Computing*, 26(4):780–797, 2014.

M. Dawande and J. N. Hooker. Inference-based sensitivity analysis for mixed integer/linear programming. *Operations Research*, 48:623–634, 2000.

S. S. Dey and L. A. Wolsey. Two row mixed-integer cuts via lifting. *Mathematical Programming*, 124(1):143–174, 2010.

S. S. Dey, A. Lodi, A. Tramontani, and L. A. Wolsey. On the practical strength of two-row tableau cuts. *INFORMS Journal on Computing*, 26(2):222–237, 2014.

M. Diana, M. M. Dessouky, and N. Xia. A model for the fleet sizing of demand responsive transportation services with time windows. *Transportation Res. Part B: Methodological*, 40(8):651–666, 2006.

D. G. Espinoza. Computing with multi-row Gomory cuts. *Operations Research Letters*, 38(2):115 – 120, 2010.

M. Fischetti and A. Lodi. Optimizing over the first Chvátal closure. *Mathematical Programming*, 110(1):3–20, 2007.

M. Fischetti and D. Salvagnin. An in-out approach to disjunctive optimization. In *Proceedings of CPAIOR*, pages 136–140. 2010.

M. Fischetti, A. Lodi, and A. Tramontani. On the separation of disjunctive cuts. *Mathematical Programming A*, 128:205–230, 2011.

Christodoulos A Floudas and Xiaoxia Lin. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Computers & Chemical Engineering*, 28(11):2109–2129, 2004.

R. Fukasawa, L. Poirrier, and Á. S. Xavier. The (not so) trivial lifting in two dimensions. Technical report, University of Waterloo, 2016.

G. Gamrath, B. Hiller, and J. Witzig. Reoptimization techniques for MIP solvers. In E. Bampis, editor, *Proceedings of SEA*, pages 181–192. Springer, 2015.

GAMS Support Wiki. Getting a list of best integer solutions of my MIP, 2013. URL `https://support.gams.com`.

A. M. Geoffrion and R. Nauss. Parametric and postoptimality analysis in integer linear programming. *Management Science*, 23(5):453–466, 1977.

F. Glover, A. Løkketangen, and D. L. Woodruff. An annotated bibliography for post-solution analysis in mixed integer programming and combinatorial optimization. In M. Laguna and J. L. González-Velarde, editors, *OR Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research*, pages 299–317. Kluwer, 2000.

A. Goetzendorff, M. Bichler, P. Shabalin, and R. W. Day. Compact bid languages and core pricing in large multi-item auctions. *Management Science*, 61(7):1684–1703, 2015.

R. E. Gomory. An algorithm for the mixed integer problem. Technical report, The Rand Corporation, 1960.

R.E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bull. Am. Math. Soc.*, 64:275–278, 1958.

P. Greistorfer, A. Løkketangen, S. Voß, and D. L. Woodruff. Experiments concerning sequential versus simultaneous maximization of objective function and distance. *Journal of Heuristics*, 14: 613–625, 2008.

T. Hadžić and J. N. Hooker. Postoptimality analysis for integer programming using binary decision diagrams. Technical report, Carnegie Mellon University, 2006a.

T. Hadžić and J. N. Hooker. Discrete global optimization with binary decision diagrams. In *Workshop on Global Optimization: Integrating Convexity, Optimization, Logic Programming, and Computational Algebraic Geometry (GICOLAG)*, Vienna, 2006b.

T. Hadžić and J. N. Hooker. Cost-bounded binary decision diagrams for 0–1 programming. In P. van Hentemryck and L. Wolsey, editors, *CPAIOR Proceedings*, volume 4510 of *Lecture Notes in Computer Science*, pages 332–345. Springer, 2007.

S. Hoda, W.-J. van Hoeve, and John N. Hooker. A systematic approach to MDD-based constraint programming. In *Proceedings of the 16th International Conference on Principles and Practices of Constraint Programming*, Lecture Notes in Computer Science. Springer, 2010.

S. Holm and D. Klein. Three methods for postoptimal analysis in integer linear programming. *Mathematical Programming Study*, 21:97–109, 1984.

M. E. Horn. Multi-modal and demand-responsive passenger transport systems: A modelling framework with embedded control systems. *Transportation Res. Part A: Policy Practice*, 36(2):167–188, 2002a.

M. E. Horn. Fleet scheduling and dispatching for demand responsive passenger services. *Transportation Res. Part C: Emerging Tech.*, 10(1):35–63, 2002b.

A. J. Hu. Techniques for efficient formal verification using binary decision diagrams. Thesis CS-TR-95-1561, Stanford University, Department of Computer Science, December 1996.

IBM Support. Using CPLEX to examine alternate optimal solutions, 2010. URL `http://www-01.ibm.com/support/docview.wss?uid=swg21399929`.

J. J. Jaw, A. R. Odoni, H. N. Psaraftis, and N. H. Wilson. A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows. *Transportation Res. Part B: Methodological*, 20(3):243–257, 1986.

F. Kilinç-Karzan, A. Toriello, S. Ahmed, G. Nemhauser, and M. Savelsbergh. Approximating the stability region for binary mixed-integer programs. *Operations Research Letters*, 37:250–254, 2009.

T. Kis. Lift-and-project for general two-term disjunctions. *Discrete Optimization*, 12:98 – 114, 2014.

J. Kronqvist, A. Lundell, and T. Westerlund. The extended supporting hyperplane algorithm for convex mixed-integer nonlinear programming. *J. Glob. Optim.*, 64:249–272, 2016.

C. Y. Lee. Representation of switching circuits by binary-decision programs. *Bell Systems Technical Journal*, 38:985–999, 1959.

J. D. Lees-Miller, J. C. Hammersley, and N. Davenport. Ride sharing in personal rapid transit capacity planning. In *12th Internat. Conf. Automated People Movers*, pages 321–332, 2009.

J. D. Lees-Miller, J. C. Hammersley, and R. E. Wilson. Theoretical maximum capacity as benchmark for empty vehicle redistribution in personal rapid transit. *Transportation Res. Record: J. Transportation Res. Board*, 2146(1):76–83, 2010.

H. Lei, G. Laporte, and B. Guo. Districting for routing with stochastic customers. *Eur. J. Transportation Logist.*, 1(1?2):67–85, 2012.

Y. Li and J.-P. P. Richard. Cook, Kannan and Schrijver's example revisited. *Discrete Optimization*, 5(4):724 – 734, 2008.

Z. Liu, X. Jiang, and W. Cheng. Solving in the last mile problem: Ensure the success of public bicycle system in beijing. *Procedia Soc. Behav. Sci.*, 43:73–78, 2012.

Q. Louveaux, L. Poirrier, and D. Salvagnin. The strength of multi-row models. *Mathematical Programming Computation*, 7(2):113–148, 2015.

J. Mageean and J. D. Nelson. The evaluation of demand responsive transport services in europe. *J.Transport Geography*, 11(4):255–270, 2003.

A. Mahéo, P. Kilby, and P. Van Hentenryck. Benders decomposition for the design of a hub and shuttle public transit system. *Transportation Science*, 2018.

E. Miller-Hooks and B. Yang. Updating paths in time-varying networks given arc weight changes. *Transportation Science*, 39:451–464, 2005.

K. Mueller and S. P. Sgouridis. Simulation-based analysis of personal rapid transit systems: Service and energy performance assessment of the masdar city prt case. *J. Adv. Transportation*, 45(4): 252–270, 2011.

K. Palmer, M. Dessouky, and T. Abdelmaguid. Impacts of management practices and advanced technologies on demand responsive transit systems. *Transportation Res. Part A: Policy Practice*, 38(7):495–509, 2004.

L. Quadrifoglio, M. M. Dessouky, and F. Ordóñez. A simulation study of demand responsive transit system design. *Transportation Res. Part A: Policy Practice*, 42(4):718?737, 2008.

A. Raghunathan, D. Bergman, J. Hooker, T. Serra, and S. Kobori. The integrated last-mile transportation problem (ilmtp). In *Proceedings, International Conference on Automated Planning and Scheduling (ICAPS) – to appear*, 2018.

L. Schrage and L. Wolsey. Sensitivity analysis for branch and bound integer programming. *Operations Research*, 33:1008–1023, 1985.

T. Serra. Reformulating the disjunctive cut generating linear program. Technical report, 2018.

T. Serra and J. Hooker. Compact representation of near-optimal integer programming solutions. Technical report, 2017.

S. Shaheen. U.S. carsharing & station car policy considerations: Monitoring growth, trends & overall impacts. Technical report, Institute of transportation studies, Working paper series, Institute of Transportation Studies, UC Davis, 2004.

Y. Shen, H. Zhang, and J. Zhao. Simulating the First Mile Service to Access Train Stations by Shared Autonomous Vehicle. In *Transportation Research Board 96th Annual Meeting*, 2017.

N. D. Thien. *Fair cost sharing auction mechanisms in last mile ridesharing.* PhD thesis, Singapore Management University, 2013.

C. Tjandraatmadja and W.-J. van Hoeve. Target cuts from relaxed decision diagrams. *Submitted*, 2016.

P. Toth and D. Vigo. *Vehicle Routing: Problems, Methods and Applications.* SIAM, second edition, 2014.

S. Van Hoesel and A. Wagelmans. On the complexity of postoptimality analysis of 0/1 programs. *Discrete Applied Mathematics*, 91:251–263, 1999.

A.F. Jr. Veinott. The supporting hyperplane method for unimodal programming. *Operations Research*, 15(1):147–152, 1967.

H. Wang. Routing and Scheduling for a Last-Mile Transportation Problem. *Transportation Science*, pages 1–17 (forthcoming), 2017. doi: https://doi.org/10.1287/trsc.2017.0753. URL `http://pubsonline.informs.org/doi/abs/10.1287/trsc.2017.0753`.

I. Wegener. *Branching Programs and Binary Decision Diagrams: Theory and Applications.* Society for Industrial and Applied Mathematics, 2000.

N. H. Wilson and C. Hendrickson. Performance models of flexibly routed transportation services. *Transportation Res. Part B: Methodological*, 14(1):67–78, 1980.