Resource Constrained Shortest Paths and Extensions

A Thesis
Presented to
The Academic Faculty

by

Renan Garcia

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Industrial and Systems Engineering
Georgia Institute of Technology
May 2009

Resource Constrained Shortest Paths and Extensions

Approved by:

Prof. Shabbir Ahmed, Co-Advisor
School of Industrial and Systems Engineering
Georgia Institute of Technology

Prof. George L. Nemhauser, Co-Advisor
School of Industrial and Systems Engineering
Georgia Institute of Technology

Prof. R. Gary Parker
School of Industrial and Systems Engineering
Georgia Institute of Technology

Prof. Martin W. P. Savelsbergh
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Zonghao Gu
Chief Technical Officer
Gurobi Optimization

Date Approved: January 7, 2009

ACKNOWLEDGEMENTS

First and foremost, I would like to thank God for all of the blessings and wonderful opportunities He has bestowed upon me in my journey through life. It is only by His grace that I have had the ability and strength to overcome life's challenges.

Next, I would like to express the deepest gratitude to my advisors, Dr. Shabbir Ahmed and Dr. George Nemhauser, for supporting me throughout my PhD research. Their encouragement and limitless patience, along with their guidance and knowledgeable insight, were instrumental in bringing this thesis to fruition. Working with them was an invaluable and most enjoyable experience, and it is impossible for me to put into words how much of a positive influence it has had on my life.

I would also like to express my sincere appreciation to the remaining members of my committee for their participation. I thank Dr. Gary Parker, who advised me in my first year at Georgia Tech, for his timely advice and for his willingness to help me with any problem I encountered. I learned a great deal from Dr. Martin Savelsbergh while working together on a research project for the DayJet Corporation, and I thank him for all of the stimulating discussions we had along the way. Thanks also to Dr. Zonghao Gu for taking the time out of his busy schedule to share some of his practical expertise with me.

During my studies, I was extremely fortunate to have interacted with a number of friends and colleagues at Georgia Tech. I offer special thanks to Deniz Dogan, Daniel Espinoza and Marcos Goycoolea for the countless times they helped me throughout the years and for being incredible friends. Thanks also to Faram Engineer, Yongpei Guan, Yetkin Ileri, Gizem Keysan, Seunghyun Kong, Jim Luedtke, Sriram Subramanian, Juan Pablo Vielma and all of the others not mentioned here for making the entire experience a lot of fun.

A significant portion of this thesis was written while I was working at the DayJet Corporation, and I thank Ed Iacobucci for giving me my first opportunity in industry before I had completed my PhD research. Thanks also to my manager at DayJet, Bob

Spaulding, for providing moral support as I finished the thesis while maintaining a full workload.

This would not be complete without the mention of my parents. I thank them for their unconditional love and support and for making the many sacrifices necessary for me to have this opportunity. Thanks also to my sister for her friendship and advice and for always helping me find humor in any situation. Le doy las gracias a mi abuelita Iraida por siempre preocuparse de mi, llamando me cada noche y mandando me pozuelitos de comida deliciosa por el correo. I apologize to my girlfriend Yari for all of the boring weekends we spent together while I was finishing up. I am so grateful to her for being a tremendous source of strength for me throughout this process and for all the love and encouragement she provided along the way. Last, but not least, I thank the rest of my family and friends in Miami who have always been there for me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

In this thesis, we use integer programming techniques to solve the resource constrained shortest path problem (RCSPP) which seeks a minimum cost path between two nodes in a directed graph subject to a finite set of resource constraints. Although $\mathcal{NP}$-hard, the RCSPP is extremely useful in practice and often appears as a subproblem in many decomposition schemes for difficult optimization problems.

We begin with a study of the RCSPP polytope for the single resource case and obtain several new valid inequality classes. Separation routines are provided, along with a polynomial time algorithm for constructing an auxiliary conflict graph which can be used to separate well known valid inequalities for the node packing polytope. We establish some facet defining conditions when the underlying graph is acyclic and develop a polynomial time sequential lifting algorithm which can be used to strengthen one of the inequality classes.

Next, we outline a branch-and-cut algorithm for the RCSPP. We present preprocessing techniques and branching schemes which lead to strengthened linear programming relaxations and balanced search trees, and the majority of the new inequality classes are generalized to consider multiple resources. We describe a primal heuristic scheme that uses fractional solutions, along with the current incumbent, to search for new feasible solutions throughout the branch-and-bound tree. A computational study is conducted to evaluate several implementation choices, and the results demonstrate that our algorithm outperforms the default branch-and-cut algorithm of a leading integer programming software package.

Finally, we consider the dial-a-flight problem (DAFP), a new vehicle routing problem that arises in the context of on-demand air transportation and is concerned with the scheduling of a set of travel requests for a single day of operations. The DAFP can be formulated as an integer multicommodity network flow model consisting of several RCSPPs linked together by set partitioning constraints which guarantee that all travel requests are satisfied.

Therefore, we extend our branch-and-cut algorithm for the RCSPP to solve the DAFP. Computational experiments with practical instances provided by the DayJet Corporation verify that the extended algorithm also outperforms the default branch-and-cut algorithm of a leading integer programming software package.

CHAPTER 1

INTRODUCTION

Mathematical programs are among the most widely used models in operations research. The goal in mathematical programming is optimization which involves maximizing (or minimizing) a real-valued function subject to a set of constraints. Any finite dimensional optimization problem can be written as

$$\max \{f(x) : x \in X\}, \tag{1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $X \subseteq \mathbb{R}^n$. The function f is called the *objective function*, the set X is called the *feasibility set*, and any $x \in X$ is called a *feasible solution*. If $x^* \in X$ is such that $f(x^*) \geq f(x)$ for all $x \in X$, then we say x^* is an *optimal solution* of (1.1) with *optimal value* $f(x^*)$. This formulation is extremely general and can be used to model a wide variety of practical problems.

Integer programming (IP) is a special case of (1.1) where f must be linear in x, and the feasibility set X is defined by a set of linear equalities and inequalities along with the additional restriction that any feasible solution must be integral. Using standard transformations, any IP problem can be expressed as

$$\max \left\{cx : Ax \leq b,\ x \in \mathbb{Z}^n_+\right\}, \tag{1.2}$$

where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. The entries in c, A and b are typically assumed to be integral. While this may seem somewhat restrictive, IP is an extremely flexible modeling framework that allows us to represent a huge number of real world applications in areas such as production planning, facility location, telecommunication and vehicle routing. For a more rigorous treatment of IP theory and applications, we suggest the books by Nemhauser and Wolsey [95], Wolsey [112], and Schrijver [103].

In this chapter, we review some of the classical approaches for solving IP problems such as branch-and-bound and cutting planes. Then we introduce basic graph theory notation

and review three fundamental network flow problems that frequently appear as subproblems in many algorithms. These problems can be modeled using IP, but as we will see, they possess structure that allows them to be solved more efficiently with specialized algorithms. Finally, we outline the remainder of the thesis.

1.1 Integer Programming Approaches

In complexity theory (see [49]), polynomial time solvability is the key criterion used to classify problems as either "easy" or "difficult". If an algorithm runs in *polynomial time*, then the number of basic operations performed should be bounded above by some function which is polynomial in the size of the input. We say a problem is in class $\mathcal{P}$ if there exists a polynomial time algorithm which can be used to solve it. We say a problem is in class $\mathcal{NP}$ if a proposed solution can be verified in polynomial time. Any problem in $\mathcal{P}$ is considered easy, and by definition, $\mathcal{P}$ is contained in $\mathcal{NP}$. A problem is said to be $\mathcal{NP}$-hard if it has the property that if it belongs in $\mathcal{P}$, then so does any problem in $\mathcal{NP}$. In a certain sense, $\mathcal{NP}$-hard problems are the most difficult problems in $\mathcal{NP}$. Although it remains an open question, it is the popular belief that $\mathcal{P} \neq \mathcal{NP}$ and there does not exist a polynomial time algorithm for any $\mathcal{NP}$-hard problem.

Unfortunately, Cook [19] showed that (1.2) is $\mathcal{NP}$-hard. In fact, Karp [76] showed that a special case of (1.2) known as *binary integer programming* (BIP), where the variable values are restricted to zero or one, is also $\mathcal{NP}$-hard. Despite these negative results, significant algorithmic progress has been made in the last few decades, and researchers are now able to solve large practical IP problems with thousands of variables and constraints. In this section, we describe some of the approaches used to solve such problems.

1.1.1 Branch-and-Bound

Using complete enumeration to solve IP problems is practically impossible since the number of possible solutions is exponential in the number of variables. To solve these problems, we must be more clever and use some form of implicit enumeration which eliminates the need to evaluate a large portion of the possible feasible solutions. One way to accomplish this is to decompose the feasibility set $X = \left\{x \in \mathbb{Z}_+^n : Ax \leq b\right\}$ into smaller subsets X^k such

that $X = \bigcup_{k \in K} X^k$, and then use bound information to avoid enumerating some of these subsets.

Let $z^* = \max \{cx : x \in X\}$. Clearly, $z^* = \max_{k \in K} \{z_k\}$, where $z_k = \max \left\{cx : x \in X^k\right\}$ for $k \in K$, and the objective value of the best known feasible solution, $\underline{z}$, provides a trivial lower bound for z^*. However, if we have an upper bound $\overline{z}_k$ on z_k for some $k \in K$, and $\overline{z}_k \leq \underline{z}$, then we can discard X^k in our search for the optimal solution since it cannot contain a solution that is better than the one associated with $\underline{z}$. Once we have discarded all such subsets (and potentially found a better solution), we can proceed by recursively decomposing any remaining subsets until all subsets have been discarded. It should be noted that if (1.2) was a minimization problem, the roles of the lower and upper bounds would be reversed.

This is the basic idea behind the *branch-and-bound* method which was first used to solve general IP problems to optimality by Land and Doig [86] in 1960. In fact, it remains at the heart of the algorithms used in modern commercial IP solvers like CPLEX [69] and XPRESS [27]. The effectiveness of these algorithms is dependent on rapid calculation of the upper bounds for each IP subproblem $\max \left\{cx : x \in X^k\right\}$ for $k \in K$, which is accomplished by relaxing their feasibility sets.

Typically, IP problems are relaxed by allowing fractional solutions. If we remove the integrality restriction from (1.2), we get the *linear programming* (LP) problem

$$\max \left\{cx : Ax \leq b,\ x \in \mathbb{R}^n_+\right\}. \tag{1.3}$$

Clearly, the optimal value of this LP relaxation provides an upper bound on z^*. Relaxing the integrality restriction also makes the problem much easier to solve. Khachiyan [80] showed that (1.3) is polynomially solvable using the ellipsoid method. However, this was only a theoretical breakthrough since the ellipsoid method does not perform well in practice. The *simplex* method, which was developed earlier by Dantzig [25], is the algorithm used to solve LP relaxations in most branch-and-bound implementations. While the simplex method is not a polynomial time algorithm, the number of iterations required for most practical problems is linear in the number of constraints. It also provides "warm start"

capabilities, or the ability to quickly reoptimize a problem that has been modified.

Therefore, if we choose our decompositions such that each of the subsets X^k is the feasibility set of a slightly altered IP problem, we can solve their LP relaxations quickly with the simplex method, and the optimal values will provide the upper bounds $\overline{z}_k$ for $k \in K$. Moreover, warm start should speed up the procedure if the consecutive subproblems are related.

The most commonly used decomposition scheme with this property is dichotomous and can be represented with a binary search tree known as the *branch-and-bound tree*. The search begins at the root node which corresponds to the original IP problem. After solving the LP relaxation, we get a solution x'. If $x' \in \mathbb{Z}_+^n$, then we have solved the problem. If not, then we *branch* on some fractional variable x'_j and create two new nodes in the tree. The subproblem in one node has the additional constraint $x_j \leq \lfloor x'_j \rfloor$, and the other $x_j \leq \lceil x'_j \rceil$. Note that both of these subproblems are also IP problems, and we can solve their LP relaxations. We continue branching from nodes until they are all *fathomed* due to one of the following three reasons:

1. *Optimality*: If the solution to the LP relaxation at a node is integral, then we have solved the subproblem. If this solution is better than the best known feasible solution to the original problem, also known as the *incumbent solution*, then we update.

2. *Bound*: If the optimal value of the LP relaxation at a node is less than or equal to the value of the incumbent solution, then the subproblem cannot yield a better solution.

3. *Infeasibility*: If the LP relaxation at a node is infeasible, then the subproblem is infeasible.

If we have an incumbent solution after all nodes have been fathomed, then it is optimal. Otherwise, we have fathomed all nodes by infeasibility and the original problem is infeasible. In many practical applications, it is acceptable to terminate the algorithm early if the incumbent is known to be within some tolerance of the optimal value.

If we increase the likelihood of fathoming by bound, we improve the method's performance by minimizing the number of nodes in the branch-and-bound tree that must be

evaluated. This can be accomplished by finding better bounds. Each time we fathom by optimality, we have potentially found a new incumbent and improved the lower bound given by the best feasible solution. To improve this bound, we can search for feasible solutions throughout the tree at nodes which do not necessarily yield integral solutions. A problem specific heuristic is typically used at each of these nodes to search for new solutions. The idea is to use the fractional solutions to guide the search with the hope that good solutions have similar values. To improve the upper bounds, we can improve the strength of our subproblem relaxations.

1.1.2 Cutting Planes

The feasibility set of any LP relaxation can be described by a finite set of linear inequalities, namely, those from the original IP formulation. Such a feasibility set is called a *polyhedron*, or *polytope* if it is bounded. One way to strengthen the relaxation is to add inequalities to the original formulation which do not remove any of the feasible integral solutions.

An inequality $\pi x \leq \pi_0$ is a *valid inequality* for $X \subseteq \mathbb{R}^n$ if $\pi x \leq \pi_0$ for all $x \in X$. Therefore, given $X = \left\{x \in \mathbb{Z}_+^n : Ax \leq b\right\}$ for some $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, we can add a valid inequality without altering the set of feasible solutions since $X = X \cap \{x \in \mathbb{R}^n : \pi x \leq \pi_0\}$. However, this inequality may alter the polyhedron $P = \left\{x \in \mathbb{R}_+^n : Ax \leq b\right\}$ corresponding to the LP relaxation of X. By definition, $P \supseteq P \cap \{x \in \mathbb{R}^n : \pi x \leq \pi_0\}$, but there might be fractional solutions in P that are violated by $\pi x \leq \pi_0$ (see Figure 1). This can lead to a tighter upper bound if $\max\{cx : x \in P,\ \pi x \leq \pi_0\} < \max\{cx : x \in P\}$ for some $c \in \mathbb{R}^n$.

If we know of a family $\mathcal{F}$ of valid inequalities for X, we can add them all to the formulation a priori. Care must be taken, however, since adding too many inequalities to the formulation can significantly increase the time necessary to solve the LP relaxation. In the worst possible case, there may be exponentially many inequalities in $\mathcal{F}$. Rather than adding all of them, we can search for only those that are useful. Valid inequalities in $\mathcal{F}$ that are violated by some fractional solution to the LP relaxation are called *cutting planes* or *cuts*. Given a fractional solution x^* to the LP relaxation, the search for a cutting plane $(\pi, \pi_0) \in \mathcal{F}$ such that $\pi x^* > \pi_0$ is known as the *separation problem* for $\mathcal{F}$. This naturally

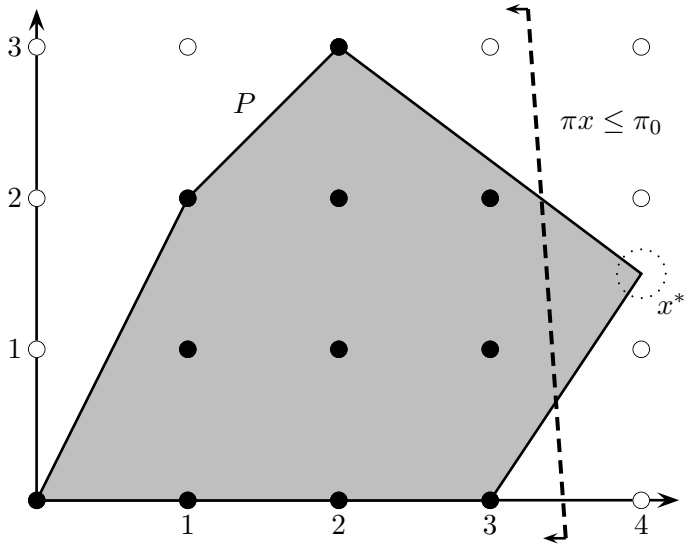

Figure 1: Fractional x^* violating a valid inequality for $X \subseteq \mathbb{Z}^2_+$

leads to the approach presented in Algorithm 1 which iteratively searches for many cutting planes.

Algorithm 1 Cutting plane algorithm for inequality family $\mathcal{F}$

1: $P_0 \leftarrow \left\{x \in \mathbb{R}^n_+ : Ax \leq b\right\}$, $t \leftarrow 0$
2: **loop**
3: let x^t be an optimal solution to $\max\{cx : x \in P_t\}$
4: **if** $x^t \in \mathbb{Z}^n$ **then**
5: **stop** x^t is an optimal solution to $\max\left\{cx : Ax \leq b,\ x \in \mathbb{Z}^n_+\right\}$
6: **end if**
7: solve the separation problem for $\mathcal{F}$ given x^t
8: **if** $\nexists\ (\pi^t, \pi^t_0) \in \mathcal{F}$ such that $\pi^t x^t > \pi^t_0$ **then**
9: **stop** $X' = P_t \cap \mathbb{Z}^n_+$ is an improved formulation
10: **end if**
11: $P_{t+1} \leftarrow P_t \cap \left\{x \in \mathbb{R}^n : \pi^t x \leq \pi^t_0\right\}$, $t \leftarrow t+1$
12: **end loop**

This algorithm searches for cuts violated by the optimal solutions to the continually improving LP relaxation. All of the work is done on lines 3 and 7. The linear program on line 3, with feasibility set P_t, can be solved using the simplex method. Fortunately, since the polyhedra P_{t-1} and P_t differ by a single inequality, x^{t-1} can be used to warm start the simplex method and solve $\max\{cx : x \in P_t\}$ with fewer iterations. Given a family $\mathcal{F}$, the separation problem on line 7 may require a lot of work. If the problem is $\mathcal{NP}$-hard,

the separation problem is just as hard as the original IP problem, and we might resort to separation heuristics which rapidly search a subset of $\mathcal{F}$ for violated inequalities.

Note that in solving the separation problem for $\mathcal{F}$, we could find several inequalities. In practice, it is often better to add more than one inequality at each iteration in a cutting plane algorithm. The hope is that by adding more inequalities, we converge to the optimal solution faster and limit the number of LP relaxations and separation problems that must be solved. Additional inequalities can also be found by solving the separation problem for multiple families at each iteration.

The first cutting plane algorithm reported was used to solve a 49-city Traveling Salesman Problem (TSP) by Dantzig, Fulkerson and Johnson [26]. The authors deduced some inequalities that must be satisfied by the optimal solution, and the optimal integral solution was obtained after adding these to the LP relaxation. Gomory [53] proposed Algorithm 1 with a family of cuts that are guaranteed to be violated if the solution to the LP relaxation is fractional. Therefore, the condition on line 8 is always false, and the algorithm always solves the original problem to optimality. While Gomory's fractional cuts are valid for any IP problem, the algorithm converges quite slowly in practice.

Even if Algorithm 1 is used with a family that does not guarantee an integral solution, it still terminates with an improved formulation (if at least one cut is found) which can be given to a branch-and-bound algorithm. This is known as the *cut-and-branch* method. The *branch-and-cut* method goes one step further and employs a cutting plane algorithm at various nodes throughout the branch-and-bound tree. The subproblem at each node in the tree is itself an IP problem with its own set of valid inequalities that are also valid for any of its children. If the generated cuts are valid for the original problem, then we can add them to the LP relaxation at every node in the tree. The first branch-and-cut algorithm reported was used to solve a linear ordering problem by Grötschel, Jünger and Reinelt [56].

1.1.3 Strong Valid Inequalities

The goal of any cutting plane algorithm is to add valid inequalities to the IP formulation so as to reduce the size of the polyhedron associated with the LP relaxation. Theoretically,

it is possible to characterize the smallest polyhedron which contains the feasibility set.

Definition 1.1. Given a set $X \subseteq \mathbb{R}^n$, the *convex hull* of X, denoted $\text{conv}(X)$, is defined as $\text{conv}(X) = \left\{x \in \mathbb{R}^n : x = \sum_{k=1}^{K} \lambda_k x^k,\ \sum_{k=1}^{K} \lambda_k = 1,\ \lambda_k \geq 0 \text{ for } k = 1, \ldots, K \text{ over all finite subsets } \{x^1, \ldots, x^K\} \text{ of } X\right\}$.

Proposition 1.1.1 (**See Wolsey [112]**)**.**

i. $\text{conv}(X)$ *is a polyhedron, and*

ii. the extreme points of $\text{conv}(X)$ *all lie in* X*.*

If $X = \left\{x \in \mathbb{Z}_+^n : Ax \leq b\right\}$, any valid inequality for X is implied by the inequalities that describe the polyhedron $\text{conv}(X)$. Hence, $\text{conv}(X)$ is the best formulation we could hope for after a cutting plane algorithm. Moreover, we know from LP theory that if $\max\{cx : x \in P\}$ has an optimal solution for some polyhedron P, then there exists an optimal solution which is an extreme point of P (see [95]). Therefore, we can solve the IP problem $\max\{cx : x \in X\}$ by solving the equivalent LP problem $\max\{cx : x \in \text{conv}(X)\}$. Unfortunately, this is only a theoretical result. In most cases, the number of inequalities needed to describe $\text{conv}(X)$ is exponential in the size of the input, and no simple characterization exists for them. Nevertheless, $\text{conv}(X)$ can be useful in practice. When selecting candidate valid inequalities for a cutting plane algorithm, polyhedral theory can be used in conjunction with $\text{conv}(X)$ to determine the strength of each inequality.

Definition 1.2. A set of points $x^1, \ldots, x^K \in \mathbb{R}^n$ is *affinely independent* if the unique solution of $\sum_{k=1}^{K} \lambda_k x^k = 0$, $\sum_{k=1}^{K} \lambda_k = 0$ is $\lambda_k = 0$ for $k = 1, \ldots, K$, or alternatively the $K - 1$ directions $x^2 - x^1, \ldots, x^K - x^1$ are linearly independent.

Definition 1.3. The *dimension* of a polyhedron P, denoted $\dim(P)$, is one less than the maximum number of affinely independent points in P.

Definition 1.4. The valid inequality $\pi x \leq \pi_0$ defines a *face* of P if $F = \{x \in P : \pi x = \pi_0\}$. A face F of P is *proper* if $F \neq \emptyset$ and $F \neq P$. A face F is a *facet* of P if $\dim(F) = \dim(P) - 1$.

Proposition 1.1.2 (**See Wolsey [112]**)**.** *If* $P \subseteq \mathbb{R}^n$ *is full-dimensional, a valid inequality is necessary in the description of* P *if and only if it defines a facet of* P*.*

Note that if P is not full-dimensional, $\dim(P) = n - k$ for some $k \in 1, \ldots, n$. This implies that P is contained in the subspace $\{x \in \mathbb{R}^n : A^= x = b^=\}$ for some $b^= \in \mathbb{R}^k$ and $A^= \in \mathbb{R}^{k \times n}$ with full row rank. A similar result holds for this case, except that each facet can be described by any inequality in an equivalence class of inequalities that is defined using linear combinations with $(A^=, b^=)$ (see [95]).

Therefore, the strength of a valid inequality is directly related to the dimension of the face of $\text{conv}(X)$ it defines, and the strongest valid inequalities for X define facets of $\text{conv}(X)$. However, it is possible to strengthen inequalities defining lower dimensional faces of $\text{conv}(X)$ through the notion of *sequential lifting*. The idea was introduced by Gomory [54] for the group problem and extended for general integer programs by Wolsey [110]. For simplicity, we restrict the feasibility set X to the unit cube $\mathbb{B}^n = \{0,1\}^n$, and illustrate lifting for the BIP case with the following result from Nemhauser and Wolsey [95].

Proposition 1.1.3. *Suppose $X \subseteq \mathbb{B}^n$, $X^\delta = X \cap \{x \in \mathbb{B}^n : x_1 = \delta\}$, and*

$$\sum_{j=2}^{n} \pi_j x_j \leq \pi_0 \tag{1.4}$$

is valid for X^0. If $X^1 = \emptyset$, then $x_1 \leq 0$ is valid for X. If $X^1 \neq \emptyset$, then

$$\alpha_1 x_1 + \sum_{j=2}^{n} \pi_j x_j \leq \pi_0 \tag{1.5}$$

is valid for X for any $\alpha_1 \leq \pi_0 - \zeta$, where $\zeta = \max\left\{\sum_{j=2}^{n} \pi_j x_j : x \in X^1\right\}$. Moreover, if $\alpha_1 = \pi_0 - \zeta$ and (1.4) gives a face of dimension k of $\text{conv}(X^0)$, then (1.5) gives a face of dimension at least $k+1$ of $\text{conv}(X)$.

When $\alpha_1 = \pi_0 - \zeta$, we say that the lifting is *maximum*, or we have lifted *exactly*. Note that if (1.4) is a facet of the lower dimensional space X^0 and lifting is maximum, the resulting inequality is a facet of $\text{conv}(X)$. If calculating ζ is $\mathcal{NP}$-hard, it is possible to use an upper bound on ζ which is easier to compute, and the inequality will still be valid. In this case, the lifting is said to be *approximate*. The hope is that if this bound is strong enough, the resulting inequality will be strong as well.

Proposition 1.1.3 is typically applied sequentially to lift a set of variables after projecting the feasibility set onto a lower dimensional space. The coefficients in the resulting inequality

depend on the order in which the variables are lifted. A better lifting coefficient for a given variable is obtained if the variable is lifted earlier in the sequence, with the maximum value of α_j obtained by lifting x_j first.

1.1.4 Lagrangean Relaxation

Many branch-and-bound algorithms fail because the bounds obtained from LP relaxations are too weak. Reformulation with cutting planes can help strengthen the LP relaxations, but there are problem classes for which strong valid inequalities have not been identified. Even when inequalities are available, they can be difficult to separate efficiently, or they may cause the solution of each LP subproblem to become computationally expensive. Alternatively, we can improve the bounds by switching to an entirely different relaxation for our subproblems.

Let us consider an IP problem of the form

$$z^* = \max \left\{ cx : Ax \leq b,\ Ex \leq f,\ x \in \mathbb{Z}^n_+ \right\}. \tag{1.6}$$

Assume that $Ax \leq b$ is a set of "complicating constraints", and problem (1.6) becomes much easier to solve if they are removed. The resulting bound, however, may be too weak. Instead, the constraints $Ax \leq b$ can be "dualized" by removing them from the constraint set and penalizing them in the objective function. If $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, we introduce the multipliers (dual variables) $\mu \in \mathbb{R}^m_+$ and formulate the *Lagrangean relaxation* subproblem

$$z(\mu) = \max \left\{ cx + \mu(b - Ax) : Ex \leq f,\ x \in \mathbb{Z}^n_+ \right\}. \tag{1.7}$$

The optimal value of problem (1.7), $z(\mu)$, is known as a *dual bound* since $z(\mu) \geq z^*$ for all $\mu \geq 0$. Since this IP problem is much easier to solve, $z(\mu)$ is typically computed with a rapid specialized algorithm (instead of branch-and-bound) which takes advantage of the structure inherent in $Ex \leq f$. To find the best dual bound, we can solve the *Lagrangean dual* problem

$$z_{LD} = \min \left\{ z(\mu) : \mu \in \mathbb{R}^m_+ \right\}. \tag{1.8}$$

Note that the *duality gap*, $z_{LD} - z^*$, is positive in most cases.

Problem (1.8) is often used as a relaxation for the IP subproblems in a branch-and-bound algorithm. Held and Karp [64] used the Lagrangean relaxation approach to solve

large TSPs. In their work, they used a subgradient algorithm to solve the Lagrangean dual problem. Since $z(\mu)$ is a piecewise linear convex function in μ, problem (1.8) can also be solved using Kelley's cutting plane method [79].

If the integrality restriction is removed from the original IP problem (1.6), we get the LP relaxation $z_{LP} = \max\left\{cx : Ax \leq b,\ Ex \leq f,\ x \in \mathbb{R}^n_+\right\}$. We have already observed that the optimal value of this LP, z_{LP}, is also a dual bound. The following result shows that the duality gap is at least as tight as the *integrality gap*, $z_{LP} - z^*$.

Proposition 1.1.4 (**See Wolsey [112]**). *If* $X = \left\{x \in \mathbb{Z}^n_+ : Ex \leq f\right\}$, *then*

$$z_{LD} = \max\left\{cx : Ax \leq b,\ x \in \text{conv}(X)\right\}.$$

By solving the Lagrangean relaxation subproblem (1.7) as an IP for all $\mu \in \mathbb{R}^m_+$, we have convexified the set $X = \left\{x \in \mathbb{Z}^n_+ : Ex \leq f\right\}$. Therefore, if $\text{conv}(X) = \left\{x \in \mathbb{R}^n_+ : Ex \leq f\right\}$, then $z_{LD} = z_{LP}$. Otherwise, the Lagrangean dual provides a stronger bound than the LP relaxation. Even if the bounds are equivalent, the Lagrangean dual may still provide an advantage over LP relaxation if the subproblems solve very quickly. This is typically the case if $\text{conv}(X) = \left\{x \in \mathbb{R}^n_+ : Ex \leq f\right\}$.

1.1.5 Column Generation

Decomposition approaches, such as Lagrangean relaxation, are some of the most useful ideas for solving difficult IP problems. These approaches typically lead to a master problem whose solution depends on the repeated solution of one or more subproblems. *Column generation* for LP problems, which was first used by Gilmore and Gomory [50, 51] to address a cutting stock problem, is another example of decomposition that has been vital in the solution of very large formulations.

Column generation is typically used when there are too many variables in the original formulation to handle efficiently. When used to solve IP problems, the master problem becomes the LP relaxation defined on a subset of the variables in the original formulation, with all other variables set to zero. After initially solving the master problem, we solve a subproblem known as the *pricing problem* which determines if any new variables need to be

added to the master problem. If new variables are added, the master problem is resolved, a new pricing problem is defined, and the process is repeated; otherwise, the master problem has been solved to optimality.

The *branch-and-price* method uses column generation to solve the LP relaxations in a branch-and-bound tree. The first use of branch-and-price was reported by Desrosiers et al. [32] for a routing problem. The philosophy is similar to that of branch-and-cut, but the focus is on variable generation rather than constraint generation. Actually, these two procedures are complementary when tightening an LP relaxation.

Although considering a formulation with a large number of variables may seem counterintuitive, there are several reasons why it may be useful:

1. A compact formulation of an IP problem may have a weak LP relaxation bound which might be improved by a reformulation involving a large number of variables.

2. A reformulation involving a large number of variables may destroy the inherent symmetry in a compact formulation that degrades the performance of branch-and-bound due to alternate choices after branching.

3. The decomposition into a master problem and pricing problem may allow for the incorporation of additional important constraints if the decomposition has a natural interpretation in the contextual setting.

4. It is the only formulation available.

There are a few subtle challenges that surface when using branch-and-price. Variable branching, which is the most common branching scheme used in branch-and-bound algorithms, may destroy the structure of the pricing problem. This might impede the use of specialized algorithms that use this structure to solve the problem efficiently. Hence, we are forced to use alternate branching schemes. Also, it is often inefficient to solve the master problem or the pricing problems to optimality. This requires different rules to be used when managing the branch-and-bound tree.

For an overview of branch-and-price, we refer the reader to the reports by Barnhart et al. [9] and Soumis [105].

1.2 Fundamental Network Flows

In this section we present three of the most fundamental optimization problems in graph theory. These problems, which are of interest in their own right, also arise throughout the literature (and this thesis) as subproblems in algorithms used to preprocess or solve other problems which are more difficult in nature. We now give a minimal introduction for each problem and mention some of the well known solution methods. It is important to note that all of these problems can be modeled as integer programs. However, the feasibility set X is equivalent to $\text{conv}(X)$ in each case, allowing them to be solved as linear programs. Before describing these problems, we give several basic definitions from graph theory and present the notation which will be used throughout the rest of this thesis. We use a modification of the notation used in the book by Ahuja et al. [1].

1.2.1 Definitions

A *directed graph* $G = (V, A)$ consists of a set V of *nodes* and a set A of *arcs* whose elements are ordered pairs of distinct nodes. Only graphs that are simple (no parallel arcs or loops) are considered. For each $S \subset V$, let us define the *out node adjacency list* of S as $V^+(S) = \{j \in V \setminus S : \exists \text{ an } (i,j) \in A \text{ for some } i \in S\}$, the *in node adjacency list* of S as $V^-(S) = \{i \in V \setminus S : \exists \text{ an } (i,j) \in A \text{ for some } j \in S\}$, the set of *outgoing arcs* of S as $\delta^+(S) = \{(i,j) \in A : i \in S,\ j \in V \setminus S\}$, and the set of *incoming arcs* of S as $\delta^-(S) = \{(i,j) \in A : j \in S,\ i \in V \setminus S\}$. We denote the set of arcs with both ends in S as $\gamma(S) = \{(i,j) \in A : i,j \in S\}$. For convenience, when S is a single node i, we will use i to denote S as the argument for these sets rather than $\{i\}$, e.g., we write $\delta^+(i)$ rather than $\delta^+(\{i\})$.

For $S \subset V$, a *cut* is a partition of the node set V into S and $\bar{S} = V \setminus S$. Any cut defines a *cutset* of arcs, denoted $(S : \bar{S})$, that have one endpoint in S and the other in $\bar{S}$, that is, $(S : \bar{S}) = \delta^+(S) \cup \delta^-(S)$. For two distinguished nodes s and t, an *s-t cut* is a cut such that $s \in S$ and $t \in \bar{S}$. A graph G is *connected* if $(S : \bar{S}) \neq \emptyset$ for all $S \subset V$.

A graph $G' = (V', A')$ is a *subgraph* of G if $V' \subseteq V$ and $A' \subseteq A$ such that any arc in A' has both endpoints in V'. For any subgraph $G' = (V', A')$, we denote the set of nodes in G' as $V(G')$ and the set of arcs of G' as $A(G')$. A *walk* W in G is a subgraph of G defined by a

sequence of nodes $(i_1, i_2, \ldots, i_p)$ such that $(i_k, i_{k+1}) \in A$ for all $k = 1, \ldots, p-1$. The set of arcs in W, $A(W)$, is precisely the set of arcs (i_k, i_{k+1}) which link the nodes in the sequence. We call W an *i-j walk* if the sequence begins with i and ends with j. To denote the set of all nodes that are reachable from $i \in V$, we define $\rho^+(i) = \{k \in V : \exists \text{ an } i\text{-}k \text{ walk in } G\}$. Similarly, $\rho^-(i) = \{k \in V : \exists \text{ a } k\text{-}i \text{ walk in } G\}$ is the set of all nodes from which i can be reached.

Remark 1. *For all $i, j \in V$,*

i. $i \in \rho^+(i)$ and $i \in \rho^-(i)$ (since $W = (i)$ is an i-i walk in G), and

ii. $j \in \rho^+(i)$ if and only if $i \in \rho^-(j)$.

A graph G is *s-t connected* if $\rho^+(s) = \rho^-(t) = V$ for distinct nodes s and t.

Remark 2. *If G is s-t connected for distinct nodes s and t, then G is connected.*

A *path* P in G is a walk which is open and elementary, i.e., $i_k \neq i_l$ for $1 \leq k < l \leq p$. A path Q in G is a *subpath* of P if Q is a connected subgraph of P. A *cycle* C in G is a path $P = (i_1, \ldots, i_p)$ together with the arc (i_p, i_1), that is, $V(C) = V(P)$ and $A(C) = A(P) \cup \{(i_p, i_1)\}$. We say G is *acyclic* if it does not contain any cycles.

Most of the time, we will associate numerical values such as costs, weights and capacities to arcs in A. Hence, we will have real vectors in $\mathbb{R}^A$ (typically integer) whose components are indexed by A. For any $B \subseteq A$ and vector $v \in \mathbb{R}^A$, we use the abbreviation $v(B)$ to denote $\sum_{(i,j)\in B} v_{ij}$. For convenience, when we want to sum over all arcs in a subgraph $G' = (V', A')$ such as a path, we will use $v(G')$ to denote this sum rather than $v(A(G'))$. Conversely, we say a subgraph $G' = (V', A')$ is the *support graph* of the incidence vector v if $A' = \{(i, j) \in A : v_{ij} \neq 0\}$ and V' is the set of all nodes in V that are incident to some $(i, j) \in A'$.

While most of the graphs in this thesis will be directed, we will also use the concept of an *undirected graph* which is defined in the same manner as a directed graph except that edges are unordered pairs of distinct nodes. Hence, (i, j) is the same as (j, i) in the undirected case. The definitions of the sets defined above follow naturally for the undirected

case with only a few modifications, e.g., ignoring the distinction between in and out arcs we have $\delta(S)$ rather than $\delta^+(S)$ and $\delta^-(S)$. As shown in [1], undirected graphs are a special case of directed graphs because any undirected graph can be converted to a directed one by replacing each undirected arc $\{i, j\}$ by two directed arcs (i, j) and (j, i).

1.2.2 Shortest Path Problem

Shortest path problems are among the most studied optimization problems in graph theory. They arise frequently in practice in a variety of settings and often appear as subproblems in many combinatorial optimization algorithms. The basic *shortest path problem* (SPP) tries to find the shortest path in a graph under some measure such as cost between a given set of origin/destination pairs. This problem can be solved efficiently in polynomial time as long as the graph does not contain a *negative cost cycle*, i.e., a cycle in which the sum of the arc costs is less than zero; otherwise, the problem becomes $\mathcal{NP}$-hard. This is because all algorithms used to solve SPP with negative arc costs in polynomial time find a shortest walk rather than a shortest path. Since we can traverse this cycle an infinite number of times, we must add the restriction that nodes cannot be revisited which complicates the problem. While most of the algorithms detect negative cost cycles, we will focus our attention on the case where no such cycles are present in the graph.

Given a directed graph $G = (V, A)$, a source node $s \in V$, a sink node $t \in V$ and arc costs $c_{ij} \in \mathbb{R}$, we can formulate the SPP as a BIP:

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} c_{ij} x_{ij} \\
\text{s.t.} \quad & x(\delta^+(i)) - x(\delta^-(i)) = \begin{cases} 1, & i = s \\ -1, & i = t \\ 0, & \forall\, i \in V \setminus \{s, t\} \end{cases} \\
& x_{ij} \in \{0, 1\}\,,\ \forall\, (i, j) \in A.
\end{aligned}
$$

Any s-t path in G is equivalent to a (directed) flow of one unit from s to t through the arcs in G. The set of equalities in this formulation are known as *flow balance constraints*. These constraints ensure that the flow into a node is equal to the flow out of that node for all

nodes in the graph other than s and t. In the case of the source, the flow out of the node must be exactly one unit greater than the flow into the node. Similarly, in the case of the sink, the flow into the node must be exactly one unit greater than the flow out of the node.

This IP formulation is a special case of the *network flow problem* where units of flow are sent across the arcs in a capacitated graph from supply nodes to demand nodes. Each arc has an associated transportation cost and capacity, and the objective is to send the units at a minimal cost while respecting capacity limitations. All network flows are IPs (fractional flow is not allowed) with a constraint matrix that is always totally unimodular, implying that the feasibility of the LP relaxation is equivalent to its convex hull if the right hand side coefficients and the variable bounds are integral (see [112]). Therefore, we can remove the integrality restriction and solve the problem using an LP algorithm. In fact, specialized network simplex algorithms have been developed that utilize this structure. However, since the simplex method is not a polynomial time algorithm, researchers rely on faster algorithms for solving the SPP.

Efficient algorithms for the SPP are classified under two approaches: label-setting and label-correcting. Label-setting algorithms work for graphs with non-negative arc costs and identify a permanent distance label for one or more nodes at each iteration. The first such algorithm was developed by Dijkstra [33], with a worst case running time of $\mathcal{O}(|V|^2)$. Note that a function $f(x)$ is said to be $\mathcal{O}(g(x))$ if there exists a constant K such that $f(x) \leq Kg(x)$ for all possible x in the domain. Many variations of Dijkstra's algorithm have been proposed to improve this running time, including one that uses Fibonacci heaps and runs in $\mathcal{O}(|A| + |V| \log |V|)$ [47]. In the special case when G is acyclic, we can use an algorithm which finds a topological ordering of the nodes in V and assigns permanent labels in that order (see [1]). This algorithm is valid even when the graph has negative arc costs and has a running time of $\mathcal{O}(|A|)$. Label-correcting algorithms allow negative arc costs even if G is cyclic and maintain distance labels for each node which are iteratively updated until they satisfy optimality conditions. The first label correcting algorithm was by Ford [45]. Bellman's algorithm [12] can also be viewed as a label-correcting algorithm and runs in $\mathcal{O}(|V||A|)$.

Any shortest s-i_p path $(s, i_1, \ldots, i_p)$ in G has the nice property that the subpath $(s, i_1, \ldots, i_k)$ is a shortest s-i_k path in G for all $k = 1, \ldots, p-1$. In fact, both label-setting and label-correcting algorithms automatically find the shortest path from a single source node to all other nodes in the graph. The *all-pairs shortest path problem* attempts to find the shortest paths between every pair of nodes in the graph. The Floyd-Warshall algorithm [44] solves the all-pairs shortest path problem in $\mathcal{O}(|V|^3)$ using matrix algebra. We can also find the all-pairs shortest paths by iteratively solving the basic shortest path problem for each node in the graph. This results in a total run time of $\mathcal{O}(|V|\mathcal{SP}(V, A))$, where $\mathcal{SP}(V, A)$ is the polynomial time bound on the complexity of solving the SPP on a graph without negative cost cycles. To the best of our knowledge, $\mathcal{SP}(V, A)$ is equal to $|A|$ if G is acyclic, $|A| + |V| \log |V|$ if G is cyclic with non-negative arc costs, and $|V||A|$ if G is cyclic with negative arc costs.

A comprehensive treatment of the SPP can be found in the book by Ahuja et al. [1]. Cherkassky et al. [17] provide a detailed survey about the performance of several algorithms used to solve the problem. For an overview of topological orderings, we recommend the book by Cormen et al. [21].

1.2.3 Maximum Flow Problem and Minimum Cut Problem

We now review two other fundamental problems in graph theory that appear in many combinatorial algorithms. The *maximum flow problem* tries to send as much flow as possible between two nodes in a capacitated graph without exceeding the capacity of any arc. Like SPP, the maximum flow problem is a special case of the network flow problem (with no costs). Given a directed graph $G = (V, A)$, a source node $s \in V$, a sink node $t \in V$ and arc capacities $u_{ij} \in \mathbb{R}_+$, we can formulate the problem as an LP:

$$
\begin{aligned}
\max \quad & f \\
\text{s.t.} \quad & x(\delta^+(i)) - x(\delta^-(i)) = \begin{cases} f, & i = s \\ -f, & i = t \\ 0, & \forall\ i \in V \setminus \{s, t\} \end{cases} \\
& 0 \leq x_{ij} \leq u_{ij},\ \forall\ (i, j) \in A.
\end{aligned}
$$

As mentioned above, this LP formulation gives an integral solution if the arc capacities are integral. Of course, if the graph contains an s-t path with infinite capacity, the optimal value is unbounded.

Efficient algorithms for the maximum flow problem are classified under two approaches: augmenting path algorithms and preflow-push algorithms. Augmenting path algorithms iteratively augment flow along s-t paths in G while respecting the flow balance constraints. Ford and Fulkerson [46] developed the first such algorithm which had a worst case running time of $\mathcal{O}(|V||A|U)$ if all capacities are integral, where $U = \max_{(i,j)\in A}\{u_{ij}\}$. Note that this bound is not necessarily polynomial in the size of the problem input since U can be exponential in $|V|$ or $|A|$. When a run time bound is linear (or greater) in some numerical value in the problem input, we say the algorithm runs in *pseudopolynomial time*. Preflow-push algorithms begin by allowing some nodes to have excess flow, but then iteratively remove this excess by sending flow toward the source or the sink. Several implementations of preflow-push algorithms have been suggested which run in $\mathcal{O}(|V|^3)$ time [52, 77, 104].

The same input for the maximum flow problem gives rise to a related problem in graph theory. For any s-t cut S in G, we say the capacity of S is equal to $\sum_{(i,j)\in\delta^+(S)} u_{ij}$. The *minimum cut problem* tries to identify the s-t cut in G with the minimum capacity. Clearly, the capacity of any s-t cut in G provides an upper bound on the maximum flow between s and t. The following theorem gives an important relationship between the maximum flow and the minimum cut in G.

Theorem 1.2.1 (**Max-Flow Min-Cut Theorem [46]**)**.** *The maximum value of the flow from a source node s to a sink node t in a capacitated graph equals the minimum capacity among all s-t cuts.*

This result implies that each time we solve a maximum flow problem, we also solve a complementary minimum cut problem. Therefore, we can also solve the minimum cut problem in $\mathcal{O}(|V|^3)$. To convert a maximum flow solution x^* to an s-t cut S, we can remove any arc from the graph such that $x^*_{ij} = u_{ij}$, and then perform a breadth-first search from the source in $\mathcal{O}(|A|)$. Any node that is reached belongs in S.

The Max-Flow Min-Cut theorem is actually a special case of the well-known strong duality theorem of LP (see [95]). Given any LP problem, there exists a dual problem that provides an upper bound (in the case of maximization) on the optimal value. This problem can be derived from the Lagrangean dual problem defined above for IP problems. In the case of LP, however, there is no duality gap. These problems form a strong-dual pair and share the same optimal value. Hence, we can solve one of the problems by solving the other. The dual for the maximum flow problem is the LP formulation for the minimum cut problem and can be expressed as

$$\begin{aligned} \min \quad & \sum_{(i,j)\in A} u_{ij} z_{ij} \\ \text{s.t.} \quad & -y_i + y_j + z_{ij} \geq 0, \ \forall \ (i,j) \in A \\ & y_s = 1, \ y_t = 0 \\ & z_{ij} \geq 0, \ \forall \ (i,j) \in A. \end{aligned}$$

Ahuja et al. also give a more detailed treatment for both the maximum flow problem and the minimum cut problem, along with several other algorithms used to solve the maximum flow problem. For a presentation of graph searching algorithms like breadth-first search, we refer the reader to the book by Cormen et al. [21].

1.3 Thesis Outline

In the remainder of this thesis, we study the resource constrained shortest path problem (RCSPP). This problem is a generalization of the SPP where feasible paths must also satisfy a set of knapsack inequalities. Although $\mathcal{NP}$-hard, the RCSPP is extremely useful in practice and often appears as a subproblem in many decomposition schemes for difficult optimization problems. It also appears as a substructure in many problems, including the dial-a-flight problem (DAFP) which can be modeled as several resource constrained shortest paths linked by side constraints. We extend the approaches developed for the RCSPP to help solve the DAFP more efficiently.

In Chapter 2, we define the RCSPP and review some of the existing approaches for solving the problem. Then we formulate the problem using IP and motivate the need for

polyhedral analysis along with a branch-and-cut approach.

In Chapter 3, we introduce several valid inequalities for the weight constrained shortest path problem (WCSPP) which is the special case of the RCSPP with a single knapsack inequality. This problem can be viewed as a relaxation for the RCSPP which can be used to find valid inequalities. We provide separation routines for each of the inequalities along with their associated complexity bounds.

Chapter 4 examines the WCSPP when the underlying graph is acyclic. We begin with a result that defines the subgraph induced by all s-t paths that contain a given subpath. Next, we introduce a new valid inequality along with two heuristic separation routines. Then, we show that this inequality is facet defining for a lower dimensional relaxation, and a polynomial time sequential lifting algorithm is introduced to get facets of the full dimensional relaxation. Finally, we give the dimension of the acyclic WCSPP polytope, and show that one of the inequalities defined in Chapter 3 is facet defining for a projection of the polytope onto a specific subspace.

In Chapter 5, we outline a branch-and-cut algorithm for the RCSPP. We begin by enhancing three general IP preprocessing techniques for the acyclic case. Next we summarize all of the cutting planes to be used in the algorithm, including cycle elimination inequalities that are required when the graph has negative cost cycles, and we extend the valid inequalities obtained for the WCSPP to consider multiple resources. Then, we discuss alternative branching strategies, some of which lead to branch-and-bound trees that are more balanced. Finally, we describe primal heuristic schemes for use throughout the branch-and-bound tree.

The branch-and-cut algorithm is compared with a leading commercial IP solver in Chapter 6. We measure the efficacy of all of the elements of the algorithm introduced in the previous chapter. The computational results provided show that the algorithm significantly improves upon a generic branch-and-cut implementation.

In Chapter 7, we generalize elements of the RCSPP branch-and-cut algorithm for the DAFP. The resulting algorithm is tested with practical problems provided by the DayJet Corporation, and we demonstrate that it also improves upon a generic branch-and-cut implementation.

We conclude with Chapter 8 and present the main contributions of the thesis along with some suggestions for future research.

CHAPTER 2

THE RESOURCE CONSTRAINED SHORTEST PATH PROBLEM

2.1 Introduction

Many different generalizations of the shortest path problem (SPP) have been studied in the literature. One of the most common variants is the *weight constrained shortest path problem* (WCSPP). Given a directed graph which has a cost and a non-negative weight associated with each arc, WCSPP is the problem of finding a minimum cost path between two given nodes such that the total weight used along the path does not exceed some capacity. The WCSPP is known to be $\mathcal{NP}$-hard, even if we restrict ourselves to the case where the graph is acyclic and all costs are positive [49]. If the graph is free of negative cost cycles, the problem is $\mathcal{NP}$-hard in the weak sense and can be solved in pseudopolynomial time; otherwise, the problem is strongly $\mathcal{NP}$-hard (simple transformation from the SPP with negative cost cycles).

The *resource constrained shortest path problem* (RCSPP) is a generalization of the WCSPP where we have a finite set of resources, each with an associated capacity, and the arcs have a weight associated with each resource. Like the WCSPP, the RCSPP seeks a minimum cost path between two given nodes such that the capacity for all of the resources is not exceeded along the path. For a fixed number of resources, the RCSPP without negative cost cycles is also $\mathcal{NP}$-hard in the weak sense and can be solved in pseudopolynomial time by extending label setting algorithms for the WCSPP [35]. Some researchers allow minimum cost walks in their definition of the RCSPP. To make the distinction, they refer to the problem which allows only minimum cost paths as the "elementary" resource constrained shortest path problem. In this thesis, we are always interested in elementary paths and walks are never allowed.

The RCSPP will be the primary focus of this thesis, and we now define the problem formally. Let $G = (V, A)$ be a directed graph with a distinguished source node s and sink node t. We assume there are R resources, with capacities $W_r \in \mathbb{Z}_+$ for each resource $r = 1, \ldots, R$. With each arc $(i, j) \in A$, we associate an arc cost $c_{ij} \in \mathbb{Z}$ and a non-negative arc weight $w_{ij}^r \in \mathbb{Z}_+$ for each resource r. For any path P in G, let $c(P) = \sum_{(i,j)\in A(P)} c_{ij}$ and $w^r(P) = \sum_{(i,j)\in A(P)} w_{ij}^r$. We say P is *weight feasible* if $w^r(P) \leq W_r$ for each resource $r = 1, \ldots, R$. The RCSPP searches for a weight feasible s-t path P in G that minimizes $c(P)$. Note that with a simple modification, this model also allows node related costs and weights. This can be captured by adding the quantities consumed on a node i to all of the arcs in $\delta^-(i)$.

Of course, the most obvious application of the RCSPP is in the transportation domain. In this setting, we would like to find the quickest path between an origin and destination with budget constraints on mileage, fuel consumption and tolls. Besides this obvious example, the problem has many practical applications in diverse areas including railroad management [60], military aircraft systems [115], quality of service routing in communication networks [113], waste water treatment [36] and linear curve approximation [96].

The RCSPP is also related to the *multi-objective shortest path problem* (MOSPP), which seeks all Pareto optimal solutions to a shortest path problem with multiple objectives. For any RCSPP, there exists an optimal solution that is also Pareto optimal for the MOSPP with objectives defined by the arc costs and arc weights. Therefore, any method that generates all Pareto optimal solutions for the MOSPP also generates an optimal solution to the RCSPP. However, this approach is unlikely to be as efficient because there may be a considerable number of Pareto optimal solutions that are not weight feasible which can be discarded.

An additive notion of path feasibility is used in the RCSPP, but researchers have also considered other, more general, resource constraints. Pallottino and Scutellà [98] survey several interesting complications in the context of transportation such as time windows. In this case, each arc has an associated time duration, and each node i has an associated time window, $[a_i, b_i]$, requiring that processing at that node must take place no earlier than a_i

and no later than b_i. This problem is known as the *shortest path problem with time windows* (SPPTW). The time window constraints are more complicated than a simple weight constraint since waiting time is allowed before processing begins at each node. However, the WCSPP is a special case of the SPPTW where $a_i = 0$ and $b_i = W$ for all $i \in V$, and algorithms for the WCSPP can be extended to the case with time windows.

The RCSPP and variants such as the SPPTW, also appear frequently as a subproblem in decomposition schemes for difficult optimization problems. Many column generation approaches for solving vehicle routing and crew scheduling problems tend to have pricing problems which are RCSPP variants (see for example [8, 16, 48, 83, 108]). In this context, the subproblems may be solved many times, and the run time of the algorithm used to solve them is critical. Irnich and Desaulniers [70] review some of the common variants in that application domain and propose a classification scheme along with a generic formulation.

It should be noted that many of these column generation applications impose negative costs on some of the arcs. This is not an issue if the underlying graph is acyclic, which is the case for some time expanded formulations. However, the subproblems become strongly $\mathcal{NP}$-hard when negative cost cycles are possible. While some researchers have attempted to forbid some or all of the negative cycles in the graph, the most common approach is to allow non-elementary paths and search for minimum cost walks. This can be done in pseudopolynomial time, but it often results in very weak linear programming (LP) relaxations in a branch-and-price algorithm [71].

The RCSPP is also a common substructure in difficult optimization problems. For example, Avella et al. [7] discuss a generalization of the RCSPP used in fleet management. The authors consider an RCSPP with various additional constraint types that cannot be expressed as resource constraints with non-negative coefficients. In another example, Espinoza et al. [39, 40] study the dial-a-flight problem. The authors present an integer multicommodity network flow formulation with side constraints which is essentially several RCSPPs (one for each jet) linked together by equalities that force all customer requests to be satisfied.

In the remainder of this chapter, we review previous work for solving the RCSPP. Then

we formulate the problem as a binary integer program (BIP) and motivate the need for polyhedral analysis along with a branch-and-cut approach.

2.2 Previous Work

Several exact and approximate methods have been proposed to solve the RCSPP. We present an overview of these methods in this section, classifying them into several categories including those based on path ranking, node labeling and Lagrangean relaxation, although some methods may combine elements of more than one category. Many of these algorithms do not explicitly forbid cycles, and may find minimum cost walks (instead of paths) if negative arc costs are present. Approaches that guarantee elementary paths are also presented in a separate category. Finally, we end the section with a review of fully polynomial approximation schemes used to solve the WCSPP.

2.2.1 Path Ranking Methods

Path ranking, often known as the K *shortest path problem*, seeks the K (unconstrained) shortest paths in a graph. This problem was first studied by Hoffman and Pavley [67]. Since then, several methods have been proposed (see for example [37, 73, 78, 91, 114]). Most of the methods have polynomial time complexity for fixed K. Eppstein [37] describes an algorithm with complexity $\mathcal{O}(|A| + |V| \log |V| + K)$ that solves the problem when cycles are allowed. When used to solve the RCSPP, however, K is ignored and paths are enumerated in non-decreasing order of cost until the first weight feasible path is found, resulting in an exponential time algorithm. While this discourages the use of path ranking to solve the RCSPP directly, it has been used successfully as a subroutine in more complicated approaches [61, 92]. An extensive bibliography for the K shortest path problem can be found on the website of Eppstein [38].

2.2.2 Node Labeling Methods

Dynamic programming (DP) is a recursive technique used to solve optimization problems that involve a sequence of decisions. The key to solving a problem by DP is to break the problem down into stages at which decisions take place, and then define a recurrence

relation that describes the optimal value of the current stage given that we have solved the previous stage.

Initial DP formulations for the WCSPP were given by Joksch [74] and Lawler [88]. After that, several node labeling algorithms for the RCSPP and variants such as the SPPTW, have been developed based on these formulations (see for example [2, 29, 31, 35, 72]). At any node i in the graph, these algorithms extend only non-dominated labels corresponding to s-i subpaths which are weight feasible. The label setting algorithm of Desrochers and Soumis [29], which was originally stated for the SPPTW, achieves a complexity bound of $\mathcal{O}(|A|W)$ for the WCSPP if all of the arc weights are positive. Dumitrescu and Boland [35] present a modified version of this algorithm which utilizes preprocessing and Lagrange multiplier information, and they claim the bound is $\mathcal{O}(|V|^2W)$ if zero weights are allowed. Since the complexity is pseudopolynomial, the algorithm in [29] is frequently the algorithm of choice for solving the WCSPP. A nice introduction to these methods can be found in [30].

2.2.3 Lagrangean Relaxation Methods

Many researchers have turned to Lagrangean relaxation as a viable alternative for solving the RCSPP (see for example [11, 61, 92, 101]). If the resource constraints are relaxed, the subproblem becomes an instance of the SPP which can be solved repeatedly with minimal computational effort. These methods typically consist of three different steps: (1) compute lower and upper bounds for the optimal value of the RCSPP by solving the relaxation, (2) use these bounds to reduce the graph, and (3) close the duality gap.

Handler and Zang [61] address the WCSPP problem by specializing Kelley's cutting plane method to solve the dual problem, and they close the duality gap with the K shortest path algorithm of Yen [114]. Beasley and Christofides [11] solve the dual of the RCSPP with subgradient optimization, and apply a binary depth-first tree search procedure to close the duality gap. Mehlhorn and Ziegelmann [92] use a slightly different formulation for the RCSPP and solve a similar dual using a hull approach. The duality gap is closed using one of three methods: (1) the algorithm of Hassin [63], (2) the K shortest path algorithm of

Yen [114], and (3) a dynamic programming algorithm that prunes subpaths which cannot be part of the optimal solution.

Relaxation has also been used in heuristic schemes. Ribeiro and Minoux [101] propose such an approach for a generalization of the WCSPP with upper and lower resource limits. The approach is based on Lagrangean relaxation, and it follows from their work that the Lagrangean dual can be solved in $\mathcal{O}(|V||A|W)$ using a parametric approach [34]. Avella et al. [6] relax the resource constraints for the RCSPP into an exponential penalty function and solve the problem approximately using a constrained steepest descent algorithm.

2.2.4 Approaches that Guarantee Elementary Paths

In the presence of negative cost cycles, many researchers solve a relaxed version of the RCSPP, or variants such as the SPPTW, which searches for minimum cost walks. This relaxed problem is typically solved with one of the pseudopolynomial node labeling methods discussed above. This has been a successful approach in a number of column generation algorithms (see for example [28, 55, 87]). However, this approach may lead to weak LP relaxations and prohibitively large branch-and-price trees [71].

Very little work has been done which guarantees elementary paths. Some authors compromise by only allowing cycles of length greater than K for some small K. Houck et al. [68] and Christofides et al. [18] first addressed this idea, with $K = 2$, for shortest path algorithms. Subpaths of the form (i, j, i) are forbidden with a simple constraint which ensures that the predecessor of a node along any path is not equal to its successor. Kolen et al. [84] and Desrochers et al. [28] applied this idea to help solve vehicle routing problems. Recently, Irnich and Villeneuve [71] used a node labeling method with new dominance rules to extend this for arbitrary K, and improved the lower bounds for some previously unsolved vehicle routing problems.

Beasley and Christofides [11] suggested adding an additional binary resource for each node in the graph. Each time a path traverses a node, the resource would be consumed, and the path would not be allowed to visit the node again. Unfortunately, this problem forces an exponential run time for the node labeling approaches discussed above, and the authors

abandoned the idea. Kohl [83] pursued the concept of node resources, and suggested a node labeling algorithm with a state space relaxation approach. Only the resources for a subset of the nodes are added to the problem. After solving the relaxation, a new node resource is added for some node that was visited more than once. This is repeated until there are no cycles in the solution to the relaxation. The approach was also accelerated with the observation that nodes which can never appear on a path together only require a single resource. Boland et al. [14] implemented the ideas in [83] and tested several strategies for augmenting the set of node resources. Feillet et al. [41] also considered node resources. In their approach, however, strong dominance is used in a node labeling algorithm with a full-dimensional state space.

2.2.5 Approximation Schemes

Given $\epsilon > 0$, an algorithm is called an *ϵ-approximation algorithm* for a minimization problem if it provides a feasible solution with value z_ϵ such that

$$z^* \leq z_\epsilon \leq (1+\epsilon)z^*,$$

where z^* is the optimal value of the instance. We say a class of algorithms (parameterized by ϵ) is a *fully polynomial approximation scheme* (FPAS) if there exists an ϵ-approximation algorithm for each $\epsilon > 0$ with complexity that is polynomial in the size of the input and $\frac{1}{\epsilon}$. Since the WCSPP is $\mathcal{NP}$-hard in the weak sense if negative cost cycles are not present, it admits an FPAS.

One of the most common approaches for deriving an FPAS for weakly $\mathcal{NP}$-hard problems is *cost scaling and rounding*. Hansen [62] used this approach to develop an FPAS for the MOSPP with only two objectives (the bicriteria shortest path problem). Warburton [109] took a similar approach for the MOSPP, and also described an application to the WCSPP.

The first FPAS applied directly to the WCSPP was proposed by Hassin [63]. The author used cost scaling and rounding to develop two versions of an FPAS with respective complexities of $\mathcal{O}(\log\log U(|A|\frac{|V|}{\epsilon} + \log\log U))$ and $\mathcal{O}(|A|\frac{|V|^2}{\epsilon}(\log\frac{|V|}{\epsilon}))$, where U is an upper bound on the cost of the optimal path. Phillips [100] also proposed an FPAS for the WCSPP with complexity $\mathcal{O}(|A|\frac{|V|}{\epsilon} + \frac{|V|^2}{\epsilon}(\log\frac{|V|}{\epsilon}))$. Lorenz and Raz [90] used cost scaling and

rounding to propose an FPAS for the WCSPP with complexity $\mathcal{O}(\frac{|A||V|}{\epsilon} + |A||V| \log\log \frac{U}{L})$, where L is a lower bound on the cost of the optimal path. Dumitrescu and Boland [35] proposed minor modifications to the first algorithm in [63] which improved its performance in practice.

2.3 Motivating a Branch-and-Cut Approach

An alternative approach to those discussed above is to use LP based branch-and-bound. Let us define the incidence vector $x \in \mathbb{R}_+^A$ as follows: $x_{ij} = 1$ if $(i,j) \in A$ is used in the optimal s-t path in G; $x_{ij} = 0$ otherwise. Then we can formulate the RCSPP as a BIP:

$$\min \quad \sum_{(i,j)\in A} c_{ij} x_{ij}$$

$$\text{s.t.} \quad x(\delta^+(i)) - x(\delta^-(i)) = \begin{cases} 1, & i = s \\ -1, & i = t \\ 0, & \forall\ i \in V \setminus \{s,t\} \end{cases} \tag{2.1}$$

$$x(\gamma(S)) \leq |S| - 1,\ \forall\ S \subset V,\ 2 \leq |S| \leq |V| - 2 \tag{2.2}$$

$$\sum_{(i,j)\in A} w_{ij}^r x_{ij} \leq W_r,\ \forall\ r = 1, \ldots, R \tag{2.3}$$

$$x_{ij} \in \{0,1\},\ \forall\ (i,j) \in A. \tag{2.4}$$

The flow balance constraints (2.1) guarantee that x defines an s-t walk in G, and the *subtour elimination inequalities* (2.2), which are only necessary if negative cost cycles are present, ensure that the path is elementary. Inequalities (2.3) force the optimal path to be weight feasible, and the integrality constraints (2.4) forbid fractional paths.

Because of the widespread success of some of the node labeling methods discussed above, however, few researchers have turned to branch-and-cut for solving the RCSPP. Two recent examples use branch-and-cut to solve for elementary paths in the presence of negative cycles. Avella et al. [7] were forced to use generalized subtour elimination inequalities to solve an extension of the RCSPP with negative arc weights because none of the approaches discussed in Section 2.2 can be used to solve the problem (they all assume non-negative weights). Spoorendonk et al. [106] also used branch-and-cut to solve an extension of the RCSPP with negative arc costs which arises when solving a capacitated vehicle routing

problem with column generation. The authors gave computational results for the single resource case and claim the algorithm actually outperformed node labeling methods in a large majority of the instances.

Even if the underlying graph is acyclic and all costs and weights are positive, the best node labeling algorithm, which mimics the $\mathcal{O}(|A|)$ algorithm for the acyclic SPP (see [1]) and extends non-dominated labels in topological order, still has a worst case complexity of $\mathcal{O}(|A|W_1W_2\cdots W_R)$. If R is large and the resource capacities are of reasonable size, this may result in a prohibitively huge state space. For example, if $|A| = 1000$, $R = 10$ and $W_1 = \cdots = W_R = 20$, this complexity bound would be $\approx 10^{16}$.

As Bauer et al. [10] observed, solving pricing problems in column generation approaches with branch-and-cut may have computational advantages as well. First, a branch-and-cut algorithm may find feasible solutions for the pricing problem more quickly because node labeling methods only find feasible solutions when the sink node is labeled, which may take a long time if the graph is large. Also, it is not necessary to solve the pricing problem to optimality to show that no new variables need to be added to the master problem. The best lower bound given by the LP relaxation at any node in the branch-and-bound tree may be used to prove that no new variables are necessary long before the problem is solved.

Due to the infrequent use of branch-and-cut, very little is known about the RCSPP polytope. Dahl and Realfsen [23] studied a special case of the WCSPP with an acyclic 2-graph and unit weight coefficients. In this case, they show that the polytope associated with the LP relaxation is integral, and the problem can be solved using LP methods. Dumitrescu [34] showed that the subtour elimination inequalities (2.2) can be separated as cutting planes (since there are exponentially many) in polynomial time. Bauer et al. [10] showed several inequalities to be valid for the cardinality constrained circuit problem. This problem is similar to the WCSPP because any path can be viewed as a rooted cycle where the source and sink have been collapsed into a single node. Inequalities which forbid infeasible paths have also been defined for related problems such as the asymmetric Traveling Salesman Problem with time windows [3, 4] and the vehicle routing problem with time windows [75].

To the best of our knowledge, the algorithm in [106] is the only work where cutting

planes, other than some variant of the subtour elimination inequalities, are used to solve the RCSPP. Besides the generalized subtour elimination inequalities, the final algorithm includes two other families of valid inequalities. The first is the lifted knapsack cover inequalities (see [57, 58]) which are valid for general BIPs and found in most commercial solvers. The second is a conditional cut used to forbid any future solutions from using a specified subset of arcs. Two additional families were tested, but not included in the final algorithm because they led to longer overall run times. Despite significantly improving the LP relaxation bound, separation was too expensive for a family of generalized capacity inequalities adapted from the fractional capacity inequalities for the vehicle routing problem. A variant of infeasible path inequalities was also discarded because the LP relaxation became too complicated.

Up to now, research on lower bounds for the RCSPP has focused on Lagrangean relaxation because the subproblem is an (unconstrained) SPP which can be solved rapidly in $\mathcal{O}(\mathcal{SP}(V, A))$ (refer to Section 1.2.2). However, since the SPP polytope is integral, this bound is equivalent to the LP relaxation bound (see Proposition 1.1.4). We believe that if effective valid inequalities are identified for the RCSPP polytope, a cutting plane algorithm can generate stronger bounds and help to make branch-and-cut a viable option in certain situations, e.g., when negative cost cycles are present. Moreover, branch-and-cut may be our only option for cases in which the RCSPP appears as a substructure in a more difficult problem like the extension in [7] or the dial-a-flight problem in [39, 40]. In these cases, we can use our understanding of the substructure to help solve the problem more effectively. We begin our search for valid inequalities in the next chapter.

CHAPTER 3

VALID INEQUALITIES FOR WEIGHT CONSTRAINED PATHS

3.1 Introduction

If we can express a feasibility set $X \subseteq \mathbb{B}^n$ such that $X = \bigcap_r X^r$, it may be worthwhile to find valid inequalities for each X^r separately. If $\pi x \leq \pi_0$ is valid for some X^r, then it is also valid for $X \subseteq X^r$. Our hope is that if these inequalities are strong for the individual X^r, they are also strong for their intersection. This strategy has been used successfully in the study of general binary integer programs (BIPs) of the form

$$\max \{cx : Ax \leq b,\ x \in \mathbb{B}^n\}.$$

Crowder et al. [22] obtained cuts for this problem by isolating a single row r from the constraint set and adding valid inequalities for the feasibility set $X^r = \{x \in \mathbb{B}^n : a^r x \leq b_r\}$ defined by the *knapsack inequality* $a^r x \leq b_r$.

The use of this approach, however, has not been limited to single row relaxations. If the feasibility set for the BIP also possesses some combinatorial substructure, denoted by S, then we can add valid inequalities for each of the sets $S \cap X^r$ which should be stronger than those obtained for X^r. One substructure that has been studied extensively is a set of non-overlapping generalized upper bound (GUB) constraints which specify that at most one variable in each set can be fixed to one (see [22, 58, 94, 111]). Another example is a set of precedence constraints, each of which forbids some variable from being fixed to one unless another variable is also fixed to one (see [15, 99, 107]).

We choose our combinatorial substructure to be the set of s-t paths in a directed graph $G = (V, A)$, and $S \cap X^r$ can be expressed as the feasibility set of the weight constrained

shortest path problem (WCSPP) which is defined as follows

$$x(\delta^+(i)) - x(\delta^-(i)) = \begin{cases} 1, & i = s \\ -1, & i = t \\ 0, & \forall\ i \in V \setminus \{s, t\} \end{cases} \tag{3.1}$$

$$x(\gamma(S)) \leq |S| - 1,\ \forall\ S \subset V,\ 2 \leq |S| \leq |V| - 2 \tag{3.2}$$

$$\sum_{(i,j) \in A} w_{ij} x_{ij} \leq W \tag{3.3}$$

$$x_{ij} \in \{0, 1\},\ \forall\ (i, j) \in A. \tag{3.4}$$

We can think of this set as a relaxation of the resource constrained shortest path problem (RCSPP) where only one resource constraint (knapsack inequality) is considered. We explicitly include the subtour elimination inequalities (3.2) in the constraint set to forbid cycles since the graph is not assumed to be acyclic, and an objective function with negative coefficients may encourage cycles. In practice, these inequalities are not included in the original formulation and separated only as needed.

Typically, the WCSPP is assumed to have resource weights that are all non-negative integers. Throughout this chapter, we allow $W \in \mathbb{R}$ and $w_{ij} \in \mathbb{R}$ for all $(i, j) \in A$. We make heavy use of the shortest paths in G with respect to the resource weights. This is only possible if we can find the shortest path from a single node to all other nodes in polynomial time $\mathcal{O}(\mathcal{SP}(V, A))$ (refer to Section 1.2.2). Therefore, we make the following weaker assumption.

Assumption (A1). *G does not contain a negative weight cycle.*

Now that we have defined a relaxation for the original problem, it can be used to derive valid inequalities. Of course, we could always add well known, general purpose inequalities (as would be the case if we gave the problem to a commercial BIP solver). A trivial approach is to ignore the substructure altogether and add inequalities such as the lifted cover inequalities (see [22, 57, 58]) for the resource constraint (3.3). Another option is to relax our substructure to something more familiar. If we denote our s-t path substructure

by

$$X^P = \left\{ x \in \mathbb{B}^A : \text{subject to (3.1) and (3.2)} \right\},$$

there are two distinct sets of non-overlapping GUB constraints,

$$x(\delta^+(i)) \leq 1, \ \forall \ i \in V \tag{3.5}$$

and

$$x(\delta^-(i)) \leq 1, \ \forall \ i \in V, \tag{3.6}$$

that are implied by X^P since a path cannot enter or exit any node $i \in V$ more than once (otherwise we would have a cycle). Therefore, we can also add inequalities such as the lifted GUB cover inequalities (see [57, 58, 94]) for both of the sets $\left\{ x \in \mathbb{B}^A : \text{subject to (3.3) and (3.5)} \right\}$ and $\left\{ x \in \mathbb{B}^A : \text{subject to (3.3) and (3.6)} \right\}$. However, if we use the actual substructure rather than these relaxations, the resulting inequalities should be stronger.

In the remainder of this chapter, we introduce new classes of valid inequalities for the feasibility set

$$X^{WCP} = \left\{ x \in \mathbb{B}^A : \text{subject to (3.1), (3.2) and (3.3)} \right\}.$$

For each of these classes, a separation routine is given along with its complexity bound. We also outline a polynomial time algorithm for constructing an improved auxiliary conflict graph which is used to separate well known valid inequalities for the node packing polytope.

3.2 Node Precedence Inequalities

Let σ_{ij} be the value of the shortest path from node i to node j with arc lengths given by the weight vector w. Aneja et al. [2] preprocess G by eliminating any arc (i, j) such that $\sigma_{si} + w_{ij} + \sigma_{jt} > W$ and any node k such that $\sigma_{sk} + \sigma_{kt} > W$, since they cannot appear in a weight feasible s-t path. This results in the trivial valid inequalities $x_{ij} \leq 0$, $x_{lk} \leq 0$ for all $(l, k) \in \delta^-(k)$, and $x_{kl} \leq 0$ for all $(k, l) \in \delta^+(k)$, which can be enforced by fixing variable bounds. In this section, we study the sets $\delta^-(k)$ and $\delta^+(k)$ for each $k \in V$ to find stronger inequalities.

3.2.1 Valid Inequalities

Let $(i,j) \in A$ be such that $j \neq s,t$ and consider the arcs leaving j. It follows from the flow balance equation (3.1) for j that the trivial precedence inequality

$$x_{ij} \leq x(\delta^+(j)) \tag{3.7}$$

is valid for X^{WCP}. This inequality implies that any s-t path entering j through (i,j) must leave j through some arc in $\delta^+(j)$ in order to reach t. However, if we know that some $(j,k') \in \delta^+(j)$ can never appear in a weight feasible s-t path along with (i,j), then we can exclude it from the summation on the right hand side and strengthen the inequality (see Figure 2). Note that if $j = s$, the associated trivial precedence inequality is $x_{is} \leq x(\delta^+(s)) - 1$, which together with the GUB inequality (3.5) for s implies that $x_{is} \leq 0$. Hence, we can assume $\delta^-(s) = \emptyset$. If $j = t$, the associated trivial precedence inequality, $x_{it} \leq x(\delta^+(t)) + 1$, is always implied by the variable bound $x_{it} \leq 1$.

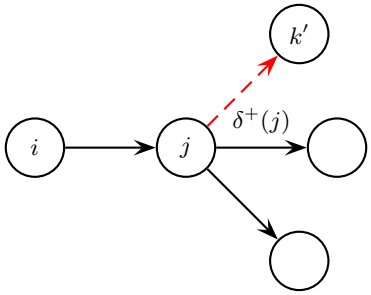

Figure 2: Flow through (i,j) and $\delta^+(j)$

For $(i,j) \in A$, let ϕ_{ij}^+ be the set of arcs out of j which can appear in a weight feasible s-t walk passing through arc (i,j), or

$$\phi_{ij}^+ = \left\{(j,k) \in \delta^+(j) : \sigma_{si} + w_{ij} + w_{jk} + \sigma_{kt} \leq W\right\}.$$

We use the word "walk" here because there is no guarantee that the shortest s-i and k-t paths in G do not share nodes in common, and $(j,i) \in \phi_{ij}^+$ is also possible.

Definition 3.1. Given $(i,j) \in A$ such that $j \neq s,t$, let $F_{ij}^+ = \left\{(j,k) \in \phi_{ij}^+ : k \neq i\right\}$. Then, the *node precedence inequality* is

$$x_{ij} \leq x(F_{ij}^+). \tag{3.8}$$

Proposition 3.2.1. *The node precedence inequality (3.8) is valid for* X^{WCP}.

Proof. Let $(i,j) \in A$ such that $j \neq s,t$. Assume $\bar{x} \in X^{WCP}$, and let P be the corresponding s-t path in G. If $\bar{x}_{ij} = 0$, then (3.8) is trivially valid since $\bar{x}_{jk} \geq 0$ for all $(j,k) \in F_{ij}^{+}$.

Now, suppose $\bar{x}_{ij} = 1$. Then $(i,j) \in A(P)$, and there must exist some $(j,k) \in \delta^{+}(j)$ such that $\bar{x}_{jk} = 1$ and $P = Q \cup (i,j,k) \cup Q'$ for some s-i path Q and k-t path Q' in G. If $(j,k) \notin \phi_{ij}^{+}$, then

$$w(P) = w(Q) + w_{ij} + w_{jk} + w(Q') \geq \sigma_{si} + w_{ij} + w_{jk} + \sigma_{kt} > W$$

and $\bar{x} \notin X^{WCP}$. If $k = i$, then (i,j,k) would be a cycle and $\bar{x} \notin X^{WCP}$. Therefore, $(j,k) \in F_{ij}^{+}$ and $\bar{x}_{ij} = \bar{x}_{jk} \leq \bar{x}(F_{ij}^{+})$, hence (3.8) is valid. □

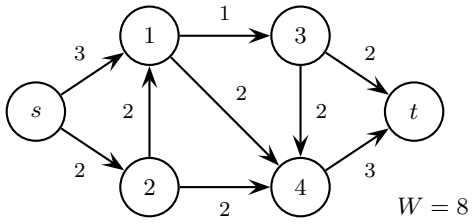

Figure 3: Constrained path example

Example 3.1. Consider the weight constrained path set in Figure 3. Assume we want to derive inequality (3.8) for arc $(1,3)$. We see that $(3,4)$ is not in $F_{13}^{+} \subseteq \{(3,4),(3,t)\}$ since $\sigma_{s1} + w_{13} + w_{34} + \sigma_{4t} = 3 + 1 + 2 + 3 = 9 > 8$. Hence, we get $x_{13} \leq x_{3t}$. ■

Remark 3.

i. If $F_{ij}^{+} = \delta^{+}(j)$*, then inequality (3.8) is redundant.*

ii. If $F_{ij}^{+} = \emptyset$*, then* $\sigma_{si} + w_{ij} + \sigma_{jt} > W$*, and the preprocessing condition of Aneja et al. is implied by (3.8).*

For $(i,j) \in A$, we can also define the set

$$\phi_{ij}^{-} = \left\{(k,i) \in \delta^{-}(i) : \sigma_{sk} + w_{ki} + w_{ij} + \sigma_{jt} \leq W\right\},$$

which suggests the following inequality for the arcs entering i.

Definition 3.2. Given $(i,j) \in A$ such that $i \neq s,t$, let $F_{ij}^- = \left\{(k,i) \in \phi_{ij}^- : k \neq j\right\}$. Then, the *reverse node precedence inequality* is

$$x_{ij} \leq x(F_{ij}^-). \tag{3.9}$$

Proposition 3.2.2. *The reverse node precedence inequality (3.9) is valid for* X^{WCP}.

Proof. The proof of validity is almost identical to the one for inequality (3.8). □

3.2.2 Separation

There are only $\mathcal{O}(|A|)$ possible inequalities (3.8) and (3.9). Hence, they can be added to the formulation a priori. If this number is large with respect to the number of original constraints in the BIP, however, we may want to separate them only as needed. Let $x^* \in [0,1]^A$ be a fractional solution, and let $G^* = (V^*, A^*)$ be the corresponding support graph. In Algorithm 2, we present a simple routine to find all inequalities (3.8) violated by x^*. The algorithm for separating the reverse node precedence inequalities (3.9) is not included here, but proceeds in much the same way.

Algorithm 2 Separating all violated inequalities (3.8) for $x^* \in [0,1]^A$

```
1:  C ← ∅
2:  for all (i,j) ∈ A* such that j ≠ s,t do
3:     v ← x*_ij
4:     for all (j,k) ∈ δ+(j) such that x*_jk > 0 do
5:        if (j,k) ∈ φ+_ij and k ≠ i then
6:           v ← v − x*_jk
7:        end if
8:     end for
9:     if v > 0 then
10:       C ← C ∪ {x_ij ≤ x(F+_ij)}
11:    end if
12: end for
13: return C
```

The work done by the algorithm is simply inspection. For each fractional arc (i,j) whose head is not s or t, we inspect the arc set $\delta^+(j)$ and calculate the violation v. If we find a violated inequality (3.8), we add it to the cut pool $\mathcal{C}$. Each test of $(j,k) \in \phi_{ij}^+$ on line 5 can be done in $\mathcal{O}(1)$ if we know σ_{si} and σ_{kt}. Therefore, the inner loop on lines 4-8 requires

$\mathcal{O}(|V^*|)$ work. This loop is repeated once for each arc in A^*, and so the total work required is $\mathcal{O}(|A^*||V^*|)$. Before running Algorithm 2, though, we must calculate σ_{si} and σ_{it} for all $i \in V$ to define ϕ_{ij}^+. This requires $\mathcal{O}(\mathcal{SP}(V, A))$ work up front, but only needs to be done once.

3.3 Subpath Precedence Inequalities

When deriving node precedence inequalities, we identified pairs of arcs (i, j) and (j, k) that can never appear in a weight feasible s-t path together. We can also view this pair as an infeasible subpath (i, j, k). In this section, we generalize this notion by identifying infeasible subpaths of arbitrary length $p \geq 3$. We use an approach similar to that of the lifted path inequalities used by Kallehauge and Boland [75] for the vehicle routing problem with time windows (VRPTW). However, there are two fundamental attributes which differentiate our inequalities. First, the VRPTW assumes positive flow through every node in the graph. This is not the case for s-t paths, and we are forced to express inequalities as a precedence condition on the flow through some $(i, j) \in A$. Secondly, our notion of an infeasible path is stronger than that of Kallehauge and Boland because they do not incorporate shortest paths while deriving their inequalities.

3.3.1 Valid Inequalities

Definition 3.3. A path Q in G is an *s-t subpath* if Q is a subpath of some s-t path in G. An s-t subpath $Q = (i_1, i_2, \ldots, i_p)$ is *infeasible* if $\sigma_{s,i_1} + w(Q) + \sigma_{i_p,t} > W$.

Remark 4. *If $Q = (i_1, i_2, \ldots, i_p)$ is an infeasible s-t subpath and $X^{WCP} \neq \emptyset$, then $s \neq i_2, \ldots, i_p$ and $t \neq i_1, \ldots, i_{p-1}$.*

If $p < 3$, an s-t subpath corresponds to an infeasible node or arc. As mentioned before, this can be handled without adding inequalities to the BIP formulation by setting variable bounds. Therefore, we only consider infeasible s-t subpaths such that $p \geq 3$.

For any infeasible s-t subpath Q in G, the infeasible subpath inequality

$$x(Q) \leq |A(Q)| - 1 = p - 2 \tag{3.10}$$

is valid for X^{WCP} and forbids Q in any s-t path. However, this inequality can be weak for longer paths, and we can use the following precedence inequality which is stronger.

Lemma 3.3.1. *Let $Q = (i_1, i_2, \ldots, i_p)$ be an infeasible s-t subpath in G. Then, inequality (3.10) is implied by*

$$x_{i_1,i_2} \leq \sum_{k=2}^{p-1} x(\delta^+(i_k) \setminus \{(i_k, i_{k+1})\}). \tag{3.11}$$

Proof. If we add the flow balance equations (3.1) for $i_2, \ldots, i_{p-1}$ to inequality (3.11) we get $x(Q) \leq \sum_{k=2}^{p-1} x(\delta^-(i_k))$, and by (3.6) we know $\sum_{k=2}^{p-1} x(\delta^-(i_k)) \leq p-2$. □

We present the example below to show that inequality (3.11) is indeed stronger than inequality (3.10).

Example 3.1. (cont'd) Note that $Q = (2, 1, 4)$ is an infeasible s-t subpath since $\sigma_{s2} + w_{21} + w_{14} + \sigma_{4t} = 2+2+2+3 = 9 > 8$. Consider the weight feasible fractional solution given by allowing a flow of $\frac{1}{2}$ along path $(s, 2, 4, t)$ and $\frac{1}{2}$ along path $(s, 2, 1, 4, t)$. Inequality (3.10) for Q is given by $x_{21} + x_{14} \leq 1$ and is not violated by the fractional solution. However, inequality (3.11) for Q is given by $x_{21} \leq x_{13}$ and is violated. ■

Inequality (3.11) implies that any s-t path entering Q through (i_1, i_2) must leave Q through some arc in $\delta^+(i_k) \setminus \{(i_k, i_{k+1})\}$ for some $k < p$ in order to reach t. As in the case of the node precedence inequalities, if we know that there exists some $(i_k, j') \in \delta^+(i_k)$ with $j' \neq i_{k+1}$ that can never appear in a weight feasible s-t path along with $(i_1, \ldots, i_k)$, then we can exclude it from the summation on the right hand side and strengthen the inequality (see Figure 4).

For $k = 2, \ldots, p$, let $\phi_Q^+(i_k)$ be the set of arcs out of i_k which can appear in a weight feasible s-t walk passing through $(i_1, \ldots, i_k)$, or

$$\phi_Q^+(i_k) = \left\{ (i_k, j) \in \delta^+(i_k) : \sigma_{s,i_1} + \sum_{h=2}^{k} w_{i_{h-1},i_h} + w_{i_k,j} + \sigma_{jt} \leq W \right\}.$$

Not only do we remove the next arc (i_k, i_{k+1}) in Q from this set, but we can also remove all arcs that go to earlier nodes in Q because they create cycles.

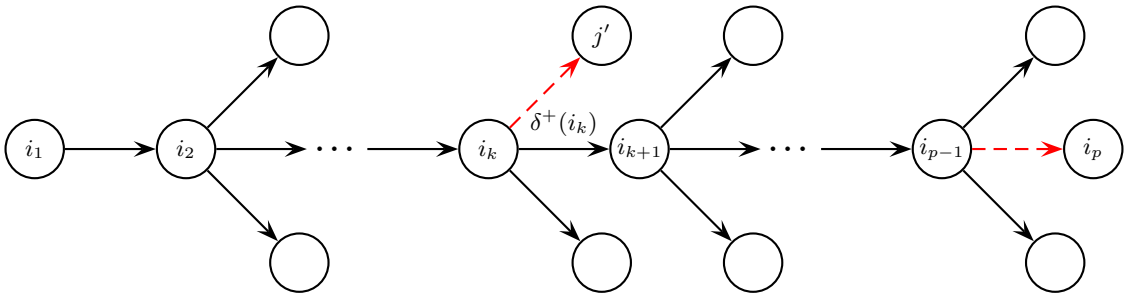

Figure 4: Flow through infeasible subpath $Q = (i_1, i_2, \ldots, i_p)$

Definition 3.4. Given an infeasible s-t subpath $Q = (i_1, i_2, \ldots, i_p)$ in G, let

$$F_Q^+(i_k) = \left\{(i_k, j) \in \phi_Q^+(i_k) : j \neq i_{k+1},\ j \neq i_1, \ldots, i_{k-1}\right\} \text{ for } k = 2, \ldots, p-1$$

and

$$F_Q^+ = \bigcup_{k=2}^{p-1} F_Q^+(i_k).$$

Then, the *subpath precedence inequality* is

$$x_{i_1,i_2} \leq x(F_Q^+). \tag{3.12}$$

Proposition 3.3.2. *The subpath precedence inequality (3.12) is valid for* X^{WCP}.

Proof. Let $Q = (i_1, i_2, \ldots, i_p)$ be an infeasible s-t subpath in G. Assume $\bar{x} \in X^{WCP}$, and let P be the corresponding s-t path in G. If $\bar{x}_{i_1,i_2} = 0$, then (3.12) is trivially valid since $\bar{x}_{i_k,j} \geq 0$ for all $(i_k, j) \in F_Q^+$.

Now, suppose $\bar{x}_{i_1,i_2} = 1$. Then $(i_1, i_2, \ldots, i_k)$ is a subpath of P for some $2 \leq k \leq p-1$ (since Q itself is infeasible), and there must exist some $(i_k, j) \in \delta^+(i_k)$ with $j \neq i_{k+1}$ such that $\bar{x}_{i_k,j} = 1$ and $P = Q' \cup (i_1, i_2, \ldots, i_k, j) \cup Q''$ for some s-i_1 path Q' and j-t path Q'' in G. If $(i_k, j) \notin \phi_Q^+(i_k)$, then

$$w(P) = w(Q') + \sum_{h=2}^{k} w_{i_{h-1},i_h} + w_{i_k,j} + w(Q'') \geq \sigma_{s,i_1} + \sum_{h=2}^{k} w_{i_{h-1},i_h} + w_{i_k,j} + \sigma_{jt} > W$$

and $\bar{x} \notin X^{WCP}$. If $j \in \{i_1, \ldots, i_{k-1}\}$, then $(i_1, \ldots, i_k, j)$ would contain a cycle and $\bar{x} \notin X^{WCP}$. Therefore, $(i_k, j) \in F_Q^+(i_k) \subseteq F_Q^+$ and $\bar{x}_{i_1,i_2} = \bar{x}_{i_k,j} \leq \bar{x}(F_Q^+)$, hence (3.12) is valid. ∎

Example 3.1. (cont'd) To derive inequality (3.12) for the infeasible s-t subpath $Q = (2, 1, 4)$ considered above, we need to find $F_Q^+ = F_Q^+(1) \subseteq \{(1, 3)\}$. The arc $(1, 3)$ is feasible since $\sigma_{s2} + w_{21} + w_{13} + \sigma_{3t} = 2 + 2 + 1 + 2 = 7 \leq 8$. Hence, we get $x_{21} \leq x_{13}$. ■

Remark 5. *If $p = 3$, we get the node precedence inequality (3.8) for (i_1, i_2) since*

$$\phi^+_{i_1,i_2} = \phi^+_{(i_1,i_2)}(i_2).$$

Hence, the preprocessing condition of Aneja et al. is also implied by (3.12).

For $k = 1, \ldots, p-1$, we can also define the set

$$\phi^-_Q(i_k) = \left\{ (j, i_k) \in \delta^-(i_k) : \sigma_{sj} + w_{j,i_k} + \sum_{h=k}^{p-1} w_{i_h,i_{h+1}} + \sigma_{i_p,t} \leq W \right\},$$

which suggests the following inequality for the arcs entering $i_2, \ldots, i_{p-1}$.

Definition 3.5. Given an infeasible s-t subpath $Q = (i_1, i_2, \ldots, i_p)$ in G, let

$$F^-_Q(i_k) = \left\{ (j, i_k) \in \phi^-_Q(i_k) : j \neq i_{k-1},\ j \neq i_{k+1}, \ldots, i_p \right\} \text{ for } k = 2, \ldots, p-1$$

and

$$F^-_Q = \bigcup_{k=2}^{p-1} F^-_Q(i_k).$$

Then, the *reverse subpath precedence inequality* is

$$x_{i_{p-1},i_p} \leq x(F^-_Q). \tag{3.13}$$

Proposition 3.3.3. *The reverse subpath precedence inequality (3.13) is valid for X^{WCP}.*

Proof. The proof of validity is almost identical to the one for inequality (3.12). □

3.3.2 Separation

While we are ultimately interested in separating the subpath precedence inequalities (3.12) and (3.13), we begin by showing the separation problem for the infeasible subpath inequality (3.10) is $\mathcal{NP}$-hard. To separate (3.10) for a fractional solution $x^* \in [0, 1]^A$, we want to find an infeasible s-t subpath Q such that $x^*(Q) > |A(Q)| - 1$. This inequality can be restated as

$$\sum_{(i,j) \in A(Q)} (1 - x^*_{ij}) < 1.$$

Therefore, we can define the separation problem as follows:

Problem (IS-SEP). Infeasible Subpath Separation

Instance: We are given a directed graph $G = (V, A)$, a fractional solution $x^* \in [0,1]^A$, arc weights $w_{ij} \in \mathbb{R}$, specified vertices $s, t \in V$, and capacity $W \in \mathbb{R}$.

Question: Is there an s-t subpath $Q = (i_1, \ldots, i_p)$ in G such that $\sigma_{s,i_1} + w(Q) + \sigma_{i_p,t} > W$ and $\sum_{k=1}^{p-1}(1 - x^*_{i_k,i_{k+1}}) < 1$?

Proposition 3.3.4. *IS-SEP is $\mathcal{NP}$-hard.*

Proof. We transform from Knapsack Cover Separation (KC-SEP) which has been shown to be $\mathcal{NP}$-hard [81].

Instance: We are given a set $N = \{1, \ldots, n\}$, a fractional solution $y^* \in [0,1]^N$, weights $a_j \in \mathbb{Z}_+$ for each $j \in N$, and positive integer B.

Question: Is there a subset $C \subseteq N$ such that $a(C) > B$ and $\sum_{j \in C}(1 - y^*_j) < 1$?

Let $W = B$ and $V = \{s, 1, 1', 2, 2', \ldots, n, n', t\}$. We construct (acyclic) $G = (V, A)$ such that A is the union of the arc sets $\{(s,1), (s,1'), (n,t), (n',t)\}$, $\{(j, j+1) : j \in N \setminus \{n\}\}$, $\{(j,j') : j \in N\}$, $\{(j', j+1) : j \in N \setminus \{n\}\}$ and $\{(j', (j+1)') : j \in N \setminus \{n\}\}$. For all $j \in N$, let $x^*_{jj'} = y^*_j$ and $w_{jj'} = a_j$. For all remaining arcs $(i,j) \in A$, let $x^*_{ij} = 1$ and $w_{ij} = 0$.

By construction,

$$\sum_{k=1}^{p-1}(1 - x^*_{i_k,i_{k+1}}) = \sum_{(j,j') \in A(Q)}(1 - x^*_{jj'}) = \sum_{(j,j') \in A(Q)}(1 - y^*_j)$$

and

$$\sigma_{s,i_1} + w(Q) + \sigma_{i_p,t} = w(Q) = \sum_{(j,j') \in A(Q)} w_{jj'} = \sum_{(j,j') \in A(Q)} a_j$$

for any s-t subpath $Q = (i_1, \ldots, i_p)$ in G, since $w_{ij} \in \mathbb{Z}_+$ and $\sigma_{si} = \sigma_{it} = 0$ for all $i \in V$. Therefore, if Q is a feasible solution to IS-SEP on G, then $C = \{j \in N : (j,j') \in A(Q)\}$ is a feasible solution to KC-SEP. Conversely, let C be a feasible solution to KC-SEP, and $(j_1, \ldots, j_{|C|})$ be the natural ordering of the elements in C. If $Q_k = (j'_k, j_k+1, j_k+2, \ldots, j_{k+1})$ for $k = 1, \ldots, |C|-1$, then the s-t subpath $Q = (j_1, j'_1) \cup Q_1 \cup (j_2, j'_2) \cup \cdots \cup Q_{|C|-1} \cup (j_{|C|}, j'_{|C|})$ is a feasible solution to IS-SEP. □

Remark 6. *IS-SEP is still $\mathcal{NP}$-hard if we restrict ourselves to the case where G is acyclic, and $w_{ij} \in \mathbb{Z}_+$ for all $(i,j) \in A$ (see the transformation in the proof).*

This result does not prove that the separation problems for the stronger subpath precedence inequalities (3.12) and (3.13) are $\mathcal{NP}$-hard, but we conjecture this to be true since any feasible solution Q to IS-SEP is also a feasible solution to the separation problem for both (3.12) and (3.13). Therefore, we focus our efforts on separation heuristics.

Definition 3.6. An infeasible s-t subpath $Q = (i_1, i_2, \ldots, i_p)$ is *minimal* if the truncated s-t subpaths $(i_2, \ldots, i_p)$ and $(i_1, \ldots, i_{p-1})$ are feasible.

If Q is not minimal, then either $F_Q^+(i_{p-1}) = \emptyset$ or $F_Q^-(i_2) = \emptyset$, and we can find an inequality (3.12) for $(i_1, \ldots, i_{p-1})$ or (3.13) for $(i_2, \ldots, i_p)$ which implies the respective inequality for Q. Therefore, it is sufficient to find minimal infeasible s-t subpaths when separating (3.12) and (3.13).

Let $G^* = (V^*, A^*)$ be the support graph for $x^* \in [0,1]^A$. To find many inequalities (3.12) violated by x^*, we have modified the enumeration procedure used by Kallehauge and Boland [75] and present it in Algorithm 3. The heuristic for separating inequalities (3.13) is not included here, but proceeds in much the same way.

For each fractional arc (i_1, i_2) whose head is not s or t, we enumerate over all fractional s-t subpaths $Q = (i_1, i_2, \ldots, i_k)$ such that the violation $v = x^*_{i_1,i_2} - x^*(F_Q^+) > 0$. We backtrack if $v \leq 0$, otherwise, we extend the fractional paths to all nodes $j \in V^+(i_k) \setminus V(Q)$. If an infeasible path Q is detected, we first check to see if it is minimal and add inequality (3.12) if violated; then we backtrack.

Most of the work is done by the recursive function SUBPATH_PREC_DFS. We call this function each time we add a new node to the end of the subpath Q. When called for node i_k, our first task is to classify all arcs in $\delta^+(i_k)$ as either feasible or infeasible in the loop on lines 10-21. If (i_k, j) is a feasible arc in A^* and does not cause a cycle, we add it to the list of arcs $\mathcal{F}[k]$ to continue exploring out of i_k and update the violation v with $x^*_{i_k,j}$. If (i_k, j) is not feasible, we check to see if the subpath $(i_2, \ldots, i_k, j)$ is feasible (or $(i_1, \ldots, i_k, j)$ is minimal infeasible) and record it in the boolean I. Once we have classified all arcs in $\delta^+(i_k)$, we check to see if any minimal infeasible s-t subpaths $(i_1, \ldots, i_k, j)$ were found ($I =$ true), and if the violation $v > 0$, we add inequality (3.12) to the cut pool $\mathcal{C}$. Finally, we call

Algorithm 3 Heuristic for finding all violated inequalities (3.12) for $x^* \in [0,1]^A$

$\mathcal{C} \leftarrow \emptyset$
for all $(i_1, i_2) \in A^*$ such that $i_2 \neq s, t$ **do**
  $v \leftarrow x^*_{i_1,i_2}$, $Q \leftarrow (i_1, i_2)$
  SUBPATH_PREC_DFS(2)
end for
return $\mathcal{C}$

SUBPATH_PREC_DFS(k):
$\mathcal{F}[k] \leftarrow \emptyset, I \leftarrow$ false
for all $(i_k, j) \in \delta^+(i_k)$ **do**
  if $(i_k, j) \in \phi^+_Q(i_k)$ **then**
    if $x^*_{i_k,j} > 0$ and $j \neq i_1, \ldots, i_{k-1}$ and $j \neq s$ **then**
      $\mathcal{F}[k] \leftarrow \mathcal{F}[k] \cup \{(i_k, j)\}$
      $v \leftarrow v - x^*_{i_k,j}$
    end if
  else
    if $(i_k, j) \in \phi^+_{(i_2,\ldots,i_k)}(i_k)$ **then**
      $I \leftarrow$ true
    end if
  end if
end for
if $I =$ true and $v > 0$ **then**
  $\mathcal{C} \leftarrow \mathcal{C} \cup \left\{ x_{i_1,i_2} \leq x(F^+_Q) \right\}$
end if
for all $(i_k, j) \in \mathcal{F}[k]$ such that $j \neq t$ and $v + x^*_{i_k,j} > 0$ **do**
  $v \leftarrow v + x^*_{i_k,j}$, $Q \leftarrow (i_1, \ldots, i_k, j)$
  SUBPATH_PREC_DFS($k+1$)
  $v \leftarrow v - x^*_{i_k,j}$, $Q \leftarrow (i_1, \ldots, i_k)$
end for
return

SUBPATH_PREC_DFS for each j such that $(i_k, j) \in \mathcal{F}[k]$ and $j \neq t$ with sufficient violation.

Each test of $(i_k, j) \in \phi_Q^+(i_k)$ on line 11 or $(i_k, j) \in \phi^+_{(i_2,\ldots,i_k)}(i_k)$ on line 17 can be done in $\mathcal{O}(1)$ by updating a variable $w_Q = \sigma_{s,i_1} + w(Q)$ each time we modify Q. Testing if $j \neq i_1, \ldots, i_{k-1}$ on line 12 can also be done in $\mathcal{O}(1)$ by updating an array of labels that indicates whether a node is in Q or not. Hence, every time we call SUBPATH_PREC_DFS, $\mathcal{O}(|V|)$ work is required, and we must call it for each node in a fractional s-t subpath. Therefore, the total work required by Algorithm 3 for any enumerated subpath is $\mathcal{O}(|V^*||V|)$, but there could be an exponential number of such subpaths. Our hope is that the number of fractional paths in G^* is not prohibitive in applications. Backtracking when $v \leq 0$ should help reduce the number of subpaths enumerated, but if the heuristic becomes too expensive, we can always place a limit on the length of the subpaths to be considered. In theory, such a limit can give us a polynomial bound on the running time. Before using Algorithm 3, though, we must calculate σ_{si} and σ_{it} for all $i \in V$ to define $\phi_Q^+(i_k)$. This requires $\mathcal{O}(\mathcal{SP}(V, A))$ work up front, but only needs to be done once.

3.4 s-t Cut Precedence Inequalities

Up until now, we have only considered infeasible combinations of sequential arcs (subpaths). In this section, we generalize by identifying infeasible pairs of arcs which do not necessarily share any nodes in common. One of them, however, will be required to appear in a certain s-t cutset. To do this, we will need the all-pairs shortest paths σ_{ij} for all $i, j \in V$, as opposed to just σ_{si} and σ_{it} for all $i \in V$.

3.4.1 Valid Inequalities

Let S be an s-t cut in G. If we sum up all the flow balance equations (3.1) for $i \in S$ we get

$$x(\delta^+(S)) - x(\delta^-(S)) = 1.$$

Hence, the total (net) flow across any s-t cut is one, and in order for a path to get from s to t, it must traverse the arcs in $\delta^+(S)$ at least once. Furthermore, if $(i, j) \in A$ such that $j \in S \setminus \{s\}$, we know that any s-t path through (i, j) must contain some j-t subpath which must traverse the arcs in $\delta^+(S) \cap \gamma(\rho^+(j))$ at least once. Therefore, we know the trivial

precedence inequality

$$x_{ij} \leq x(\delta^+(S) \cap \gamma(\rho^+(j))) \tag{3.14}$$

is valid for X^{WCP}. However, if we know that some $(k', l') \in \gamma(\rho^+(j))$ can never appear in a weight feasible s-t path along with (i, j), then we can exclude it from the summation on the right hand side and strengthen the inequality (see Figure 5). Note that we need not consider $j = s$ because, as we showed for node precedence inequalities, we can assume $\delta^-(s) = \emptyset$.

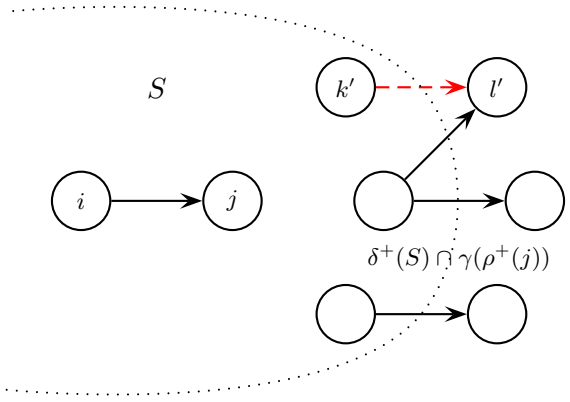

Figure 5: Flow through (i, j) and $\delta^+(S) \cap \gamma(\rho^+(j))$

For $(i, j) \in A$, let Φ^+_{ij} be the set of all arcs which can appear in a weight feasible s-t walk passing through arc (i, j) first, or

$$\Phi^+_{ij} = \{(k, l) \in A : \sigma_{si} + w_{ij} + \sigma_{jk} + w_{kl} + \sigma_{lt} \leq W\}.$$

We can remove any arc in $\delta^+(i)$ (which includes (i, j)) since only one arc can exit i. There is nothing that prohibits $i \in \bar{S}$, so we can also remove any arc in $\delta^-(i)$ (which includes (j, i)) because it creates a cycle. The arc set $\delta^-(j)$ could be removed as well, but any arc in this set will never appear in $\delta^+(S)$.

Definition 3.7. Given an s-t cut $S \subset V$ and $(i, j) \in A$ such that $j \in S \setminus \{s\}$, let

$$F^+_{ij}(S) = \left\{(k, l) \in \delta^+(S) : (k, l) \in \Phi^+_{ij},\ k \neq i,\ l \neq i\right\}.$$

Then, the *s-t cut precedence inequality* is

$$x_{ij} \leq x(F^+_{ij}(S)). \tag{3.15}$$

Proposition 3.4.1. *The s-t cut precedence inequality (3.15) is valid for* X^{WCP}.

Proof. Let $(i,j) \in A$ and $S \subset V$ be an s-t cut such that $j \in S \setminus \{s\}$. Assume $\bar{x} \in X^{WCP}$, and let P be the corresponding s-t path in G. If $\bar{x}_{ij} = 0$, then (3.15) is trivially valid since $\bar{x}_{kl} \geq 0$ for all $(k,l) \in F_{ij}^{+}(S)$.

Now, suppose $\bar{x}_{ij} = 1$. Then $(i,j) \in A(P)$, and since S is an s-t cut with $j \in S$, there must exist some j-t subpath of P that uses at least one of the arcs in $\delta^{+}(S)$. Therefore, there exists some $(k,l) \in \delta^{+}(S)$ such that $\bar{x}_{kl} = 1$ and $P = Q \cup (i,j) \cup Q' \cup (k,l) \cup Q''$ for some s-i path Q, j-k path Q', and l-t path Q'' in G. If $(k,l) \notin \Phi_{ij}^{+}$, then

$$w(P) = w(Q) + w_{ij} + w(Q') + w_{kl} + w(Q'') \geq \sigma_{si} + w_{ij} + \sigma_{jk} + w_{kl} + \sigma_{lt} > W$$

and $\bar{x} \notin X^{WCP}$. If $k = i$, then $(i,j) \cup Q'$ would be a cycle and $\bar{x} \notin X^{WCP}$. If $l = i$, then $(i,j) \cup Q' \cup (k,l)$ would be a cycle and $\bar{x} \notin X^{WCP}$. Therefore, $(k,l) \in F_{ij}^{+}(S)$ and $\bar{x}_{ij} = \bar{x}_{kl} \leq \bar{x}(F_{ij}^{+}(S))$, hence (3.15) is valid. □

Example 3.1. (cont'd) Let $S = \{s, 1, 2\}$. To derive inequality (3.15) for arc $(s,2)$ we need to find $F_{s2}^{+}(S) \subseteq \{(1,3),(1,4),(2,4)\}$. We see that $(1,4)$ is not feasible since $\sigma_{ss} + w_{s2} + \sigma_{21} + w_{14} + \sigma_{4t} = 0 + 2 + 2 + 2 + 3 = 9 > 8$. Hence, we get $x_{s2} \leq x_{13} + x_{24}$. ■

Remark 7. *Let* $(i,j) \in A$ *such that* $j \neq s, t$.

i. $\Phi_{ij}^{+} \subseteq \gamma(\rho^{+}(j))$ *(since* $\sigma_{jk} = \infty$ *for all* $k \notin \rho^{+}(j)$*).*

ii. $\phi_{ij}^{+} \subseteq \Phi_{ij}^{+}$ *(since* $\sigma_{jj} = 0$ *for all* $j \in V$*), and if* $V^{+}(j) \in \bar{S}$*, then inequality (3.15) is implied by the node precedence inequality (3.8).*

For $(i,j) \in A$, we can also define the set

$$\Phi_{ij}^{-} = \{(k,l) \in A : \sigma_{sk} + w_{kl} + \sigma_{li} + w_{ij} + \sigma_{jt} \leq W\},$$

which suggests the following inequality for any s-t cut S such that $i \in \bar{S} \setminus \{t\}$.

Definition 3.8. Given an s-t cut $S \subset V$ and $(i,j) \in A$ such that $i \in \bar{S} \setminus \{t\}$, let

$$F_{ij}^{-}(S) = \left\{(k,l) \in \delta^{+}(S) : (k,l) \in \Phi_{ij}^{-},\ k \neq j,\ l \neq j\right\}.$$

Then, the *reverse s-t cut precedence inequality* is

$$x_{ij} \leq x(F_{ij}^{-}(S)). \tag{3.16}$$

Proposition 3.4.2. *The reverse s-t cut precedence inequality (3.16) is valid for* X^{WCP}.

Proof. The proof of validity is almost identical to the one for inequality (3.15). □

3.4.2 Separation

When separating inequalities (3.15) and (3.16) for some arc (i, j), there are exponentially many choices for S, and we resort to finding a maximally violated inequality over all S. Let $x^* \in [0,1]^A$ be a fractional solution, and let $G^* = (V^*, A^*)$ be the corresponding support graph. In Algorithm 4, we separate an inequality (3.15) for each $(i, j) \in A^*$ that is maximally violated by x^*. The algorithm for separating inequalities (3.16) is not included here, but proceeds in much the same way.

Algorithm 4 Separating maximally violated inequalities (3.15) for $x^* \in [0,1]^A$

$\mathcal{C} \leftarrow \emptyset$
for all $(i, j) \in A^*$ such that $j \neq s, t$ **do**
  $v \leftarrow x^*_{ij}$
  for all $(k, l) \in A^*$ **do**
    $u_{kl} \leftarrow \begin{cases} x^*_{kl}, & \text{if } (k,l) \in \Phi^+_{ij} \text{ and } k \neq i \text{ and } l \neq i, j \\ 0, & \text{otherwise} \end{cases}$
  end for
  find a minimum s-t cut S in G^* such that $j \in S$ with arc capacities u_{kl}
  $v \leftarrow v - u(\delta^+(S))$
  if $v > 0$ **then**
    $\mathcal{C} \leftarrow \mathcal{C} \cup \left\{ x_{ij} \leq x(F^+_{ij}(S)) \right\}$
  end if
end for
return $\mathcal{C}$

For each fractional arc (i, j) whose head is not s or t, we must solve a minimum s-t cut problem on G^* where the arc capacities are given by x^*_{kl} if feasible and zero otherwise. We must also enforce that j appears on the same side of the cut as s. This is not a major obstacle and can be overcome by adding an infinite capacity arc (s, j). If the value of this minimum cut is at least x^*_{ij}, then inequality (3.15) is satisfied for all s-t cuts. Otherwise,

we have found a maximally violated inequality and its corresponding s-t cut S, and we can add it to the cut pool $\mathcal{C}$.

The work done for each $(i,j) \in A^*$, consists of defining the arc capacities and solving the minimum s-t cut problem. Each test of $(k,l) \in \Phi_{ij}^+$ on line 5 can be done in $\mathcal{O}(1)$, so the inner loop on lines 4-6 which defines the arc capacities can be done in $\mathcal{O}(|A^*|)$. Solving the minimum s-t cut problem on line 7 can be done in $\mathcal{O}(|V^*|^3)$ by solving a maximum flow problem on G^*. Therefore, the total work required is $\mathcal{O}(|A^*||V^*|^3)$. Before running Algorithm 4, though, we must calculate σ_{ij} for all $i,j \in V$ to define Φ_{ij}^+. This requires $\mathcal{O}(|V|\mathcal{SP}(V,A))$ work up front, which is more work than the previous inequalities required, but only needs to be done once.

3.5 Strengthened Precedence Inequalities

For all of the precedence inequalities (node, subpath and s-t cut) we have an expression of the form $x_{ij} \leq x(F)$ for some $(i,j) \in A$ and a feasible set of arcs $F \subseteq A \setminus \{(i,j)\}$. This expression guarantees that if the path traverses (i,j), then it must also traverse at least one of the arcs in F. In each case, we arrive at the strongest inequality possible by finding the smallest set F for which the expression remains valid for X^{WCP}. In this section, we want to consider strengthening these inequalities further by adding arcs to the left hand side.

Lemma 3.5.1. *Let $j \in V \setminus \{s,t\}$, $E \subseteq \delta^-(j)$ and $F \subseteq A$. If $x_{ij} \leq x(F)$ is valid for X^{WCP} for all $(i,j) \in E$, then $x(E) \leq x(F)$ is valid for X^{WCP}.*

Proof. Let $\bar{x} \in X^{WCP}$. If $\bar{x}(E) = 0$, then the inequality is trivially valid since $\bar{x}_{kl} \geq 0$ for all $(k,l) \in F$. Now, suppose $\bar{x}(E) > 0$. By the GUB inequality (3.6) for j, we know that $\bar{x}(E) \leq \bar{x}(\delta^-(j)) \leq 1$. Hence, $\bar{x}(E) = 1$ and there exists exactly one $(i,j) \in E$ fixed to one. Therefore, $\bar{x}(E) = \bar{x}_{ij} \leq \bar{x}(F)$ and the inequality is valid. □

For a guide on how to strengthen the precedence inequalities with this lemma, let us examine the node precedence inequality (3.8) for a particular $(i,j) \in A$. If we relax slightly and allow (j,i) to appear in the right hand side if feasible, we get the valid inequality $x_{ij} \leq x(\phi_{ij}^+)$ since $F_{ij}^+ \subseteq \phi_{ij}^+$. For $(i,j) \in A$ such that $j \neq s,t$, let ∇_{ij}^- be the set of arcs into

j which have shortest s-j paths through them that are at least as long as the shortest s-j path through (i,j), or

$$\nabla_{ij}^{-} = \left\{(k,j) \in \delta^{-}(j) : \sigma_{sk} + w_{kj} \geq \sigma_{si} + w_{ij}\right\}.$$

Proposition 3.5.2. *Given $(i,j) \in A$ such that $j \neq s,t$, then*

$$x(\nabla_{ij}^{-}) \leq x(\phi_{ij}^{+}) \tag{3.17}$$

is valid for X^{WCP}.

Proof. For all $(k,j) \in \nabla_{ij}^{-}$, we know that $\phi_{kj}^{+} \subseteq \phi_{ij}^{+}$. Hence, $x_{kj} \leq x(\phi_{kj}^{+}) \leq x(\phi_{ij}^{+})$ for all $(k,j) \in \nabla_{ij}^{-}$, and by the previous lemma we have that inequality (3.17) is valid. □

Arc (j,i) can always be removed from the right hand side of inequality (3.8) because if (i,j) is included in the s-t path, then including (j,i) causes a cycle. In order for the right hand side of inequality (3.17) to be valid, we must add in (j,i) if $(j,i) \in \phi_{ij}^{+}$ because the s-t subpath (k,j,i,l) may be feasible for some $(k,j) \in \nabla_{ij}^{-}$ and $i \neq k \neq j \neq l$. We may have weakened the inequality slightly by adding the arc to the right hand side, but we still require it to be weight feasible with respect to (i,j), and we now have the possibility of a much stronger left hand side. Note that if the graph is acyclic, (j,i) cannot appear in the graph, and the right hand side is exactly the same.

Separating inequality (3.17) can be done with two simple modifications to Algorithm 2. Rather than initializing the violation v to x_{ij}^{*} on line 3, we initialize to $x^{*}(\nabla_{ij}^{-})$ in $\mathcal{O}(|V^{*}|)$. The second change is to remove the $k \neq i$ condition in the test on line 5. Neither of these modifications affect the complexity of the algorithm.

For $(i,j) \in A$ such that $i \neq s,t$, we can also define the set

$$\nabla_{ij}^{+} = \left\{(i,k) \in \delta^{+}(i) : w_{ik} + \sigma_{kt} \geq w_{ij} + \sigma_{jt}\right\}$$

which allows us to strengthen the reverse node precedence inequality (3.9).

Proposition 3.5.3. *Given $(i,j) \in A$ such that $i \neq s,t$, then*

$$x(\nabla_{ij}^{+}) \leq x(\phi_{ij}^{-}) \tag{3.18}$$

is valid for X^{WCP}.

Proof. The proof of validity is almost identical to the one for inequality (3.17). □

Similar inequalities can be derived for the subpath precedence inequalities and the s-t cut precedence inequalities (and their reverse versions). For subpath precedence inequalities, we can exclude all backwards arcs $(i_k, j) \in \phi_Q^+(i_k)$ such that $j \in \{i_2, \ldots, i_{k-1}\}$, but we must allow (i_k, i_1) (allow (i_p, i_k) for the reverse version). In the case of s-t cut precedence inequalities, we must allow arcs (i, l) and (k, i) in $\delta^+(S) \cap \Phi_{ij}^+$ (allow (j, l) and (k, j) for the reverse version). Algorithms 3 and 4 can also be modified slightly to separate the stronger inequalities with no added complexity.

3.6 Conflict Graph Inequalities

In deriving, s-t cut precedence inequalities, we identified infeasible pairs of arcs in which one of them appeared in a certain s-t cutset. In this section, we generalize by considering any pair of arcs which can never appear in a feasible s-t path.

3.6.1 Valid Inequalities from Conflict Graphs

A *conflict graph* is an undirected graph used to represent logical relationships between binary variables in a BIP. This graph is constructed by having a node for each binary variable, and an arc between two nodes when at most one of the variables represented by the nodes can be set to one in a feasible solution. Conflict graphs have been used to generate valid inequalities for BIPs in many applications areas including airline crew scheduling [66], truck dispatching and scheduling [13], and facility location [82]. Atamturk et al. [5] investigated the use of conflict graphs in solving general BIPs, which includes a role in preprocessing.

Any solution to the BIP defines a node packing in the conflict graph, i.e., a subset of nodes in which no two nodes are joined by an arc. Hence, the *node packing polytope* associated with the conflict graph $G_C = (V_C, E_C)$, which is defined by

$$X^{NP} = \left\{ x \in \mathbb{B}^{V_C} : x_i + x_j \leq 1 \text{ for all } (i, j) \in E_C \right\},$$

describes a relaxation of the convex hull of feasible solutions to the BIP. Therefore, any valid inequalities for the node packing polytope X^{NP} are also valid for the BIP. While there have been many valid inequalities developed for X^{NP}, we consider two fundamental ones.

A set $C \subseteq V_C$ is a *clique* if each pair of nodes in C is joined by an arc in E_C. When C is a maximal clique, the *clique inequality*

$$x(C) \leq 1 \tag{3.19}$$

defines a facet of $\text{conv}(X^{NP})$ [95]. To separate inequality (3.19) for some $x^* \in [0,1]^{V_C}$, we must solve a maximum weighted clique problem on G_C with weight vector x^* which is $\mathcal{NP}$-hard [76]. If the maximum weight clique has value less than or equal to one, then (3.19) is satisfied over all cliques C. Otherwise, we have found a maximally violated inequality, and it can be added to the cut pool. Rather, than solve this problem exactly, we will use a simple greedy separation routine. For each $i \in V_C$ such that $0 < x_i^* < 1$, we will generate a clique C by initializing $C = \{i\}$ and adding elements in the set $\{j \in V_C \setminus C : (j,k) \in E_C \text{ for all } k \in C\}$ in non-increasing x_j^* order. Once the set $V_C \setminus C$ is empty, we test C for violation. Note that if $x_i^* = 1$, then (3.19) can never be violated for any clique containing i.

A chordless cycle $H \subseteq G_C$ with an odd number of nodes is called an *odd hole*. An odd hole with $|H| = 3$ is a clique, so we restrict ourselves to odd holes with $|H| \geq 5$. If H is an odd hole, the *odd hole inequality*

$$x(H) \leq \frac{|H|-1}{2} \tag{3.20}$$

is valid for X^{NP} [95]. Since this inequality generally does not define a facet for $\text{conv}(X^{NP})$, we prefer a *lifted odd hole inequality* of the form

$$x(H) + \sum_{i \in V_C \setminus H} \alpha_i x_i \leq \frac{|H|-1}{2}. \tag{3.21}$$

To separate inequality (3.21) for some $x^* \in [0,1]^{V_C}$, we use the approach by Hoffman and Padberg [66]. Initially, the authors search for a most violated inequality (3.20) by heuristically finding a maximum weight odd hole H with weight vector x^*. They build an auxiliary "layered graph" for each $i \in V_C$ such that $0 < x_i^* < 1$ and use shortest paths on this layered graph to construct a best H if it exists. Once such an odd hole H is found, they approximately lift in the remaining $i \in V_C \setminus H$ by solving an integer version of the dual to the node packing problem with a greedy algorithm. For an alternative to this approach, we refer the reader to the paper by Nemhauser and Sigismondi [93].

3.6.2 Building the Conflict Graph

Typically, the arc set E_C for the conflict graph is approximated with probing techniques using feasibility or optimality considerations as in [5]. For WCSPP, we can use the structure of the problem to define E_C more accurately. In this case, each node in the conflict graph will correspond to an arc in our original graph $G = (V, A)$, i.e., $V_C = A$. For each pair of arcs $(i, j), (k, l) \in A$, we want to include the arc $((i, j), (k, l))$ in $E_C \subset A \times A$ if at most one of (i, j) or (k, l) can be found in a weight feasible s-t path in G. To test this condition we use the all-pairs shortest paths σ_{ij} for all $i, j \in V$.

Proposition 3.6.1. *Given $(i, j) \in A$, let*

$$F_{ij} = \left\{(k, l) \in A : (k, l) \in \Phi^-_{ij} \cup \Phi^+_{ij},\ k \neq i,\ l \neq j,\ (k, l) \neq (j, i)\right\}.$$

If $(k, l) \in A \setminus F_{ij}$ such that $(k, l) \neq (i, j)$, then the inequality

$$x_{ij} + x_{kl} \leq 1 \tag{3.22}$$

is valid for X^{WCP}.

Proof. Let $(i, j) \in A$. Assume $\bar{x} \in X^{WCP}$, and let P be the corresponding s-t path in G. If $\bar{x}_{ij} = 0$, then (3.22) is trivially valid since $\bar{x}_{kl} \leq 1$ for all $(k, l) \in A$.

Now, suppose $\bar{x}_{ij} = 1$ and $(i, j) \in A(P)$. Let $(k, l) \in A \setminus F_{ij}$ such that $(k, l) \neq (i, j)$. If $\bar{x}_{kl} = 1$ and $(k, l) \in A(P)$, then at least one of the following occurs:

i. $(k, l) \notin \Phi^-_{ij} \cup \Phi^+_{ij}$ and $w(P) > W$

ii. $k = i$ or $l = j$ and we violate one of the GUB inequalities (3.5) or (3.6), or

iii. $(k, l) = (j, i)$ and (i, j, i) would be a cycle.

Conditions i-iii all imply that $\bar{x} \notin X^{WCP}$. Therefore, $\bar{x}_{kl} = 0$ and $\bar{x}_{ij} + \bar{x}_{kl} = 1$, hence (3.22) is valid. □

It is now clear how proceed in constructing E_C. For each $(i, j) \in A$, we check each $(k, l) \neq (i, j)$ for membership in F_{ij}. If it is not a member, then we add $((i, j), (k, l))$ to

E_C. Testing if a particular $(k,l) \in \Phi_{ij}^- \cup \Phi_{ij}^+$ can be done in $\mathcal{O}(1)$. Therefore, the total work required is $\mathcal{O}(|A|^2)$. Before constructing E_C, though, we must calculate σ_{ij} for all $i,j \in V$ to define Φ_{ij}^- and Φ_{ij}^+, which also requires $\mathcal{O}(|V|\mathcal{SP}(V,A))$ work.

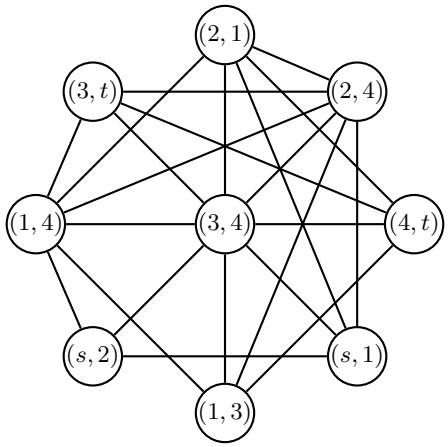

Figure 6: Conflict graph for weight constrained path example

Example 3.1. (cont'd) The conflict graph for our example is shown in Figure 6. The path $(s,1,4,t)$ is a feasible weight constrained path, and consequently the nodes in the conflict graph associated with the arcs in the path form a node packing. Note that $(3,4)$ is adjacent to every other node in the conflict graph, and can never appear in any feasible s-t path. ■

Remark 8. *Given $(i,j) \in A$ such that $j \neq s,t$, it follows from the GUB inequality (3.5) for j that $x(\phi_{ij}^+) \leq 1 - x(\delta^+(j) \setminus \phi_{ij}^+)$. Therefore, the clique inequality*

$$x(\nabla_{ij}^-) + x(\delta^+(j) \setminus \phi_{ij}^+) \leq 1$$

is implied by the strengthened node precedence inequality (3.17) since $\nabla_{ij}^- \cup (\delta^+(j) \setminus \phi_{ij}^+)$ is a clique in the conflict graph.

Conflict graphs can be large and dense. Even our trivial example has a density of 61.1% which is significant because the work and storage space required is $\mathcal{O}(|A|^2)$. Hence, it is often impractical and computationally burdensome to generate and store the entire graph G_C. We follow the approach of Hoffman and Padberg [66]. Rather than build the entire conflict graph once, we rebuild only the relevant part of it for each $x^* \in [0,1]^A$. Any arc

$(i,j) \in A$ with $x^*_{ij} = 0$ will never contribute to a violation of a clique or lifted odd hole inequality, and we ignore it altogether. This is typically the majority of the variables in the problem. If $x^*_{ij} = 1$, we can also ignore it because any adjacent node (k,l) necessarily satisfies $x^*_{kl} = 0$. Therefore, in building our relevant conflict graph G^*_C, we only need to consider the fractional arcs in A^*. This is nice because constructing each G^*_C now requires $\mathcal{O}(|A^*|^2)$ work and storage space which is significantly less. We must still calculate σ_{ij} for all $i,j \in V$ to define Φ^-_{ij} and Φ^+_{ij} which requires $\mathcal{O}(|V|\mathcal{SP}(V,A))$ work up front, but this needs to be done only once.

This reduction in work and storage space does come at a cost. In our greedy separation routine for maximum weight cliques, we are always assured maximal cliques because (i,j) is added to the clique even if $x^*_{ij} = 0$. Cliques in G^*_C may not be maximal in G_C and may not define facets of $\text{conv}(X^{NP})$. There is no effect on finding most violated odd holes, however, we cannot lift in any variable that is not fractional for lifted odd hole inequalities. While the resulting inequalities may not be as strong, there are instances of X^{WCP} where building G_C is practically impossible, and working with G^*_C becomes our only option.

CHAPTER 4

ACYCLIC WEIGHT CONSTRAINED PATHS

4.1 Introduction

In this chapter, we continue our pursuit of valid inequalities for the feasibility set of the weight constrained shortest path problem (WCSPP), however, we restrict ourselves to a special case.

Assumption (A1). *The underlying graph $G = (V, A)$ is acyclic.*

Therefore, we define the feasibility set to be

$$X^{WCP} = \left\{ x \in \mathbb{B}^A : \text{subject to (3.1) and (3.3)} \right\},$$

because the subtour elimination inequalities (3.2) are no longer needed to forbid cycles. All of the inequalities introduced in Chapter 3 remain valid. We still allow $W \in \mathbb{R}$ and $w_{ij} \in \mathbb{R}$ for all $(i, j) \in A$ (non-negativity and integrality are not required), but there is no longer a need for the assumption that G does not contain negative weight cycles to calculate shortest weight paths in polynomial time. In fact, we can replace $\mathcal{O}(\mathcal{SP}(V, A))$ in all complexity statements with $\mathcal{O}(|A|)$ since we can find shortest paths more quickly in an acyclic graph using a topological ordering of the nodes in V [1].

We also make the following assumption, which can always be enforced by implementing the preprocessing scheme of Aneja et al. [2].

Assumption (A2). *$\sigma_{sk} + \sigma_{kt} \leq W$ for all $k \in V$, and $\sigma_{si} + w_{ij} + \sigma_{jt} \leq W$ for all $(i, j) \in A$.*

If G is acyclic, a walk is always a path since no nodes can be revisited in G. Hence, if $Q = (i_1, \ldots, i_p)$ is an s-t subpath in G, then $P = Q_{s,i_1} \cup Q \cup Q_{i_p,t}$ is an s-t path (not a walk) for any s-i_1 path Q_{s,i_1} and i_p-t path $Q_{i_p,t}$ in G. Therefore, concatenating shortest paths always yields a shortest path, and (A2) guarantees that each node and arc in G appears in a weight feasible s-t path in G.

The fact that all walks are paths also ensures that the inequalities introduced in Chapter 3 are stronger in some sense. If an arc is not found in the sets used for defining these inequalities, e.g., ϕ^+_{ij}, $\phi^+_Q(i_k)$ and Φ^+_{ij}, then we can say with certainty that it is infeasible. In the cyclic case, inclusion in these sets does not guarantee feasibility because they may include arcs which can appear in a weight feasible walk, but cannot appear in a weight feasible path. To prevent this, we were forced to remove some of the arcs that may cause cycles. In the acyclic case, these sets alone give a proof of feasibility since walks are forbidden.

In the remainder of this chapter, we present additional results for the acyclic case. We begin with a result that defines the subgraph induced by all s-t paths that contain a given subpath. Next, we introduce a new class of valid inequalities. We show the separation problem for this class is $\mathcal{NP}$-hard and provide two heuristic separation routines. We also show that these inequalities are facet defining for a lower dimensional relaxation, and we give a polynomial time sequential lifting algorithm to strengthen them and arrive at facet defining inequalities for the full dimensional relaxation. Finally, we give the dimension of $\text{conv}(X^{WCP})$, and show that the strengthened node precedence inequality (3.17) is facet defining for a projection of $\text{conv}(X^{WCP})$.

4.2 Fixing Subpaths in G

There are situations, such as during preprocessing or at a node in a branch-and-bound tree, where we might want to consider G with the added restriction that no s-t path in G may contain some arc $(i,j) \in A$ or some node $k \in V$. This can be accomplished by removing the arc (i,j) or by removing the node k and the set of incident arcs $\delta^-(k) \cup \delta^+(k)$ from G, respectively. This is convenient because it introduces no new complexities to the problem or its structure, such as added constraints.

We may also want to consider the opposite type of restriction, where all s-t paths in G must contain some arc $(i,j) \in A$ or some node $k \in V$. Fortunately, if G is acyclic, there is also a way to enforce this type of restriction solely by removing appropriate nodes and arcs from G. In fact, we can enforce a more general restriction which requires that all s-t paths in G must contain some s-t subpath $Q = (i_1, \ldots, i_p)$. This is clearly a generalization since

$p = 1$ implies that Q is a node and $p = 2$ implies that Q is an arc. We begin by giving a necessary condition for any s-t path containing Q.

Definition 4.1. Given an s-t subpath $Q = (i_1, \ldots, i_p)$ in G, let

$$V[Q] = \rho^-(i_1) \cup V(Q) \cup \rho^+(i_p)$$

and

$$A[Q] = \gamma(\rho^-(i_1)) \cup A(Q) \cup \gamma(\rho^+(i_p)).$$

Proposition 4.2.1. *P is an s-t path in G containing subpath Q only if P is an s-t path in $G[Q] = (V[Q], A[Q])$.*

Proof. Let P be an s-t path in G containing $Q = (i_1, \ldots, i_p)$. Then $P = Q' \cup Q \cup Q''$ for some s-i_1 path Q' and i_p-t path Q'' in G. Since i_1 can be reached from each node in Q', then $V(Q') \subseteq \rho^-(i_1)$ and $A(Q') \subseteq \gamma(\rho^-(i_1))$. Similarly, $V(Q'') \subseteq \rho^+(i_p)$ and $A(Q'') \subseteq \gamma(\rho^+(i_p))$. Therefore, $V(P) \subseteq V[Q]$ and $A(P) \subseteq A[Q]$ and P is an s-t path in $G[Q]$. □

Remark 9. *If $Q = (i, j)$, then $\Phi_{ij}^- \cup (i, j) \cup \Phi_{ij}^+ \subseteq A[Q]$.*

The result in Proposition 4.2.1 actually holds for any graph since the fact that G is acyclic is not used in the proof. Because G is acyclic, we can also show that this condition is sufficient.

Lemma 4.2.2. *Let $Q = (i_1, \ldots, i_p)$ be an s-t subpath in G. Then,*

i. $\rho^-(i_1) \cap \rho^+(i_p) = \{i_1\} \cap \{i_p\}$,

ii. $\rho^-(i_1) \cap V(Q) = \{i_1\}$, *and*

iii. $V(Q) \cap \rho^+(i_p) = \{i_p\}$.

Proof. i. Clearly, $\rho^-(i_1) \cap \rho^+(i_p) \supseteq \{i_1\} \cap \{i_p\}$. Now assume $j \in \rho^-(i_1) \cap \rho^+(i_p)$. Then there exists a j-i_1 walk W and i_p-j walk W' in G. If $j \notin \{i_1\} \cap \{i_p\}$, then $W \cup Q \cup W'$ is a cycle in G which contradicts the assumption that G is acyclic. Hence, $j \in \{i_1\} \cap \{i_p\}$ and $\rho^-(i_1) \cap \rho^+(i_p) \subseteq \{i_1\} \cap \{i_p\}$. *ii.* If $i_k \in \rho^-(i_1)$ for $1 < k \leq p$, then there exists a i_k-i_1 walk W in G, and $W \cup (i_i, \ldots, i_k)$ is a cycle in G. *iii.* If $i_k \in \rho^+(i_p)$ for $1 \leq k < p$, then there exists a i_p-i_k walk W in G, and $(i_k, \ldots, i_p) \cup W$ is a cycle in G. □

Proposition 4.2.3. *P is an s-t path in G containing subpath Q if P is an s-t path in $G[Q] = (V[Q], A[Q])$.*

Proof. Let $Q = (i_1, \ldots, i_p)$ be an s-t subpath in G, and let P be an s-t path in $G[Q]$. $G[Q]$ is a subgraph of G by definition, therefore P is an s-t path in G. All that remains to be shown is P contains Q.

It follows from Lemma 4.2.2 and the definition of $A[Q]$ that removing i_1 disconnects $G[Q]$ (which is connected by definition) into two components with node sets $\rho^-(i_1) \setminus \{i_1\}$ and $(V(Q) \cup \rho^+(i_p)) \setminus \{i_1\}$. Hence, any path in $G[Q]$ with one end in $\rho^-(i_1)$ and the other in $\rho^+(i_p)$ must contain i_1. We know that $s \in \rho^-(i_1)$ and $t \in \rho^+(i_p)$ because Q is an s-t subpath. Therefore, $i_1 \in V(P)$ since P is an s-t path in $G[Q]$.

If $p = 1$, we are done since P contains $Q = (i_1)$. Now, suppose $p > 1$. It follows from Lemma 4.2.2 and the definition of $A[Q]$ that removing (i_k, i_{k+1}) for some $k = 1, \ldots, p-1$ disconnects $G[Q]$ into two components with node sets $\rho^-(i_1) \cup \{i_1, \ldots, i_k\}$ and $\{i_{k+1}, \ldots, i_p\} \cup \rho^+(i_p)$. Hence, any path in $G[Q]$ with one end in $\rho^-(i_1)$ and the other in $\rho^+(i_p)$ must contain (i_k, i_{k+1}). Therefore, $(i_k, i_{k+1}) \in A(P)$ and $i_{k+1} \in V(P)$ for all $k = 1, \ldots, p-1$, and P contains Q. □

For any s-t subpath $Q = (i_1, \ldots, i_p)$ in G, $G[Q]$ is the union of three subgraphs of G: $(\rho^-(i_1), \gamma(\rho^-(i_1)))$, Q, and $(\rho^+(i_p), \gamma(\rho^+(i_p)))$. Lemma 4.2.2 tells us that if G is acyclic these subgraphs are pairwise arc disjoint, and may only intersect at the nodes i_1 and i_p (see Figure 7). Propositions 4.2.1 and 4.2.3 define for us the exact set of nodes $V \setminus V[Q]$ and arcs $A \setminus A[Q]$ which must be removed in order to guarantee that all s-t paths in G must contain the subpath Q. The sets $\rho^-(i_1)$ and $\rho^+(i_p)$ can be found in $\mathcal{O}(|A|)$ using breadth-first search. Therefore, the total work required to define $G[Q]$ is $\mathcal{O}(|A|)$.

Remark 10. *It suffices to remove the arcs in $A \setminus A[Q]$. Any node in $V \setminus V[Q]$ would be a component that is disconnected from the rest of the graph $(V, A[Q])$ and cannot be found in any s-t path.*

If G contains cycles, $\rho^-(i_1)$ and $\rho^+(i_p)$ may share nodes in common other than i_1 and i_p, and there may be an s-t path which bypasses Q altogether. However, Proposition 4.2.1

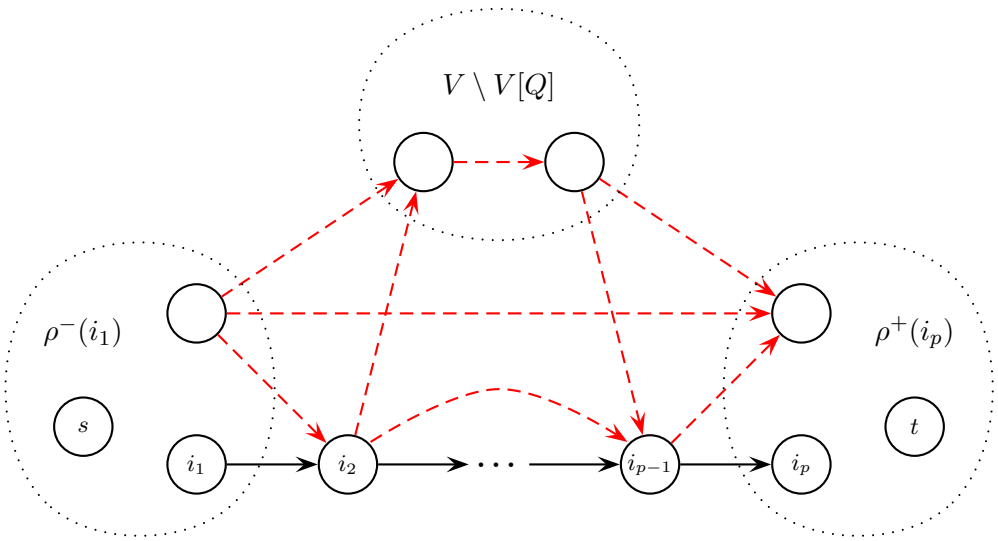

Figure 7: Candidate arcs to be removed for $A[Q]$

still gives us an $\mathcal{O}(|A|)$ preprocessing scheme that eliminates nodes and arcs. We still have to add some sort of constraint to X^{WCP} such as $x(Q) = |A(Q)|$ to ensure all s-t paths contain Q, but we will have reduced the size of the graph we must work on. Of course, this preprocessing will be weaker as the size of A gets larger, and $G = G[Q]$ if G is complete. Note that if $p > 1$ any node $i_k \in V(Q)$ may also be found in $\rho^-(i_1)$ or $\rho^+(i_p)$. If that happens, some of the arcs in $\delta^-(i_k) \cup \delta^+(i_k)$ which we know will never appear in any s-t path in G containing Q may be included in $\gamma(\rho^-(i_1))$ or $\gamma(\rho^+(i_p))$. Therefore, before defining $G[Q]$, we should remove the following arc sets:

- $\delta^+(i_k) \setminus \{(i_k, i_{k+1})\}$ for all $k = 1, \ldots, p-1$,
- $\delta^-(i_k) \setminus \{(i_{k-1}, i_k)\}$ for all $k = 2, \ldots, p$, and
- (i_p, i_1).

Of course, nothing prevents us from requiring that all s-t paths in G must contain a set of disjoint s-t subpaths. We can iteratively apply the procedure discussed above for each of the subpaths in the set.

Definition 4.2. Given a set of disjoint s-t subpaths $Q_1, \ldots, Q_K$ in G, let

$$G[Q_1, \ldots, Q_K] = (V[Q_1, \ldots, Q_K], A[Q_1, \ldots, Q_K])$$

where

$$V[Q_1, \ldots, Q_K] = \bigcap_{k=1}^{K} V[Q_k] \quad \text{and} \quad A[Q_1, \ldots, Q_K] = \bigcap_{k=1}^{K} A[Q_k].$$

For convenience, when each Q_k is an arc in some set $E = \{(i_1, j_1), \ldots, (i_K, j_K)\} \subseteq A$, we will use the notation $G[E]$ rather than $G[(i_1, j_1), \ldots, (i_K, j_K)]$. It should be noted that if there does not exist an s-t path in G which contains all of the subpaths, the resulting graph will be empty. Of course, the total work required to find $G[Q_1, \ldots, Q_K]$ is $\mathcal{O}(K|A|)$.

4.3 Path Subset Cover Inequalities

The ability to define subgraph $G[Q_1, \ldots, Q_K]$ for a set of disjoint s-t subpaths $Q_1, \ldots, Q_K$ in G, allows us to calculate the value of the shortest i-j path in G that must contain all of the $Q_1, \ldots, Q_K$. This leads us to a new class of valid inequalities for X^{WCP}. In this section, we introduce these inequalities which improve upon knapsack cover inequalities and the infeasible s-t subpath inequalities (3.10).

4.3.1 Valid Inequalities

For any set of arcs $E \subseteq A$ such that $w(E) > W$, the standard knapsack cover inequality for X^{WCP} would be

$$x(E) \leq |E| - 1.$$

However, this inequality may be weak if there are arcs in E that can never appear in any s-t path together. If all of the arcs in E do appear in some s-t path in G, we still must include all of the arcs in E in the inequality which can make the right hand side large even if E is a minimal cover.

To address these issues we could specialize our notion of a cover for X^{WCP}. While deriving the subpath precedence inequalities in Section 3.3, we introduced the inequality

$$x(Q) \leq |A(Q)| - 1$$

which is valid for X^{WCP} for any infeasible s-t subpath $Q = (i_1, \ldots, i_p)$ in G such that $\sigma_{s,i_1} + w(Q) + \sigma_{i_p,t} > W$. This notion of a subpath as a cover improves upon the knapsack cover because it does not require $w(Q) > W$, which results in a smaller cover (and right

hand side). But there are drawbacks to this inequality as well. A fractional flow of $\frac{|A(Q)|-1}{|A(Q)|}$ along each arc in Q is allowable, and that value can be close to one for any reasonably sized Q. Even if Q is minimal infeasible, there may be arcs in the middle of the subpath which have relatively small weights compared to the ends or are unnecessary in the inequality. Therefore, we continue specializing our notion of a cover by considering arc subsets of Q.

If we remove arc (i_k, i_{k+1}) from Q for some $k = 2, \ldots, p-2$, it seems like we would have to calculate $\sigma_{i_k,j}$ and $\sigma_{j,i_{k+1}}$ for all $j \in V$ to determine the feasibility of passing through both subpaths $(i_i, \ldots, i_k)$ and $(i_{k+1}, \ldots, i_p)$. Fortunately, we know how to define the subgraph of G which guarantees that all s-t paths in G must contain these subpaths, and we can test their feasibility in this subgraph with a single shortest path calculation. For any set of disjoint s-t subpaths $Q_1, \ldots, Q_K$ in G, let $\sigma_{ij}[Q_1, \ldots, Q_K]$ be the value of the shortest i-j path in $G[Q_1, \ldots, Q_K]$ with arc lengths given by the weight vector w. Note that $\sigma_{st}[Q] > W$ implies the condition $\sigma_{s,i_1} + w(Q) + \sigma_{i_p,t} > W$ above.

Definition 4.3. A set of arcs $E \subseteq A$ is an *s-t path subset* if $E \subseteq A(P)$ for some s-t path P in G. An s-t path subset is a *cover* if $\sigma_{st}[E] > W$. An s-t path subset cover is *minimal* if $E \setminus \{(i,j)\}$ is not a cover for all $(i,j) \in E$.

Definition 4.4. Given an s-t path subset cover E, the *path subset cover inequality* is

$$x(E) \leq |E| - 1. \tag{4.1}$$

Proposition 4.3.1. *The path subset cover inequality (4.1) is valid for X^{WCP}.*

Proof. Let E be an s-t path subset cover in G. Assume $\bar{x} \in X^{WCP}$, and let P be its corresponding s-t path in G. Suppose $\bar{x}(E) > |E| - 1$. Then, $\bar{x}(E) = |E|$, $E \subseteq A(P)$ and P is an s-t path in $G[E]$. Hence, $w(P) \geq \sigma_{st}[E] > W$ and $\bar{x} \notin X^{WCP}$. Therefore, $\bar{x}(E) \leq |E| - 1$ and (4.1) is valid. □

Example 4.1. Consider the weight constrained path set in Figure 8. The arc set $E_1 = \{(2,1), (1,3), (3,t), (4,t)\}$ is a knapsack cover with $w(E_1) = 11 > 10$, but inequality

$$x_{21} + x_{13} + x_{3t} + x_{4t} \leq 3$$

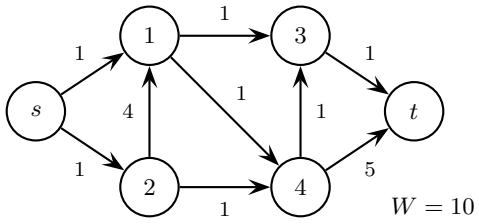

Figure 8: Constrained path example 2

is redundant since $x_{3t} + x_{4t} \leq 1$. The arc set $E_2 = \{(s,2),(2,1),(1,4),(4,t)\}$ is a minimal knapsack cover with $w(E_2) = 11 > 10$, and

$$x_{s2} + x_{21} + x_{14} + x_{4t} \leq 3$$

prohibits the only infeasible s-t path in the graph $(s,2,1,4,t)$. The s-t subpath $Q = (2,1,4,t)$ is minimal infeasible with $\sigma_{s2} + w(Q) + \sigma_{tt} = 11 > 10$, and

$$x_{21} + x_{14} + x_{4t} \leq 2$$

is stronger than the inequality for E_2. If we remove $(1,4)$ from Q, we get the s-t path subset $E_3 = \{(2,1),(4,t)\}$. It is clear that $G[E_3] = (s,2,1,4,t)$ and $\sigma_{st}[E_3] = 11 > 10$. Therefore, E_3 is a path subset cover and we get the inequality

$$x_{21} + x_{4t} \leq 1$$

which is stronger than the inequalities for E_1, E_2 and Q. ■

Remark 11.

i. By (A2), we know that $|E| > 1$.

ii. If $|E| = 2$, we get the conflict graph inequality (3.22). Hence, if there exists some clique $C \supset E$ in the conflict graph for X^{WCP}, inequality (4.1) is implied by some clique inequality (3.19).

iii. If E induces a subpath, (4.1) is implied by the subpath precedence inequalities (3.12) and (3.13).

iv. If E is minimal, we can show that (4.1) is facet-defining for a lower dimensional relaxation of X^{WCP} (see Proposition 4.4.1).

4.3.2 Separation

To separate inequality (4.1) for a fractional solution $x^* \in [0,1]^A$, we want to find an s-t path subset cover such that $x^*(E) > |E| - 1$. This inequality can be restated as

$$\sum_{(i,j)\in E} (1 - x^*_{ij}) < 1.$$

Therefore, we can define the separation problem as follows:

Problem (PSC-SEP). Path Subset Cover Separation

Instance: We are given an acyclic directed graph $G = (V, A)$, a fractional solution $x^* \in [0,1]^A$, arc weights $w_{ij} \in \mathbb{R}$, specified vertices $s, t \in V$, and capacity $W \in \mathbb{R}$.

Question: Is there an s-t path subset E in G such that $\sigma_{st}[E] > W$ and $\sum_{(i,j)\in E}(1-x^*_{ij}) < 1$?

Proposition 4.3.2. *PSC-SEP is $\mathcal{NP}$-hard.*

Proof. We transform from Knapsack Cover Separation (KC-SEP) which has been shown to be $\mathcal{NP}$-hard [42].

Instance: We are given a set $N = \{1, \ldots, n\}$, a fractional solution $y^* \in [0,1]^N$, weights $a_j \in \mathbb{Z}_+$ for each $j \in N$, and positive integer B.

Question: Is there a subset $C \subseteq N$ such that $a(C) > B$ and $\sum_{j\in C}(1 - y^*_j) < 1$?

Let $W = B$ and $V = \{s, 1, 1', 2, 2', \ldots, n, n', t\}$. We construct acyclic $G = (V, A)$ such that A is the union of the arc sets $\{(s,1),(s,1'),(n,t),(n',t)\}$, $\{(j, j+1) : j \in N \setminus \{n\}\}$, $\{(j,j') : j \in N\}$, $\{(j', j+1) : j \in N \setminus \{n\}\}$ and $\{(j', (j+1)') : j \in N \setminus \{n\}\}$. For all $j \in N$, let $x^*_{jj'} = y^*_j$ and $w_{jj'} = a_j$. For all remaining arcs $(i,j) \in A$, let $x^*_{ij} = 1$ and $w_{ij} = 0$.

By construction,

$$\sum_{(i,j)\in E} (1 - x^*_{ij}) = \sum_{(j,j')\in E} (1 - x^*_{jj'}) = \sum_{(j,j')\in E} (1 - y^*_j)$$

and

$$\sigma_{st}[E] = w(E) = \sum_{(j,j')\in E} w_{jj'} = \sum_{(j,j')\in E} a_j$$

for any s-t path subset E in G, since $w_{ij} \in \mathbb{Z}_+$ and $\sigma_{jk} = 0$ for all $j, k \in V$ such that $k \in \rho^+(j)$ and $k \neq j'$. Therefore, if E is a feasible solution to PSC-SEP on G, then

$C = \{j \in N : (j, j') \in E\}$ is a feasible solution to KC-SEP. Conversely, if $C = \{j_1, \ldots, j_{|C|}\}$ is a feasible solution to KC-SEP, then the s-t path subset $E = \left\{(j_1, j'_1), \ldots, (j_{|C|}, j'_{|C|})\right\}$ is a feasible solution to PSC-SEP. □

Remark 12. *IS-SEP, as defined in Section 3.3, is at least as hard as PSC-SEP. We could have shown that IS-SEP is $\mathcal{NP}$-hard by transforming from PSC-SEP.*

Because this separation problem is hard, we have developed two separation heuristics for inequality (4.1). The first heuristic attempts to find a maximally violated path subset cover in a greedy fashion, and is presented in Algorithm 5.

Algorithm 5 Heuristic for finding a maximally violated inequality (4.1) for $x^* \in [0,1]^A$

```
PATH_SUBSET_COVER_HEUR( ):
λ = σ_st, E ← ∅
while λ ≤ W do
    define G[E] = (V[E], A[E])
    find σ_si[E] and σ_it[E] for all i ∈ V[E]
    let λ_ij ← σ_si[E] + w_ij + σ_jt[E] − σ_st[E] for all (i,j) ∈ A[E]
    let A*_λ ← {(i,j) ∈ A[E] \ E : x*_ij > 0, λ_ij > 0}
    if A*_λ = ∅ then
        return ∅
    end if
    (i',j') ← arg min {(1 − x*_ij)/λ_ij : (i,j) ∈ A*_λ}
    λ ← λ_{i'j'} + σ_st[E], E ← E ∪ (i',j')
end while
E ← MINIMAL_PATH_SUBSET_COVER(E)
return E
```

We follow a similar scheme to the approach Crowder et al. [22] used in separating ordinary knapsack covers. We insert arcs one by one into the path subset E, choosing the best available arc remaining in $A[E] \setminus E$. After each insertion, we check if E is a cover, i.e., $\lambda = \sigma_{st}[E] > W$. If a cover E is found that is not minimal, then there exists a cover $E' \subset E$ whose corresponding inequality (4.1) implies the inequality for E. Therefore, we remove arcs heuristically from the cover in non-decreasing order of x^*_{ij} until it is minimal before returning E.

In the context of WCSPP, Crowder et al. would insert arcs into the subset in non-decreasing order of the ratio $(1 - x^*_{ij})/w_{ij}$ until a cover was obtained. The denominator

in this ratio, w_{ij}, is a natural choice for knapsack covers because it directly measures the contribution of adding arc (i, j) to the infeasibility of E. For path subset covers, however, the contribution of (i, j) is given by the change in the value of the shortest weight s-t path, or

$$\lambda_{ij} = \sigma_{st}[E \cup \{(i, j)\}] - \sigma_{st}[E].$$

Recall that G is acyclic and concatenating a shortest s-i path and j-t path in $G[E]$ always yields a shortest path containing (i, j) since they cannot share any nodes in common. Hence,

$$\sigma_{st}[E \cup \{(i, j)\}] = \sigma_{si}[E] + w_{ij} + \sigma_{jt}[E].$$

We use λ_{ij} (which changes at each iteration) as the denominator for our ratio. Any arc (i, j) with $\lambda_{ij} = 0$ will not contribute to the infeasibility and is never included in E. Similarly, any (i, j) with $x^*_{ij} = 0$ is also ignored because it will not contribute to the violation of (4.1). As a consequence, the existence of a cover is not guaranteed. If our set of candidate arcs A^*_λ is empty, we simply return with no cover found.

Let $G^* = (V^*, A^*)$ be the support graph for $x^* \in [0, 1]^A$. The work done at each iteration consists of defining $G[E]$, calculating the shortest weight s-t path, and choosing the next arc to be added to E. Defining $G[E]$ on line 4 can be done in $\mathcal{O}(|A|)$ because we already have $G[E \setminus \{(i', j')\}]$, and we only need to include a single additional arc. Finding the shortest weight paths in $G[E]$ on line 5 requires $\mathcal{O}(\mathcal{SP}(V, A))$ work, and choosing the next arc on lines 6-12 can be done in $\mathcal{O}(|A^*|)$. We may iterate at most $\mathcal{O}(|V^*|)$ times because a fractional path subset may have at most $|V^*| - 1$ arcs. Then, the work required for the loop on lines 3-13 is $\mathcal{O}(|V^*|\mathcal{SP}(V, A))$. If we implement MINIMAL_PATH_SUBSET_COVER on line 14 with care, we can find a minimal cover in $\mathcal{O}(|V^*|\mathcal{SP}(V, A))$ as well. We omit the details here, but present it formally in Algorithm 6. Therefore, the total work required for the heuristic is $\mathcal{O}(|V^*|\mathcal{SP}(V, A))$.

Note that the complexity statements use $\mathcal{O}(\mathcal{SP}(V, A))$ for shortest path calculations rather than $\mathcal{O}(|A|)$ even though we know G is acyclic. The reason is that path subset cover inequalities can be used in the cyclic case as well (assuming no negative cycles in w). For any path subset E, we can approximate $G[E]$ as discussed in Section 4.2. There may be

Algorithm 6 Heuristic for finding a minimal path subset cover for $x^* \in [0,1]^A$

```
1: MINIMAL_PATH_SUBSET_COVER(E):
2: A_O ← A, A_M ← A, E_M ← ∅
3: sort E such that x*_{i1,j1} ≤ x*_{i2,j2} ≤ ··· ≤ x*_{ip,jp}
4: for k = p, ..., 1 do
5:    A[k] ← A_O \ A_O[(i_k, j_k)]
6:    A_O ← A_O[(i_k, j_k)]
7: end for
8: for k = 1, ..., p do
9:    A_O ← A_O ∪ A[k]
10:   let σ be the value of the shortest weight s-t path in (V, A_O ∩ A_M)
11:   if σ ≤ W then
12:      A_M ← A_M[(i_k, j_k)], E_M ← E_M ∪ (i_k, j_k)
13:   end if
14: end for
15: return E_M
```

s-t paths in $G[E]$ which do not contain E, but all paths that do are included. Therefore, $\sigma_{st}[E]$ underestimates the true value of the shortest weight s-t path containing E, and inequality (4.1) is still valid. Of course, the approximation of $G[E]$ deteriorates as the size of A increases, and the inequalities will get weaker when this happens because we will need to add a greater number of arcs to E to force $\sigma_{st}[E] > W$.

While our first heuristic tries to find the most violated inequality, it only finds a single inequality. The second heuristic focuses on finding many violated inequalities, and is presented in Algorithm 7.

If E is an s-t path subset cover such that $E \subseteq A(Q)$ for some s-t subpath Q in G, then Q is an infeasible s-t subpath. Our heuristic aims to find minimal infeasible s-t subpaths and remove arcs from them in non-decreasing order of x^*_{ij} until it is a minimal path subset cover. We modified the enumeration procedure used in Section 3.3 for finding subpath precedence inequalities. For each fractional arc (i_1, i_2) whose head is not t, we enumerate over all fractional s-t subpaths $Q = (i_1, i_2, \ldots, i_k)$. If an infeasible path Q is detected, we first check to see if it is minimal. If so, we find a minimal path subset cover in Q and add inequality (4.1) if violated; then we backtrack. Note that if Q is minimal infeasible, then the truncated s-t subpaths $(i_2, \ldots, i_p)$ and $(i_1, \ldots, i_{p-1})$ are feasible along with all of their respective path subsets. Hence, any path subset of Q must contain (i_1, i_2) and (i_{p-1}, i_p) to

Algorithm 7 Heuristic for finding all violated inequalities (4.1) for $x^* \in [0,1]^A$

$\mathcal{C} \leftarrow \emptyset$
for all $(i_1, i_2) \in A^*$ such that $i_2 \neq t$ **do**
  $Q \leftarrow (i_1, i_2)$
  PATH_SUBSET_COVER_DFS(2)
end for
return $\mathcal{C}$

PATH_SUBSET_COVER_DFS(k):
$\mathcal{F}[k] \leftarrow \emptyset$
for all $(i_k, j) \in \delta^+(i_k)$ such that $x^*_{i_k,j} > 0$ **do**
  if $(i_k, j) \in \phi^+_Q(i_k)$ **then**
    $\mathcal{F}[k] \leftarrow \mathcal{F}[k] \cup \{(i_k, j)\}$
  else
    if $(i_k, j) \in \phi^+_{(i_2,\ldots,i_k)}(i_k)$ **then**
      $E \leftarrow$ MINIMAL_PATH_SUBSET_COVER($A(Q) \cup (i_k, j)$)
      if $x^*(E) > |E| - 1$ **then**
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{x(E) \leq |E| - 1\}$
      end if
    end if
  end if
end for
for all $(i_k, j) \in \mathcal{F}[k]$ such that $j \neq t$ **do**
  $Q \leftarrow (i_1, \ldots, i_k, j)$
  PATH_SUBSET_COVER_DFS($k+1$)
  $Q \leftarrow (i_1, \ldots, i_k)$
end for
return

be a cover.

Most of the work is done by the recursive function PATH_SUBSET_COVER_DFS. We call this function each time we add a new node to the end of the subpath Q. When called for node i_k, we classify all fractional arcs in $\delta^+(i_k)$ as either feasible or infeasible in the loop on lines 10-21. If (i_k, j) is a feasible arc, we add it to the list of arcs $\mathcal{F}[k]$ to continue exploring out of i_k. If (i_k, j) is not feasible, we call MINIMAL_PATH_SUBSET_COVER if the subpath $(i_2, \ldots, i_k, j)$ is feasible (or $(i_1, \ldots, i_k, j)$ is minimal infeasible) to get a minimal path subset cover E. If E is a violated cover, we add inequality (4.1) to the cut pool $\mathcal{C}$. Finally, we call PATH_SUBSET_COVER_DFS for each j such that $(i_k, j) \in \mathcal{F}[k]$ and $j \neq t$.

Each test of $(i_k, j) \in \phi_Q^+(i_k)$ on line 11 or $(i_k, j) \in \phi^+_{(i_2,\ldots,i_k)}(i_k)$ on line 14 can be done in $\mathcal{O}(1)$ by updating a variable $w_Q = \sigma_{s,i_1} + w(Q)$ each time we modify Q. Finding a minimal cover with MINIMAL_PATH_SUBSET_COVER on line 15 requires $\mathcal{O}(|V^*|\mathcal{SP}(V, A))$ work. Thus, each call to the function PATH_SUBSET_COVER_DFS might require $\mathcal{O}(|V^*|^2\mathcal{SP}(V, A))$ work, and we must call it for each node in a fractional s-t subpath. Therefore, the total work required by Algorithm 7 for any enumerated subpath is $\mathcal{O}(|V^*|^3\mathcal{SP}(V, A))$, but there could be an exponential number of such subpaths. As with Algorithm 3, our hope is that the number of fractional paths in G^* is not prohibitive in practice; otherwise, we can always place a limit on the length of the subpaths to be considered. Before running Algorithm 7, though, we must calculate σ_{si} and σ_{it} for all $i \in V$ to define $\phi_Q^+(i_k)$. This requires $\mathcal{O}(\mathcal{SP}(V, A))$ work up front, but only needs to be done once.

If Algorithm 7 is used for cyclic G, care should be taken not to include backwards arcs in $\mathcal{F}[k]$. Also, we know that $\sigma_{st}[Q] \leq \sigma_{s,i_1} + w(Q) + \sigma_{i_p,t}$ in general. Hence, it may be the case that $\sigma_{st}[Q] \leq W$ even though Q is minimal infeasible because $G[Q]$ is just an approximation. If this happens and Q is a violated subpath, we add inequality (4.1) to the cut pool $\mathcal{C}$ with $E = A(Q)$; otherwise, we skip Q.

4.4 Lifted Path Subset Cover Inequalities

Strengthening valid inequalities through sequential lifting is a strategy that has appeared frequently in the literature. For example, lifted cover inequalities (LCIs) and lifted GUB

cover inequalities (LGCIs) have been used successfully in practice to solve BIPs (see [22, 58]), and both are widely used in general purpose BIP solvers. In this section, we consider sequential lifting for path subset cover inequalities.

4.4.1 Relaxing X^{WCP}

Clearly, $\dim(\text{conv}(X^{WCP})) < |A|$ because the flow balance equations (3.1) have rank $|V|-1$. However, when lifting sequentially it is convenient to work with a feasible set X such that $\text{conv}(X)$ is full-dimensional. For example, we mentioned the lifted odd hole inequality (3.21) in Section 3.6.1. This inequality defines a facet of the node packing polytope X^{NP} which is a full-dimensional relaxation of X^{WCP}. For an alternate full-dimensional relaxation of X^{WCP}, let us consider the set of all s-t path subsets E in G for which there exists some weight feasible s-t path P such that $E \subseteq A(P)$ (i.e., $\sigma_{st}[E] \leq W$). We define the *path subset polytope* as

$$X^{PS} = \left\{ x \in \mathbb{B}^A : x \leq x' \text{ for some } x' \in X^{WCP} \right\}.$$

Note that $X^{PS} \supset X^{WCP}$ since the maximal elements of X^{PS} define X^{WCP}. It follows from (A2) that each arc must appear in a weight feasible s-t path in G, and the $|A|$ linearly independent unit vectors are in X^{PS}. Since $0 \in X^{PS}$, we have $|A|+1$ affinely independent points in X^{PS}, and $\dim(\text{conv}(X^{PS})) = |A|$.

Remark 13. *It can be shown that the conflict graph inequality (3.22) is valid for X^{PS}. Therefore, $X^{NP} \supseteq X^{PS}$ and the path subset relaxation is at least as strong as the node packing relaxation.*

Since X^{PS} is the set of all s-t path subsets in G which are not covers, it is trivial to show that the path subset cover inequality (4.1) is valid for X^{PS}. In general, (4.1) does not give a facet of $\text{conv}(X^{PS})$, but we show that it is facet-defining if E is minimal and we restrict X^{PS} to the arcs in E.

Definition 4.5. For $E \subseteq A$, let $X^{PS}_E = \left\{ x \in X^{PS} : x_{ij} = 0,\ \forall\ (i,j) \in A \setminus E \right\}$.

Proposition 4.4.1. *If E is a minimal s-t path subset cover, then the path subset cover inequality (4.1) defines a facet of* $\text{conv}(X^{PS}_E)$.

Proof. We know that $0 \in X_E^{PS}$ and does not satisfy (4.1) at equality since (A2) ensures that $|E| > 1$. Hence, inequality (4.1) is proper. To prove the result, we show that there exists $|E|$ linearly independent points in X_E^{PS} satisfying (4.1) at equality.

Since E is minimal, there exists a feasible s-t path in G containing $E \setminus \{(i,j)\}$ for all $(i,j) \in E$. Therefore, $e - e^{ij} \in X_E^{PS}$ for all $(i,j) \in E$, where $e \in \mathbb{B}^E$ is the vector of all ones and $e^{ij} \in \mathbb{B}^E$ is the unit vector for arc (i,j). We form the matrix M with a row for each arc in E. The vectors $e - e^{ij}$ can be arranged in the columns of M to get

$$M = \begin{pmatrix} 0 & 1 & \dots & 1 \\ 1 & 0 & \dots & 1 \\ \vdots & \vdots & & \vdots \\ 1 & 1 & \dots & 0 \end{pmatrix} \begin{matrix} (i_1, j_1) \\ (i_2, j_2) \\ \vdots \\ (i_{|E|}, j_{|E|}). \end{matrix}$$

The square matrix M is invertible since $|E| > 1$, and the columns are linearly independent. Therefore, the set of vectors $e - e^{ij}$ for all $(i,j) \in E$ are linearly independent. Each of these $|E|$ vectors satisfies (4.1) at equality and the result follows. □

4.4.2 Sequential Lifting

By sequentially lifting the arcs in $A \setminus E$ into inequality (4.1), we get the *lifted path subset cover inequality*

$$x(E) + \sum_{(i,j) \in A \setminus E} \alpha_{ij} x_{ij} \leq |E| - 1 \tag{4.2}$$

which defines a facet of $\text{conv}(X^{PS})$ if each arc is lifted exactly. The procedure for lifting a single arc into a partially lifted inequality (4.2) follows directly from Nemhauser and Wolsey [95]. Let $F \subset A \setminus E$ be the set of arcs that have already been lifted. If $(k,l) \in A \setminus (E \cup F)$ and we define $X^{kl} = \{x \in \mathbb{B}^A : x_{kl} = 1\}$, then the lifting coefficient for (k,l) can be computed as

$$\alpha_{kl} \leq |E| - 1 - \zeta_{kl},$$

where

$$\zeta_{kl} = \max \left\{ x(E) + \sum_{(i,j) \in F} \alpha_{ij} x_{ij} : x \in X^{PS}_{E \cup F \cup \{(k,l)\}} \cap X^{kl} \right\}.$$

It follows from (A2) that $X^{PS}_{E \cup F \cup \{(k,l)\}} \cap X^{kl} \neq \emptyset$, and the lifting problem is always feasible.

Proposition 4.4.2. $0 \leq \zeta_{kl} \leq |E| - 1$.

Proof. ($\zeta_{kl} \geq 0$) Clearly, $(0,1) \in X^{PS}_{E \cup F \cup \{(k,l)\}} \cap X^{kl}$ for $0 \in \mathbb{B}^{E \cup F}$ and

$$\zeta_{kl} = \max\left\{ x(E) + \sum_{(i,j) \in F} \alpha_{ij} x_{ij} : x \in X^{PS}_{E \cup F \cup \{(k,l)\}} \cap X^{kl} \right\} \geq 0.$$

($\zeta_{kl} \leq |E| - 1$) Let $x^* \in \mathbb{B}^{E \cup F}$ such that $(x^*, 1) \in X^{PS}_{E \cup F \cup \{(k,l)\}} \cap X^{kl}$ and

$$\zeta_{kl} = x^*(E) + \sum_{(i,j) \in F} \alpha_{ij} x^*_{ij}.$$

By definition, $(x^*, 1, 0) \in X^{PS}$ for $0 \in \mathbb{B}^{A \setminus (E \cup F \cup \{(k,l)\})}$, and $(x^*, 1, 0) \leq x'$ for some $x' \in X^{WCP}$. Hence, $x^* \in X^{PS}_{E \cup F}$ since $(x^*, 0, 0) \leq (x^*, 1, 0) \leq x'$, and it follows from the validity of the partially lifted inequality (4.2) on $X^{PS}_{E \cup F}$ that $\zeta_{kl} \leq |E| - 1$. □

Corollary 4.4.3. *If we lift (k,l) exactly ($\alpha_{kl} = |E| - 1 - \zeta_{kl}$), then $0 \leq \alpha_{kl} \leq |E| - 1$.*

Solving for ζ_{kl} requires the solution of a weight constrained longest path subset (not path) problem on X^{PS}, where some of the arcs are fixed to zero. While this problem can be solved by modifying the typical node labeling algorithms, we can solve an equivalent WCSPP on X^{WCP}.

Proposition 4.4.4. *Let $F \subseteq A$ and $p \in \mathbb{R}^F$. If we define $c \in \mathbb{R}^A$ such that $c_{ij} = p_{ij}$ for all $(i,j) \in F^+ = \{(i,j) \in F : p_{ij} > 0\}$ and $c_{ij} = 0$ for all $(i,j) \in A \setminus F^+$, then*

$$\max\left\{ px : x \in X^{PS}_F \right\} = -\min\left\{ -cx : x \in X^{WCP} \right\}.$$

Proof. x' is an optimal solution to $\min\left\{ -cx : x \in X^{WCP} \right\}$ if and only if x' is an optimal solution to $\max\left\{ cx : x \in X^{WCP} \right\}$. Hence, it suffices to show that

$$\max\left\{ px : x \in X^{PS}_F \right\} = \max\left\{ cx : x \in X^{WCP} \right\}.$$

Assume x' is an optimal solution to $\max\left\{ cx : x \in X^{WCP} \right\}$, and let P be the corresponding s-t path in G. Let $E = A(P) \cap F^+$ with the corresponding incidence vector $x^E \in \mathbb{B}^F$. It follows from $E \subseteq A(P)$ that $x^E \in X^{PS}_F$, and since $E \subseteq F^+$, $p(E) = c(E) = c(P)$. Therefore,

$$\max\left\{ px : x \in X^{PS}_F \right\} \geq px^E = p(E) = c(E) = c(P) = cx' = \max\left\{ cx : x \in X^{WCP} \right\}.$$

Conversely, assume x^E is an optimal solution to $\max\left\{px : x \in X_F^{PS}\right\}$, and let E be the corresponding s-t path subset in G. By definition, $(x^E, 0) \in X^{PS}$ for $0 \in \mathbb{B}^{A \setminus F}$, and there exists an $x' \in X^{WCP}$ such that $(x^E, 0) \leq x'$. Since $c_{ij} \geq 0$ for all $(i,j) \in A$, $c(x^E, 0) \leq cx'$. We can assume $E \subseteq F^+$, otherwise $p(E \setminus \{(i,j)\}) \geq p(E)$ for some $(i,j) \in E \cap (F \setminus F^+)$. Therefore, $px^E = c(x^E, 0)$ and

$$\max\left\{px : x \in X_F^{PS}\right\} = px^E = c(x^E, 0) \leq cx' \leq \max\left\{cx : x \in X^{WCP}\right\}.$$

□

Using this result, we can express the exact lifting coefficient for (k,l) as

$$\alpha_{kl} = |E| - 1 + \min\left\{-x(E) + \sum_{(i,j) \in F} -\alpha_{ij} x_{ij} : x \in X^{WCP} \cap X^{kl}\right\}.$$

The set of feasible paths for the WCSPP must exclude all s-t paths in G that do not contain (k,l), but this can be enforced with no additional constraints in $\mathcal{O}(|A|)$ by removing all arcs in $A \setminus A[(k,l)]$. If G contained cycles, this problem would be strongly $\mathcal{NP}$-hard because the arc costs are all non-positive and negative cycles are possible [14]. But since G is acyclic, we can solve the problem in pseudopolynomial time $\mathcal{O}(|A|W)$ if we assume all weights are non-negative integers by modifying the labeling algorithm of [29] and extending the node labels in topological order. In fact, this assumption is unnecessary and we can solve the problem in polynomial time $\mathcal{O}(|A||E|)$ since we need at most $|E|$ labels for each node in the labeling algorithm ($\zeta_{kl}, \alpha_{kl} \in \{0, \ldots, |E| - 1\}$), and $|E|$ is polynomial in $|V|$.

To get exact lifting coefficients for all of the remaining variables in (4.1), we must solve a sequence of $|A \setminus E|$ WCSPPs. But for computational purposes, we can use a lower bound for these problems to get an approximate α_{kl} which is still valid. While there are many ways to arrive at such a lower bound, we mention three viable options.

LP Relaxation Although this method for computing lifting coefficients involves a sequence of auxiliary LPs to produce a single lifted inequality, each LP is obtained from the previous one by changing a single objective coefficient and two variable bounds. If we use a simplex solver with warm start capability, the hope is that the computational effort is not excessive and only a few iterations will be necessary to restore optimality.

Lagrangean Relaxation Another option is to dualize the knapsack constraint and solve the Lagrangean dual problem. From the work of Ribeiro and Minoux [101], it can be shown that the Lagrangean dual can be solved in worst case complexity $\mathcal{O}(|V||A||E|)$ (by reversing the roles of the objective function and resource constraint) using a parametric approach. While this may require more work than actually solving the problem, the main advantage of this approach is that we can terminate before solving the Lagrangean dual to optimality and still obtain a valid lower bound. If we have an iteration limit, we only need to solve a series of shortest path problems for each variable lifted. Note that early termination is not possible with the exact labeling algorithm since it is a primal method. This approach should give a lower bound that is no better than that of the LP relaxation since the Lagrangean relaxation is an unconstrained shortest path problem which has integral solutions.

K **Shortest Paths** This option continues in the spirit of early termination. In linear time $\mathcal{O}(|A| + K)$, we can list the K s-t paths in $G[(k, l)]$ with minimum total cost in non-decreasing order since G is acyclic [37]. If we choose a suitable K which is large enough, the value of the Kth path should provide a lower bound that is strong enough.

4.4.3 Topological Lifting

While all of the approaches for sequential lifting discussed above are viable, none of them exploit the relationship between consecutive lifting problems. Zemel [116] presented a dynamic programming algorithm which computes exact lifting coefficients for LCIs and takes advantage of the sequential nature of the lifting problems. Gu [57] uses a similar scheme along with GUB-wise lifting for LCGIs where variables in the same GUB constraint are lifted consecutively. We adapt the approach of Gu to lifted path subset cover inequalities and restrict ourselves to *topological lifting*, i.e., lifting sequences in which all outgoing arcs for the same node are lifted consecutively, and the nodes are evaluated in topological order.

Let $s = i_1, \ldots, i_{|V|} = t$ be a topological ordering of the nodes in V, and let F_k be the set of arcs in $(A \setminus E) \cap (\delta^+(i_1) \cup \cdots \cup \delta^+(i_{k-1}))$ that have already been lifted. If we evaluate node i_k, the efficient solution of the lifting coefficient for some $(i_k, l) \in \delta^+(i_k)$ is based on the observation that we can split this problem into two simpler WCSPPs (that can be solved

by dynamic programming) and combine the two solutions. The 'source' WCSPP calculates the minimum weight s-i_k path in G for a fixed value of $\zeta_{i_k,l} = z$. For $0 \leq z \leq |E|-1$, define $y_k^-(z)$ as follows

$$
\begin{aligned}
y_k^-(z) = \min \quad & \sum_{(i,j)\in A} w_{ij}x_{ij} \\
\text{s.t.} \quad & x(\delta^+(i)) - x(\delta^-(i)) = \begin{cases} 1, & i = s \\ -1, & i = i_k \\ 0, & \forall\ i \in V \setminus \{s, i_k\} \end{cases} \\
& -x(E) + \sum_{(i,j)\in F_k} -\alpha_{ij}x_{ij} \leq -z \\
& x_{ij} \in \{0,1\},\ \forall\ (i,j) \in A.
\end{aligned}
$$

The 'sink' WCSPP calculates the minimum weight l-t path in G for a fixed value of $\zeta_{i_k,l} = z$. For $0 \leq z \leq |E|-1$, define $y_l^+(z)$ as follows

$$
\begin{aligned}
y_l^+(z) = \min \quad & \sum_{(i,j)\in A} w_{ij}x_{ij} \\
\text{s.t.} \quad & x(\delta^+(i)) - x(\delta^-(i)) = \begin{cases} 1, & i = l \\ -1, & i = t \\ 0, & \forall\ i \in V \setminus \{l, t\} \end{cases} \\
& -x(E) \leq -z \\
& x_{ij} \in \{0,1\},\ \forall\ (i,j) \in A.
\end{aligned}
$$

If we define $y_{kl}(z)$ for $0 \leq z \leq |E|-1$ as

$$
y_{kl}(z) = \min\left\{y_k^-(z_1) + y_l^+(z_2) : z_1 \geq 0,\ z_2 \geq 0,\ z_1 + z_2 = z\right\},
$$

then it is easy to see that $\alpha_{i_k,l} = |E| - 1 - \max\{z : y_{kl}(z) \leq W - w_{i_k,l}\}$. The procedure for finding all of the topological lifting coefficients for the lifted path subset cover inequality (4.2) is presented in Algorithm 8.

This procedure is a modified labeling algorithm where two sets of labels are kept and the lifting coefficients (or arc costs) for $(i_k, l) \in \delta^+(i_k)$ are obtained as we evaluate node i_k. The 'sink' labels $y_l^+(z)$ for all $l \in V$ can be computed up front using a typical labeling

Algorithm 8 Topological lifting for inequality (4.2)

1: $y_1^-(0) \leftarrow 0$, $y_1^-(z) \leftarrow \infty$ for $z = 1, \ldots, |E| - 1$
2: $y_k^-(z) \leftarrow \infty$ for $k = 2, \ldots, |V|$ and $z = 0, \ldots, |E| - 1$
3: compute $y_l^+(z)$ for all $l \in V$ and $z = 0, \ldots, |E| - 1$
4: **for** $k = 1, \ldots, |V| - 1$ **do**
5: **for all** $(i_k, l) \in \delta^+(i_k) \cap (A \setminus E)$ **do**
6: $y_{kl}(z) \leftarrow \min\left\{y_k^-(z_1) + y_l^+(z_2) : z_1 \geq 0,\ z_2 \geq 0,\ z_1 + z_2 = z\right\}$ for $z = 0, \ldots, |E| - 1$
7: $\alpha_{i_k,l} \leftarrow |E| - 1 - \max\left\{z : y_{kl}(z) \leq W - w_{i_k,l}\right\}$
8: let $h > k$ be such that $i_h = l$
9: $y_h^-(z) \leftarrow \min\left\{y_h^-(z),\ y_k^-(z - \alpha_{i_k,l}) + w_{i_k,l}\right\}$ for $z = \alpha_{i_k,l}, \ldots, |E| - 1$
10: **end for**
11: **for all** $(i_k, l) \in \delta^+(i_k) \cap E$ **do**
12: let $h > k$ be such that $i_h = l$
13: $y_h^-(z) \leftarrow \min\left\{y_h^-(z),\ y_k^-(z - 1) + w_{i_k,l}\right\}$ for $z = 1, \ldots, |E| - 1$
14: **end for**
15: **end for**

algorithm. These labels represent all of the non-dominated l-t paths in G when no arcs reachable from l have been lifted. The 'source' labels $y_k^-(z)$ for $k = 1, \ldots, |V| - 1$ are maintained as the algorithm progresses. These labels represent all of the non-dominated s-i_k paths in G when all arcs on these paths have been lifted. We begin with no source labels on any node except for $y_1^-(0) = 0$ at node $s = i_1$. To evaluate a node, we extend the set of source labels at i_k to the set of source labels at l for each outgoing arc (i_k, l) by augmenting the corresponding s-i_k path with (i_k, l). If $(i_k, l) \in E$, we know the arc cost is one; otherwise, we must calculate the lifting coefficient. Clearly, the optimal path for the lifting problem will be the concatenation of a non-dominated s-i_k path in which all of the arcs have been lifted and a non-dominated l-t path in which no arcs have been lifted, since none of the nodes reachable from l have been evaluated. Therefore, we can combine the source labels at i_k with the sink labels at l to find the optimal path. The algorithm continues in this fashion until all nodes are evaluated in topological order.

Initializing the source labels on lines 1-2 requires $\mathcal{O}(|V||E|)$. Computing the sink labels on line 3 is possible in $\mathcal{O}(|A||E|)$ using a typical node labeling algorithm. Combining the source and sink labels on line 6 can be done in $\mathcal{O}(|E|^2)$, while calculating the lifting coefficient on line 7 can be done in $\mathcal{O}(\log |E|)$. Both of these lines must be executed for each arc in $A \setminus E$ and require $\mathcal{O}(|A||E|^2)$ overall. Extending the labels at node i_k is done

on lines 9 or 13 for each arc in A in $\mathcal{O}(|A||E|)$ overall. Therefore, the total work required is $\mathcal{O}(|A||E|^2)$ which is less than the $\mathcal{O}(|A|^2|E|)$ required to solve separate lifting problems exactly for each variable.

Note that it is possible to generalize this algorithm and allow multiple passes for more flexibility in the lifting sequence. If we partition $A \setminus E$ into $F_1 \cup \cdots \cup F_p$, the work required to lift the arc sets F_i in any particular order is $\mathcal{O}(|A||E|^2 + p|A||E|)$ as long as we lift the arcs within each set in topological order since we only need to combine the source and sink labels once for each arc in $A \setminus E$ over all passes. And if p is fixed, we have not changed the complexity of the algorithm. For example, we might let F_1 be the set of arcs in $A \setminus E$ with positive LP values, and F_2 be the set of arcs in $A \setminus E$ with LP values equal to zero.

4.5 Strength of the Node Precedence Inequalities

In Chapter 3, we mentioned that the clique inequality (3.19) and the lifted odd hole inequality (3.21) define facets of $\text{conv}(X^{NP})$. We have also shown that the lifted path subset cover inequality (4.2) defines a facet of $\text{conv}(X^{PS})$. However, we have not given any facetial results about the precedence inequalities introduced in Chapter 3 or their strengthened versions, none of which are valid for either of the two full-dimensional relaxations of X^{WCP}. We delayed these results because the assumption that G is acyclic simplifies the analysis. In this section, we give the dimension of $\text{conv}(X^{WCP})$ and show that the strengthened node precedence inequality (3.17) defines a facet of a projection of X^{WCP} onto a subspace of $\mathbb{R}^A$.

Proposition 4.5.1. $\dim(\text{conv}(X^{WCP})) = |A| - |V| + 1$.

Proof. Clearly $\dim(\text{conv}(X^{WCP})) \leq |A| - |V| + 1$, since $X^{WCP} \subseteq \mathbb{R}^A$ and the flow balance equations (3.1) have rank $|V| - 1$. To prove that $\dim(\text{conv}(X^{WCP})) = |A| - |V| + 1$, it suffices to show that there exists $|A| - |V| + 2$ affinely independent points in X^{WCP}.

Let T_s be a shortest path tree from s to all $i \in V$ with arc lengths given by the weight vector w, and let $\hat{Q}_{si}$ be the unique s-i path in T_s. Let Q_{jt} be a shortest j-t path in G with value σ_{jt} for all $j \in V \setminus \{s\}$. For each $(i,j) \in \widetilde{A} = A \setminus A(T_s)$, we let $P_{ij} = \hat{Q}_{si} \cup (i,j) \cup Q_{jt}$ with the corresponding incidence vector $p^{ij} \in \mathbb{B}^{A(T_s)} \times \mathbb{B}^{\widetilde{A}}$. We also define $(\hat{q}, 0) \in \mathbb{B}^{A(T_s)} \times \mathbb{B}^{\widetilde{A}}$ to

be the incidence vector for $\hat{Q}_{st}$. It follows from (A2) that all of these paths must be weight feasible. Therefore, $(\hat{q}, 0) \in X^{WCP}$ and $p^{ij} \in X^{WCP}$ for all $(i,j) \in \widetilde{A}$.

Since G is acyclic, we assume a topological ordering $i_1, i_2, \ldots, i_{|V|}$ on the nodes in V, and it follows from (A2) that $s = i_1$, $t = i_{|V|}$ and $\delta^+(t) = \emptyset$. By construction, no matter how we arrive at Q_{jt} for some $(i_k, j) \in \widetilde{A}$, $P_{i_k,j}$ will never contain an arc in $\widetilde{A} \cap \delta^+(i_l)$ for any $l \leq k$ other than (i_k, j) itself. We form the matrix M with a row for each arc in A. The rows are partitioned into the sets $\widetilde{A}$ and $A(T_s)$, and within $\widetilde{A}$, they are arranged in topological order. In the columns of M, we add the vectors $p^{ij} - (\hat{q}, 0)$ for each $(i,j) \in \widetilde{A}$ in topological order. Then,

$$M = \begin{pmatrix} I & & & & \widetilde{A} \cap \delta^+(s) \\ M_{2,1} & I & & & \widetilde{A} \cap \delta^+(i_2) \\ \vdots & \vdots & & & \vdots \\ M_{|V|-1,1} & M_{|V|-1,2} & \ldots & I & \widetilde{A} \cap \delta^+(i_{|V|-1}) \\ M_{T_s,1} & M_{T_s,2} & \ldots & M_{T_s,|V|-1} & A(T_s). \end{pmatrix}$$

The matrix M is lower triangular, and the columns are linearly independent. Therefore, $(\hat{q}, 0)$ along with the set of vectors p^{ij} for all $(ij) \in \widetilde{A}$ are affinely independent, and there are $|A| - |V| + 2$ such vectors since $|\widetilde{A}| = |A| - |A(T)|$, and (A2) assures us that G is s-t connected and $|A(T)| = |V| - 1$. □

Note that this result may not be true in the general case (cyclic G) as illustrated by the following example.

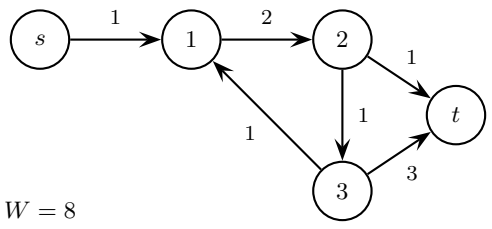

Figure 9: Constrained path example 3

Example 4.2. Consider the weight constrained path set in Figure 9. It is clear that $(3, 1)$ will never appear in a feasible s-t path even though $\sigma_{s3} + w_{31} + \sigma_{1t} = 4 + 1 + 3 = 8$. Therefore, $x_{31} = 0$ holds for all $x \in X^{WCP}$, and the dimension is reduced by one. ■

Given a set of arcs $F \subset A$, we can project X^{WCP} onto the subspace of $\mathbb{R}^A$ defined by $H = \{(x, y) \in \mathbb{R}^F \times \mathbb{R}^{A \setminus F} : y = 0\}$.

Definition 4.6. If $F \subset A$, the *projection of* X^{WCP} *on* F is the set

$$\text{proj}_F(X^{WCP}) = \left\{ x \in \mathbb{B}^F : (x, y) \in X^{WCP} \text{ for some } y \in \mathbb{B}^{A \setminus F} \right\}.$$

Example 4.3. Let $j \in V \setminus \{s, t\}$ and $F = \delta^-(j) \cup \delta^+(j)$. If $A[(j)] \subset A$, it is easy to show that we can formulate the projection of X^{WCP} on F as the $x \in \mathbb{B}^F$ satisfying

$$x(\delta^+(j)) \leq 1 \tag{4.3}$$

$$x(\delta^+(j)) - x(\delta^-(j)) = 0 \tag{4.4}$$

$$\sum_{(i,j) \in \delta^-(j)} (\sigma_{si} + w_{ij}) x_{ij} + \sum_{(j,k) \in \delta^+(j)} (w_{jk} + \sigma_{kt}) x_{jk} \leq W. \tag{4.5}$$

If $A[(j)] = A$, we can replace inequality (4.3) with $x(\delta^+(j)) = 1$. ■

If we can show that a particular inequality defines a facet of $\text{conv}(\text{proj}_F(X^{WCP}))$, we can conclude that this inequality defines a face of $\text{conv}(X^{WCP})$ of dimension at least $\dim(\text{conv}(\text{proj}_F(X^{WCP}))) - 1$. We now present this analysis for the simplest of the precedence inequalities introduced in Chapter 3. The strengthened node precedence inequality (3.17) is a special case of the strengthened version of the subpath precedence inequality (3.12) with $p = 3$, and when comparable is at least as strong as the strengthened version of the s-t cut precedence inequality (3.15) with $V^+(j) \subseteq \bar{S}$ or the clique inequality (3.19) with $C = \nabla^-_{ij} \cup (\delta^+(j) \setminus \phi^+_{ij})$.

Proposition 4.5.2. *Let* $j \in V \setminus \{s, t\}$ *and* $F = \delta^-(j) \cup \delta^+(j)$. *Then,*

$$\dim(\text{conv}(\text{proj}_F(X^{WCP}))) = \begin{cases} |F| - 1, & A[(j)] \subset A \\ |F| - 2, & A[(j)] = A. \end{cases}$$

Proof. Let $X_F = \text{proj}_F(X^{WCP})$. We show the existence of $|F| - 1$ linearly independent points in X_F.

Let Q_{si} be a shortest s-i path in G with value σ_{si} for all $i \in V^-(j)$, and Q_{kt} be a shortest k-t path in G with value σ_{kt} for all $k \in V^+(j)$. For each $i \in V^-(j)$ and $k \in V^+(j)$, let $P_{ik} = Q_{si} \cup (i, j, k) \cup Q_{kt}$ with the corresponding incidence vector $(e^{ik}, q^{ik}) \in \mathbb{B}^F \times \mathbb{B}^{A \setminus F}$.

It follows from (A2) that $V^-(j) \neq \emptyset$ and $V^+(j) \neq \emptyset$, and we define $\hat{i} = \arg\min\{\sigma_{si} + w_{ij} : i \in V^-(j)\}$ and $\hat{k} = \arg\min\{w_{jk} + \sigma_{kt} : k \in V^+(j)\}$. By definition and (A2), the paths $P_{\hat{i}k}$ for all $k \in V^+(j)$, and $P_{i\hat{k}}$ for all $i \neq \hat{i}$ such that $i \in V^-(j)$ are all weight feasible. Therefore, $e^{\hat{i}k} \in X_F$ for all $k \in V^+(j)$, and $e^{i\hat{k}} \in X_F$ for all $i \neq \hat{i}$ such that $i \in V^-(j)$.

We form the matrix M with a row for each arc in F. The rows are partitioned into four groups $(\hat{i}, j)$, $(j, \hat{k})$, $\delta^-(j) \setminus \{(\hat{i}, j)\}$ and $\delta^+(j) \setminus \{(j, \hat{k})\}$. The vectors $e^{\hat{i}\hat{k}}$, $e^{\hat{i}k}$ for each $k \in V^+(j) \setminus \{\hat{k}\}$, and $e^{i\hat{k}}$ for each $i \in V^-(j) \setminus \{\hat{i}\}$ can be arranged in the columns of M to get

$$M = \begin{pmatrix} 1 & & 1 \cdots 1 \\ 1 & 1 \cdots 1 & \\ & I & \\ & & I \end{pmatrix} \begin{matrix} (\hat{i}, j) \\ (j, \hat{k}) \\ \delta^-(j) \setminus \{(\hat{i}, j)\} \\ \delta^+(j) \setminus \{(j, \hat{k})\}. \end{matrix}$$

The matrix M is upper triangular, and the columns are linearly independent. Therefore, we have a set of $|F| - 1$ linearly independent vectors in X_F.

Clearly $\dim(\text{conv}(X_F)) \leq |F| - 1$, since $X_F \subseteq \mathbb{R}^F$ and the flow balance equation for node j has rank 1. We now consider two cases: (a) $A[(j)] \subset A$, and (b) $A[(j)] = A$.

Case (a): Let $(k, l) \in A \setminus A[(j)]$. Then there exists some shortest s-t path containing (k, l) with a corresponding incidence vector $(0, p) \in \mathbb{B}^F \times \mathbb{B}^{A \setminus F}$. It follows from (A2) this path is weight feasible. Therefore, $0 \in X_F$ and along with the $|F| - 1$ vectors defined above we have $|F|$ affinely independent vectors in X_F which implies the result $\dim(\text{conv}(X_F)) = |F| - 1$.

Case (b): Any s-t path in G must contain j, hence any solution in X^{WCP} will satisfy the equation $x(\delta^+(j)) = 1$. It follows from (A2) that $\delta^-(j) \neq \emptyset$, hence $x(\delta^+(j)) = 1$ and the flow balance equation for node j are linearly independent and have rank 2. Therefore, $\dim(\text{conv}(X_F)) \leq |F| - 2$, and using the $|F| - 1$ linearly independent vectors defined above

we get $\dim(\text{conv}(X_F)) = |F| - 2$. □

Proposition 4.5.3. *Let $(i,j) \in A$ such that $j \neq s,t$ and $F = \delta^-(j) \cup \delta^+(j)$. If*

i. $\widetilde{\phi}^+_{ij} \neq \emptyset$, *and*

ii. $\phi^+_{lj} \supset \phi^+_{ij}$ *for all* $(l,j) \in \widetilde{\nabla}^-_{ij}$

where $\widetilde{\phi}^+_{ij} = \delta^+(j) \setminus \phi^+_{ij}$ *and* $\widetilde{\nabla}^-_{ij} = \delta^-(j) \setminus \nabla^-_{ij}$, *then the strengthened node precedence inequality (3.17) defines a facet of* $\text{conv}(\text{proj}_F(X^{WCP}))$.

Proof. Let $X_F = \text{proj}_F(X^{WCP})$. We show the existence of $|F| - 2$ linearly independent points in X_F satisfying (3.17) at equality.

Let Q_{sl} be a shortest s-l path in G with value σ_{sl} for all $l \in V^-(j)$, and Q_{kt} be a shortest k-t path in G with value σ_{kt} for all $k \in V^+(j)$. For each $l \in V^-(j)$ and $k \in V^+(j)$, let $P_{lk} = Q_{sl} \cup (l,j,k) \cup Q_{kt}$ with the corresponding incidence vector $(e^{lk}, q^{lk}) \in \mathbb{B}^F \times \mathbb{B}^{A \setminus F}$.

By (A2), $\phi^+_{ij} \neq \emptyset$ and we define $\hat{k} = \arg\min\{w_{jk} + \sigma_{kt} : (j,k) \in \phi^+_{ij}\}$ and $\tilde{k} = \arg\min\{w_{jk} + \sigma_{kt} : (j,k) \in \widetilde{\phi}^+_{ij}\}$. It follows from condition i. along with (A2) that $\widetilde{\nabla}^-_{ij} \neq \emptyset$ and we define $\tilde{l} = \arg\min\{\sigma_{sl} + w_{lj} : (l,j) \in \widetilde{\nabla}^-_{ij}\}$. By the definition of $\widetilde{\nabla}^-_{ij}$ and ϕ^+_{ij}, $\tilde{l} = \arg\min\{\sigma_{sl} + w_{lj} : l \in V^-(j)\}$ and $\hat{k} = \arg\min\{w_{jk} + \sigma_{kt} : k \in V^+(j)\}$.

Hence by (A2), the paths $P_{\tilde{l}k}$ for all k such that $(j,k) \in \widetilde{\phi}^+_{ij}$ and $P_{l\hat{k}}$ for all $l \neq i$ such that $(l,j) \in \nabla^-_{ij}$ are all weight feasible. Condition ii. ensures that the paths $P_{l\tilde{k}}$ for all $l \neq \tilde{l}$ such that $(l,j) \in \widetilde{\nabla}^-_{ij}$ are weight feasible. Therefore, the vectors e^{ik} for all k such that $(j,k) \in \phi^+_{ij}$, $e^{\tilde{l}k}$ for all k such that $(j,k) \in \widetilde{\phi}^+_{ij}$, $e^{l\hat{k}}$ for all $l \neq i$ such that $(l,j) \in \nabla^-_{ij}$, and $e^{l\tilde{k}}$ for all $l \neq \tilde{l}$ such that $(l,j) \in \widetilde{\nabla}^-_{ij}$ are all in X_F and satisfy (3.17) at equality.

We form the matrix M with a row for each arc in F. The rows are partitioned into eight groups (i,j), $(\tilde{l},j)$, $(j,\hat{k})$, $(j,\tilde{k})$, $\nabla^-_{ij} \setminus \{(i,j)\}$, $\widetilde{\nabla}^-_{ij} \setminus \{(\tilde{l},j)\}$, $\phi^+_{ij} \setminus \{(j,\hat{k})\}$ and $\widetilde{\phi}^+_{ij} \setminus \{(j,\tilde{k})\}$. The vectors defined above satisfying (3.17) at equality can be arranged in the columns of

M to get

$$M = \begin{pmatrix} 1 & & & & & 1\cdots 1 & & \\ & 1 & & & & & 1\cdots 1 & \\ 1 & & 1\cdots 1 & & & & & \\ & 1 & & 1\cdots 1 & & & & \\ & & I & & & & & \\ & & & & I & & & \\ & & & & & I & & \\ & & & & & & I & \end{pmatrix} \begin{matrix} (i,j) \\ (\tilde{l},j) \\ (j,\hat{k}) \\ (j,\tilde{k}) \\ \nabla_{ij}^- \setminus \{(i,j)\} \\ \widetilde{\nabla}_{ij}^- \setminus \{(\tilde{l},j)\} \\ \phi_{ij}^+ \setminus \{(j,\hat{k})\} \\ \widetilde{\phi}_{ij}^+ \setminus \{(j,\tilde{k})\}. \end{matrix}$$

The matrix M is upper triangular, and the columns are linearly independent. Therefore, we have a set of $|F|-2$ linearly independent vectors in X_F satisfying (3.17) at equality.

Clearly, inequality (3.17) is valid for X_F, and the vector $e^{\tilde{l}\hat{k}}$ is in X_F and does not satisfy (3.17) at equality, hence it is proper. If $A[(j)] = A$, we are done since $\dim(\text{conv}(X_F)) = |F|-2$. If $A[(j)] \subset A$, we know $0 \in X_F$ and satisfies (3.17) at equality. Therefore, 0 along with the $|F|-2$ vectors defined above gives $|F|-1$ affinely independent vectors in X_F which implies the result since $\dim(\text{conv}(X_F)) = |F|-1$ if $A[(j)] \subset A$. □

Note that each inequality (3.17) is associated with some $(i,j) \in A$. The previous result gives us an indication as to which arcs (or inequalities) are actually necessary. The set ∇_{ij}^- includes all arcs into j which have shortest s-j paths through them that are at least as long as the shortest s-j path through (i,j), and we know that ϕ_{ij}^+ is valid for all of these arcs. But there may be some arc $(l,j) \notin \nabla_{ij}^-$ with an improved shortest path for which $\phi_{lj}^+ = \phi_{ij}^+$, and in that case we could have based the inequality on (l,j) rather than (i,j) and increased the left hand side by one term. Condition ii. in Proposition 4.5.3 ensures that no such (l,j) exists, while condition i. guarantees that the inequality is not redundant.

Example 4.4. Consider the projection of X^{WCP} on $F = \delta^-(j) \cup \delta^+(j)$ in Figure 10. Inequality (3.17) for $(3,j)$ is given by $x_{3j} + x_{4j} \leq x_{j5} + x_{j6}$. But $(2,j)$ has an improved shortest path and $\phi_{2j}^+ = \phi_{3j}^+ = \{(j,5),(j,6)\}$. Hence, the inequality is not facet defining because condition ii. does not hold. However, the inequality $x_{2j} + x_{3j} + x_{4j} \leq x_{j5} + x_{j6}$ for $(2,j)$ does define a facet of $\text{conv}(\text{proj}_F(X^{WCP}))$ since $\phi_{2j}^+ \subset \phi_{1j}^+ = \delta^+(j)$. ■

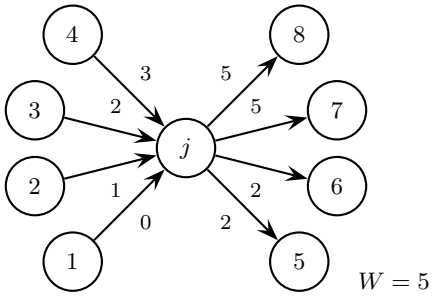

Figure 10: Example of $\text{proj}_F(X^{WCP})$ where $F = \delta^-(j) \cup \delta^+(j)$

Remark 14. *If we add the flow balance equation (3.1) for j to inequality (3.17), we get the inequality $x(\widetilde{\phi}^+_{ij}) \leq x(\widetilde{\nabla}^-_{ij})$. If conditions i. and ii. hold for Proposition 4.5.3, and we define $\tilde{k} = \arg\min\{w_{jk} + \sigma_{kt} : (j,k) \in \widetilde{\phi}^+_{ij}\}$, then this inequality is equivalent to the strengthened reverse node precedence inequality (3.18) for $(j, \tilde{k})$, because in that case $\widetilde{\phi}^+_{ij} = \nabla^+_{j\tilde{k}}$ and $\widetilde{\nabla}^-_{ij} = \phi^-_{j\tilde{k}}$.*

While a face of $\text{conv}(X^{WCP})$ of dimension at least $|F|-1$ where $F = \delta^-(j) \cup \delta^+(j)$ may not seem noteworthy, we point out that each affinely independent vector in $\text{proj}_F(X^{WCP})$ which satisfies (3.17) at equality may represent many affinely independent paths in X^{WCP}. Therefore, the actual dimension may be significantly larger. For example, $0 \in \text{proj}_F(X^{WCP})$ represents all of the affinely independent paths which contain arcs in $A \setminus A[(j)]$.

CHAPTER 5

BRANCH-AND-CUT FOR THE RESOURCE CONSTRAINED SHORTEST PATH PROBLEM

5.1 Introduction

Few researchers have used branch-and-cut as a practical method for solving the resource constrained shortest path problem (RCSPP). The work by Spoorendonk et al. [106] is the only case we are aware of which uses cutting planes, other than some variant of the subtour elimination inequalities, to solve the problem. In this chapter, we discuss several of the fundamental components involved in the design of a branch-and-cut algorithm for the RCSPP based on the inequality classes introduced in Chapters 3 and 4.

Given a directed graph $G = (V, A)$ with distinct nodes $s, t \in V$, the initial constraint set is given by

$$x(\delta^+(i)) - x(\delta^-(i)) = \begin{cases} 1, & i = s \\ -1, & i = t \\ 0, & \forall\ i \in V \setminus \{s, t\} \end{cases} \tag{5.1}$$

$$\sum_{(i,j)\in A} w_{ij}^r x_{ij} \leq W_r,\ \forall\ r = 1, \ldots, R \tag{5.2}$$

$$x_{ij} \in \{0, 1\},\ \forall\ (i, j) \in A, \tag{5.3}$$

where $w^r \in \mathbb{R}^A$ and $W_r \in \mathbb{R}$ for resources $r = 1, \ldots, R$. We solve the problem

$$\min \left\{ cx : x \in X^{RCP} \right\}$$

with arc costs $c \in \mathbb{R}^A$ and X^{RCP} defined as follows

$$X^{RCP} = \left\{ x \in \mathbb{B}^A : \text{subject to (5.1) and (5.2)} \right\}.$$

All other valid inequalities, including those used to forbid negative cost cycles, will be implicitly included by separating them only when violated.

We are interested in a minimum cost (elementary) s-t path in G. Hence, if $i \in V$ cannot be reached from s, or t cannot be reached from i, then we can exclude i from the formulation along with all of its incident arcs. Moreover, it is clear that any arc $(i, s) \in \delta^-(s)$ or $(t, j) \in \delta^+(t)$ will never appear in such a path because they would induce a cycle. Thus, they can also be excluded from the formulation. These two trivial assumptions about the structure of G are formally stated below.

Assumption (A1). *G is s-t connected, i.e., $\rho^+(s) = \rho^-(t) = V$.*

Assumption (A2). $\delta^-(s) = \delta^+(t) = \emptyset$.

The inequalities introduced in Chapters 3 and 4 will make heavy use of the shortest paths in G with respect to a particular resource vector w^r. Therefore, we also make the following assumption to ensure that these paths can be found in polynomial time, since no assumption is made about the values in the resource vectors.

Assumption (A3). *G does not contain a negative weight cycle for any resource $r = 1, \ldots, R$.*

In the remainder of this chapter, we describe the essential elements of a branch-and-cut algorithm. We begin with preprocessing and probing techniques. Two existing schemes from the literature are presented along with three novel techniques for the case when G is acyclic. Next, we summarize all of the cutting planes in the algorithm including stronger versions of many of the inequalities introduced in Chapters 3 and 4. Then, we strengthen the typical variable branching scheme with preprocessing and consider alternative branching strategies which lead to balanced branch-and-bound trees. Finally, we discuss potential primal heuristic techniques.

5.2 Preprocessing and Probing

Preprocessing and probing techniques for binary integer programs (BIPs) use reformulation to strengthen the linear programming (LP) relaxation and tighten the integrality gap (see [65]). They do so by (among other things) reducing the size of coefficients in the constraint matrix and fixing variables to one of their bounds. These techniques also try to minimize

the computational effort required to solve the problem by reducing the size of the formulation. This can be accomplished by eliminating redundant constraints and substituting for variables that have been fixed to one of their bounds. For an overview of approaches for the more general case of *mixed integer programs* (MIPs), which include both integer and continuous variables, we refer the reader to the paper by Savelsbergh [102]. In this section, we discuss preprocessing and probing techniques which have been specialized for the RCSPP.

5.2.1 Previous Work

Several preprocessing schemes for the RCSPP have been proposed in the literature. Aneja et al. [2] reduce the size of the underlying graph by eliminating nodes and arcs that cannot appear in a feasible s-t path. These nodes and arcs are identified using bounds obtained from shortest weight paths in G with respect to each resource vector. Beasley and Christofides [11] extend this approach by considering bounds on the cost of the path. The cost bounds, which are obtained through Lagrangean relaxation, are used along with minimum reduced cost paths in G to eliminate additional nodes and arcs. A similar method was also explored by Mehlhorn and Ziegelmann [92]. Dumitrescu and Boland [35] simplified the approach in [11] by using the original cost vector to calculate the bounds rather than solving the Lagrangean relaxation. However, their procedure leads to improved performance because the cost bounds are updated each time the graph is reduced. Our algorithm will utilize both of the schemes in [2] and [35], and we now present them in greater detail.

For any vector $v \in \mathbb{R}^A$, let σ_{ij}^v be the value of a shortest i-j path Q_{ij}^v in G with arc lengths given by v. Then, it is clear that any node i, such that $\sigma_{si}^{w^r} + \sigma_{it}^{w^r} > W_r$ for some resource r, will never appear in a feasible s-t path in G. Similarly, flow through any arc (i,j), such that $\sigma_{si}^{w^r} + w_{ij}^r + \sigma_{jt}^{w^r} > W_r$ for some r, will always be infeasible. The scheme of Aneja et al. [2] examines each resource constraint individually and removes any nodes and arcs satisfying these conditions. A single pass of their iterative method is presented in Algorithm 9.

One call to the function AAN_Preprocessing has complexity $\mathcal{O}(\mathcal{SP}(V,A)R)$ since we

Algorithm 9 RCSPP preprocessing by Aneja et al. [2] for $G = (V, A)$

AAN_PREPROCESSING(G):
find $\sigma_{si}^{w^r}$ for all $i \in V$ and for all $r = 1, \ldots, R$
if $\sigma_{st}^{w^r} > W_r$ for some r **then**
  stop G has no feasible s-t path
end if
find $\sigma_{it}^{w^r}$ for all $i \in V$ and for all $r = 1, \ldots, R$
for all $i \in V \setminus \{s, t\}$ **do**
  if $\sigma_{si}^{w^r} + \sigma_{it}^{w^r} > W_r$ for some r **then**
    $V \leftarrow V \setminus \{i\}$
    $A \leftarrow A \setminus (\delta^-(i) \cup \delta^+(i))$
  end if
end for
for all $(i, j) \in A$ **do**
  if $\sigma_{si}^{w^r} + w_{ij}^r + \sigma_{jt}^{w^r} > W_r$ for some r **then**
    $A \leftarrow A \setminus \{(i, j)\}$
  end if
end for

must solve two shortest path problems for each resource r to calculate $\sigma_{si}^{w^r}$ and $\sigma_{it}^{w^r}$ for all $i \in V$. This function can be called repeatedly until the problem is shown to be infeasible or no other reductions to G are possible.

Remark 15. *$\sigma_{si}^{w^r} = \infty$ for all $i \notin \rho^+(s)$ and $\sigma_{jt}^{w^r} = \infty$ for all $j \notin \rho^-(t)$. Hence, G satisfies (A1) after each call to* AAN_PREPROCESSING.

This scheme is an example of *probing*, i.e., the investigation of the logical consequences when variables or groups of variables are fixed to zero or one. The classic BIP probing technique scans each resource r and removes arc (i, j) if $\min\left\{w^r x : x \in \mathbb{B}^A,\ x_{ij} = 1\right\} > W_r$. By approximately solving the shortest weight path problem

$$z_{ij}^r = \min\left\{w^r x : x \in X^P,\ x_{ij} = 1\right\},$$

where $X^P = \left\{x \in \mathbb{B}^A : \text{subject to (5.1)}\right\}$, the approach in [2] utilizes the fact that x is the incidence vector of an s-t path in G and tests for the stronger condition $z_{ij}^r > W_r$. We say approximately because the concatenation of the shortest weight paths $Q_{si}^{w^r} \cup (i, j) \cup Q_{jt}^{w^r}$ is a walk (not necessarily a path) containing (i, j). Therefore, $\sigma_{si}^{w^r} + w_{ij}^r + \sigma_{jt}^{w^r}$ is a valid lower bound (exact if G is acyclic) on z_{ij}^r, and (i, j) can be excluded if $z_{ij}^r \geq \sigma_{si}^{w^r} + w_{ij}^r + \sigma_{jt}^{w^r} > W_r$.

Note that testing if $\sigma_{si}^{w^r} + \sigma_{it}^{w^r} > W_r$ for some $i \in V$ is equivalent to simultaneously probing on the arcs in $\delta^-(i)$ and $\delta^+(i)$.

Dumitrescu and Boland [35] extend this notion by probing with the objective function. The authors observe that if there exists an incumbent s-t path P_I which is weight feasible for all resources and has cost $U = c(P_I)$, then any node i such that $\sigma_{si}^c + \sigma_{it}^c \geq U$ will only appear in s-t paths in G which are no better in value than P_I. Similarly, flow through any arc (i, j) such that $\sigma_{si}^c + c_{ij} + \sigma_{jt}^c \geq U$ will never lead to an improved solution. Therefore, any nodes and arcs satisfying these conditions can be excluded without increasing the optimal value of the RCSPP. In order to calculate shortest cost paths in polynomial time, however, we must make the additional assumption that no negative cost cycles exist in G. A single pass of their iterative method is presented in Algorithm 10.

Unless a feasible solution has been provided by some external heuristic, the function DB_Preprocessing is initially called with no incumbent P_I. To update the incumbent, the authors take advantage of the fact that a shortest weight s-t path $Q_{st}^{w^r}$ is generated for each resource r. Each of these paths is tested for feasibility with respect to all other resources, and if any of them have an improving cost value, the incumbent is replaced. The incumbent can also be replaced with any shortest cost s-t walk $Q_{si}^c \cup (i, j) \cup Q_{jt}^c$ for some $(i, j) \in A$ if its value is less than U and it is weight feasible for all resources. This walk is a valid upper bound (exact if G is acyclic) on the optimal value of the RCSPP if there are no negative cost cycles allowed in G, and it can be converted to a feasible path by removing any cycles since (A1) guarantees no negative weight cycles in G.

Lines 4-8 and 12-16 in DB_Preprocessing detect both infeasibility and optimality. Clearly, if the shortest cost s-t path is weight feasible for all resources, then it is optimal. Otherwise, the existence of an incumbent P_I guarantees the problem is feasible. If the shortest cost s-t path is at least as long as the incumbent, or there does not exist a weight feasible s-t path in G for some resource r, then the incumbent must be an optimal solution.

If we assume that no negative cost cycles exist in G, all shortest path values can be found in $\mathcal{O}(\mathcal{SP}(V, A)R)$ because we only need to solve two additional shortest path problems for the costs. Given any shortest s-i path Q_{si}^v or shortest i-t path Q_{it}^v for some

Algorithm 10 RCSPP preprocessing by Dumitrescu and Boland [35] for $G = (V, A)$

DB_PREPROCESSING(G, P_I):
$U \leftarrow c(P_I)$ (if $P_I = \emptyset$, then $c(P_I) = \infty$)
find Q^c_{si} (if it exists) and σ^c_{si} for all $i \in V$
if Q^c_{st} does not exist or $\sigma^c_{st} \geq U$ **then**
  stop P_I is the optimal solution (infeasible if $P_I = \emptyset$)
else if $w^r(Q^c_{st}) \leq W_r$ for all $r = 1, \ldots, R$ **then**
  stop Q^c_{st} is the optimal solution
end if
find Q^c_{it} and σ^c_{it} for all $i \in V$
for all $r = 1, \ldots, R$ **do**
  find $Q^{w^r}_{si}$ and $\sigma^{w^r}_{si}$ for all $i \in V$
  if $\sigma^{w^r}_{st} > W_r$ **then**
    stop P_I is the optimal solution (infeasible if $P_I = \emptyset$)
  else if $c(Q^{w^r}_{st}) < U$ and $w^q(Q^{w^r}_{st}) \leq W_q$ for all $q = 1, \ldots, R$ **then**
    $P_I \leftarrow Q^{w^r}_{st}$, $U \leftarrow c(Q^{w^r}_{st})$
  end if
  find $Q^{w^r}_{it}$ and $\sigma^{w^r}_{it}$ for all $i \in V$
end for
for all $i \in V \setminus \{s, t\}$ **do**
  if $\sigma^c_{si} + \sigma^c_{it} \geq U$ or $\sigma^{w^r}_{si} + \sigma^{w^r}_{it} > W_r$ for some r **then**
    $V \leftarrow V \setminus \{i\}$
    $A \leftarrow A \setminus (\delta^-(i) \cup \delta^+(i))$
  end if
end for
for all $(i, j) \in A$ **do**
  if $\sigma^c_{si} + c_{ij} + \sigma^c_{jt} \geq U$ or $\sigma^{w^r}_{si} + w^r_{ij} + \sigma^{w^r}_{jt} > W_r$ for some r **then**
    $A \leftarrow A \setminus \{(i, j)\}$
  else if $w^r(Q^c_{si}) + w^r_{ij} + w^r(Q^c_{jt}) \leq W_r$ for all $r = 1, \ldots, R$ **then**
    $P_I \leftarrow Q^c_{st} \cup (i, j) \cup Q^c_{jt}$, $U \leftarrow \sigma^c_{si} + c_{ij} + \sigma^c_{jt}$
  end if
end for

vector $v \in \mathbb{R}^A$, we can calculate the values $w^r(Q^v_{si})$ and $w^r(Q^v_{it})$ for all $i \in V$ and for all $r = 1, \ldots, R$ in $\mathcal{O}(|V|R)$. Therefore, the total work required by one call to the function DB_Preprocessing has complexity $\mathcal{O}(\mathcal{SP}(V, A)R + |V|R^2)$, and it can be called repeatedly until an s-t path is proven optimal, the problem is shown to be infeasible, or no other reductions to G are possible.

5.2.2 Acyclic Preprocessing Enhancements

Several enhancements to the usual RCSPP preprocessing schemes are possible when G is acyclic. We mention three possibilities which, to the best of our knowledge, have not been presented in the literature for the RCSPP.

The first enhancement concerns the identification of redundant constraints which are always satisfied by feasible solutions. The classic BIP preprocessing technique removes any resource constraint r if $\max\left\{w^r x : x \in \mathbb{B}^A\right\} = \sum_{(i,j)\in A}[w^r_{ij}]^+ \leq W_r$, where $[x]^+ = \max\{0, x\}$. This is an approximation of the actual condition which implies that r is redundant, i.e., $\max\{w^r x : x \in X^P,\ \sum_{(i,j)\in A} w^q_{ij} x_{ij} \leq W_q,\ \forall\ q \in \{1, \ldots, R\} \setminus \{r\}\} \leq W_r$. An approximation is used because the actual condition requires us to solve an RCSPP that is equally as hard as the original problem. But since $x \in X^P$, we can solve the shortest weight path problem

$$z_r = \max\left\{w^r x : x \in X^P\right\} = -\min\left\{-w^r x : x \in X^P\right\} \tag{5.4}$$

and use the stronger approximation $z_r \leq W_r$. Note that the complexity of this test is equivalent to the classic BIP technique because we can calculate shortest paths with negative arc lengths in $\mathcal{O}(|A|)$ for acyclic graphs.

The second enhancement concerns matrix coefficient reduction. For any $(k, l) \in A$ and resource r such that $w^r_{kl} > 0$, the classic BIP preprocessing technique reduces w^r_{kl} and W_r by $\Delta^r_{kl} = W_r - \max\left\{w^r x : x \in \mathbb{B}^A,\ x_{kl} = 0\right\} = W_r - \sum_{(i,j)\in A\setminus\{(k,l)\}}[w^r_{ij}]^+$ if $\Delta^r_{kl} > 0$. However, any $\Delta^r_{kl} \leq W_r - \max\{w^r x : x \in X^P,\ \sum_{(i,j)\in A} w^q_{ij} x_{ij} \leq W_q,\ \forall\ q \in \{1, \ldots, R\} \setminus \{r\},\ x_{kl} = 0\}$ is also valid. Hence, we can solve the shortest weight path problem

$$z^r_{kl} = \max\left\{w^r x : x \in X^P,\ x_{kl} = 0\right\} = -\min\left\{-w^r x : x \in X^P,\ x_{kl} = 0\right\} \tag{5.5}$$

for a larger $\Delta_{kl}^r = W_r - z_{kl}^r$. It is clear that if resource constraint r is not redundant, i.e., $z^r = \max\left\{w^r x : x \in X^P\right\} > W_r$, then there exists an s-t path P such that $z_{kl}^r = z^r > W_r$ and $\Delta_{kl}^r = W_r - z^r < 0$ for all $(k,l) \in A \setminus A(P)$. Therefore, only the arcs in $A(P)$ are candidates for coefficient reduction, and testing them requires $\mathcal{O}(|V||A|)$ for acyclic graphs.

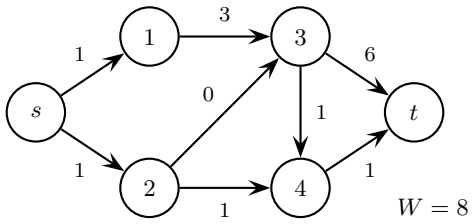

Figure 11: Constrained path example 4

Example 5.1. Consider the weight constrained path set in Figure 11 which has no opportunity for coefficient reduction if we use the classic BIP technique. The 'longest' weight path that does not contain $(3,t)$ has value 6. Therefore, $\Delta_{3t} = W - z_{3t} = 8 - 6 = 2$, and we can reduce to $w_{3t} = 4$ and $W = 6$. This procedure can be repeated for $(s,1)$ to get $\Delta_{s1} = 1$, and we can reduce further to $w_{s1} = 0$ and $W = 5$. ■

In Chapter 4, we defined the subgraph $G[Q]$ induced by all s-t paths in G containing a given s-t subpath Q. Our final enhancement uses this subgraph to extend the RCSPP preprocessing scheme of Aneja et al. discussed above which removes infeasible nodes and arcs. By definition, the function AAN_Preprocessing will never remove any node i such that $\sigma_{si}^{w^r} + \sigma_{it}^{w^r} \leq W_r$ for all $r = 1, \ldots, R$. However, the existence of a weight feasible s-t path containing i for each resource constraint does not guarantee the existence of an s-t path containing i which is feasible for all resource constraints. If we repeatedly call AAN_Preprocessing on $G[(i)]$ (rather than G) and conclude $G[(i)]$ has no feasible s-t path, then we have proven that all s-t paths in G containing i are infeasible with respect to at least one resource constraint. Therefore, i can be removed from G along with all of its incident arcs. Similarly, we can repeatedly call AAN_Preprocessing on $G[(i,j)]$ to identify an infeasible $(i,j) \in A$.

Example 5.2. Consider the resource constrained path set in Figure 12. It is easy to show

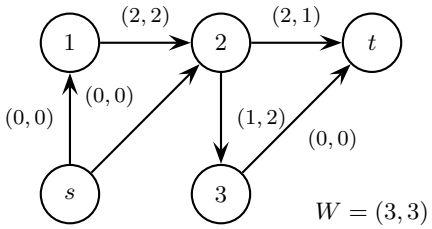

Figure 12: Constrained path example 5

that the scheme of Aneja et al. will not eliminate node 1 from the original graph G since $\sigma_{s1}^{w^r} + \sigma_{1t}^{w^r} = 3$ for $r = 1, 2$. Now consider the graph $G[(1)] = G \setminus \{(s, 2)\}$ and observe that the shortest weight path through $(2, t)$ for resource 1, and the shortest weight path through $(2, 3)$ for resource 2, both have a value of 4 in this graph. Hence, arcs $(2, 3)$ and $(2, t)$ will both be removed after a single call to AAN_Preprocessing on $G[(1)]$, and the resulting graph will be disconnected, i.e., no feasible s-t path exists in $G[(1)]$. Therefore, node 1 and its incident arcs $(s, 1)$ and $(1, 2)$ can be eliminated from G. ■

Remark 16. *Combining the scheme of Aneja et al. with node or arc fixing does nothing new if* $R = 1$.

This final preprocessing enhancement can be rather expensive. Each time a node i or an arc (i, j) is fixed in this scheme, we must first define $G[(i)]$ or $G[(i, j)]$ in $\mathcal{O}(|A|)$. Then, the function AAN_Preprocessing, which has complexity $\mathcal{O}(|A|R)$ for acyclic graphs, is called at least once. Therefore, it is probably wise to fix only a subset of the nodes and arcs in G. One possibility is to fix any node i for which $\sigma_{si}^c + \sigma_{it}^c$ is relatively small. Another option is to fix only those arcs that appear in the initial fractional solution to the LP relaxation.

Note that weakened versions of all three acyclic enhancements are applicable if G contains cycles. Rather than solving shortest path problems (5.4) and (5.5) with negative arc lengths for the first two enhancements, we can take the minimum of two trivial generalized upper bound (GUB) problems which can be solved in $\mathcal{O}(|A|)$ with a greedy algorithm, e.g.,

$$z_r = \min\left\{\max\left\{w^r x : x(\delta^-(i)) \leq 1,\ \forall\ i \in V\right\},\ \max\left\{w^r x : x(\delta^+(i)) \leq 1,\ \forall\ i \in V\right\}\right\}.$$

The bounds given by these trivial GUB problems are still stronger than the classic BIP

preprocessing approximations. The third enhancement can be handled with the cyclic approximation for $G[Q]$ discussed in Section 4.2.

5.3 Cutting Planes

After preprocessing and probing are complete, branch-and-cut algorithms continue to reformulate and strengthen the LP relaxation throughout the entire search. This is accomplished by generating cutting planes at any node in the branch-and-bound tree. Besides the generic cuts found in most commercial BIP solvers, we have a number of valid inequalities for the RCSPP at our disposal, and we discuss them in this section.

5.3.1 Cycle Elimination Inequalities

No assumptions have been made regarding the arc costs (they could be negative), and G is not assumed to be acyclic. Consequently, the presence of negative cost cycles might encourage a solution to the initial formulation, $\min\left\{cx : x \in X^{RCP}\right\}$, which is comprised of a non-elementary s-t path along with possibly many disjoint subgraphs of G. In this case, additional inequalities which forbid cycles must be added to the formulation.

The trivial GUB inequalities

$$x(\delta^+(i)) \leq 1,\ \forall\ i \in V \setminus \{t\} \tag{5.6}$$

and

$$x(\delta^-(i)) \leq 1,\ \forall\ i \in V \setminus \{s\} \tag{5.7}$$

are enough to ensure that the corresponding support graph of any integral solution is induced by an elementary s-t path along with possibly many subtours (disjoint cycles) in G. There are only $\mathcal{O}(|V|)$ such inequalities, and they can be added to the formulation a priori; otherwise, they are separated by inspection.

To forbid subtours, we can add the subtour elimination inequalities

$$x(\gamma(S)) \leq |S| - 1,\ \forall\ S \subset V,\ 2 \leq |S| \leq |V| - 2 \tag{5.8}$$

which have also been used for many other BIPs like the Traveling Salesman Problem (TSP). Of course, there are exponentially many of these inequalities, so the only practical approach

is to exclude them from the initial formulation and separate them as needed. By transforming to a maximum s-excess problem, Dumitrescu [34] showed that the separation problem for these inequalities can be solved in polynomial time.

For the TSP, the subtour elimination inequalities can be expressed in an (equivalent) alternate form. For all $\emptyset \neq S \neq V$, the inequality $x(\delta^-(S)) + x(\delta^+(S)) \geq 2$ is valid for the TSP because there must be flow through every node in G. The separation problem for these alternate inequalities can be solved exactly by solving $|V| - 1$ minimum cut problems, and very fast separation heuristics have also been developed to find many violated inequalities at once (see [97]). However, these inequalities are not valid for the RCSPP unless we condition on the flow through some node in S.

To arrive at such a conditioned inequality, let us consider the *generalized subtour elimination inequalities* which can be expressed as

$$x(\gamma(S)) \leq \sum_{k \in S \setminus \{i\}} x(\delta^+(k)),\ \forall\ i \in S,\ S \subseteq V \setminus \{t\}\,. \tag{5.9}$$

These inequalities are valid for the prize collecting TSP [112], which does not require flow through every node in G, and they are stronger than the original subtour elimination inequalities since $\sum_{k \in S \setminus \{i\}} x(\delta^+(k)) \leq |S| - 1$ for all $i \in S \subseteq V$. If we subtract $\sum_{k \in S} x(\delta^+(k))$ from both sides of the inequality and multiply by -1, we get the equivalent inequalities

$$x(\delta^+(S)) \geq x(\delta^+(i)),\ \forall\ i \in S,\ S \subseteq V \setminus \{t\} \tag{5.10}$$

which resemble the alternate form of the subtour elimination inequalities for the TSP. It is easy to show that inequalities (5.9) and (5.10) are valid for the RCSPP, and the polynomial separation routines and heuristics for the alternate TSP subtour elimination inequalities can be adapted to separate inequality (5.10).

5.3.2 Valid Inequalities for a Single Resource

In Chapters 3 and 4, we introduced several valid inequality classes for the weight constrained shortest path problem (WCSPP), i.e., the RCSPP with a single resource. We summarize these inequalities in Table 1.

Table 1: Valid inequality classes for the WCSPP

Inequality Class	*Facet Defining?*	*Separation Complexity*
node precedence[†] (3.8) and (3.9)	for a projection if strengthened	polynomial[‡]
subpath precedence[†] (3.12) and (3.13)	unknown	conjectured to be $\mathcal{NP}$-hard
s-t cut precedence[†] (3.15) and (3.16)	unknown	polynomial
clique[§] (3.19)	for conv(X^{NP})	$\mathcal{NP}$-hard
lifted odd hole[§] (3.21)	for conv(X^{NP})	$\mathcal{NP}$-hard
path subset cover (4.1)	for conv(X_E^{PS})	$\mathcal{NP}$-hard
lifted path subset cover (4.2)	for conv(X^{PS})	conjectured to be $\mathcal{NP}$-hard

[†]Can be strengthened using ∇_{ij}^- and ∇_{ij}^+ (see inequality (3.17) for example)

[‡]There are only $\mathcal{O}(|A|)$ such inequalities for each resource

[§]Derived from the conflict graph defined in Section 3.6.2

For the RCSPP, each resource r defines a set of weight constrained paths X_r^{WCP} such that $X_r^{WCP} \supseteq X^{RCP}$ for all $r = 1, \ldots, R$. As a result, any valid inequalities for these relaxations are also valid for X^{RCP}. In fact, the objective function can be used as an additional resource constraint. Feasible solutions are found throughout the branch-and-cut algorithm, and the best of these provides an upper bound U on the value of the optimal solution. Therefore, the optimality constraint

$$\sum_{(i,j)\in A} c_{ij} x_{ij} \leq U \tag{5.11}$$

can be used to derive any of the inequalities in Table 1. The resulting inequalities will not be valid for X^{RCP} in general, but they will only cut off feasible solutions which are no better than the one associated with U. When derived from constraint (5.11), all of the inequalities in Table 1 would make use of the shortest cost path values σ_{ij}^c for some $i, j \in V$. Therefore, we must make the additional assumption that no negative cost cycles exist in G so that they can be calculated in polynomial time $\mathcal{O}(\mathcal{SP}(V, A))$.

Remark 17. *The right hand side of (5.11) can be strengthened to* $U - 1$ *if* $c \in \mathbb{Z}^A$.

5.3.3 Valid Inequalities for Multiple Resources

To solve a general BIP, Crowder et al. [22] proposed adding valid inequalities for each of the polytopes defined by a single knapsack row in the constraint matrix. They reasoned that if the constraint matrix is sufficiently sparse, the intersection of the individual knapsack polytopes should give a reasonable approximation to the convex hull of feasible solutions for the BIP. They also gave empirical evidence to support this hypothesis. It seems natural to assume that the approach of finding valid inequalities for the combination of a single resource constraint with an underlying s-t path structure should perform similarly. Spoorendonk et al. [106] successfully used this approach to identify valid inequalities in their branch-and-cut algorithm for the RCSPP.

For some instances of the RCSPP, however, non-zero arc weights are given for many of the resources, and the constraint matrix may be very dense. In these cases, the single resource relaxations can become very weak. Thus, we would like to consider *jointly defined* inequalities that are derived using all of the resource constraints. Note that such an inequality should be relatively stronger and need not be valid for any single resource relaxation. Fortunately, most of the valid inequalities we have introduced for the WCSPP can be generalized to consider multiple resources without increasing the overall complexity of separation. In fact, separation for some of these jointly defined inequalities requires less total work than individually separating the original inequalities for all resources. We begin by generalizing the precedence inequalities.

For $(i,j) \in A$ such that $j \neq s,t$, the node precedence inequality (3.8) is given by

$$x_{ij} \leq x(F_{ij}^{+}),$$

where ϕ_{ij}^{+} is the set of arcs out of j which can appear in a weight feasible s-t walk passing through arc (i,j), and $F_{ij}^{+} = \left\{(j,k) \in \phi_{ij}^{+} : k \neq i\right\}$. In the case of multiple resources, we can redefine ϕ_{ij}^{+} to be the set of arcs out of j which can appear in a weight feasible s-t walk passing through arc (i,j) for each resource r, or

$$\phi_{ij}^{+} = \bigcap_{r=1}^{R} \phi_{ij}^{+w^r},$$

where

$$\phi_{ij}^{+w^r} = \left\{(j,k) \in \delta^+(j) : \sigma_{si}^{w^r} + w_{ij}^r + w_{jk}^r + \sigma_{kt}^{w^r} \leq W_r\right\}$$

for $r = 1, \ldots, R$. Using this new definition for ϕ_{ij}^+, we obtain a jointly defined version of inequality (3.8) which is stronger than the original. If $x^* \in [0,1]^A$ is a fractional solution, and $G^* = (V^*, A^*)$ is the corresponding support graph, then Algorithm 2 in Section 3.2 can be used exactly as stated to find all jointly defined inequalities violated by x^*. If the shortest weight path values have been calculated up front, each test of $(j,k) \in \phi_{ij}^+$ on line 5 now requires $\mathcal{O}(R)$ work. Therefore, the total work required is $\mathcal{O}(R|A^*||V^*|)$ which is equivalent to the work required if the original node precedence inequality (3.8) was individually separated for each resource.

A jointly defined version of the subpath precedence inequality (3.12) can also be separated with the same amount of computational effort necessary to individually separate the original inequality for each resource. For any infeasible s-t subpath $Q = (i_1, \ldots, i_p)$ in G such that $p \geq 3$, inequality (3.12) is given by

$$x_{i_1,i_2} \leq x(F_Q^+),$$

where $\phi_Q^+(i_k)$ is the set of arcs out of i_k which can appear in a weight feasible s-t walk passing through $(i_1, \ldots, i_k)$, $F_Q^+(i_k) = \left\{(i_k, j) \in \phi_Q^+(i_k) : j \neq i_{k+1},\ j \neq i_1, \ldots, i_{k-1}\right\}$ for $k = 2, \ldots, p-1$, and $F_Q^+ = \bigcup_{k=2}^{p-1} F_Q^+(i_k)$. In the case of multiple resources, we can redefine

$$\phi_Q^+(i_k) = \bigcap_{r=1}^{R} \phi_Q^{+w^r}(i_k),$$

where

$$\phi_Q^{+w^r}(i_k) = \left\{(i_k, j) \in \delta^+(i_k) : \sigma_{s,i_1}^{w^r} + \sum_{h=2}^{k} w_{i_{h-1},i_h}^r + w_{i_k,j}^r + \sigma_{jt}^{w^r} \leq W_r\right\}$$

for $r = 1, \ldots, R$. Using this new definition for $\phi_Q^+(i_k)$, we obtain a jointly defined version of inequality (3.12) which is stronger then the original and can be separated by Algorithm 3 in Section 3.3 exactly as stated. As long as a weight vector w_Q is updated each time Q is modified, each test of $(i_k, j) \in \phi_Q^+(i_k)$ on line 11 or $(i_k, j) \in \phi_{(i_2,\ldots,i_k)}^+(i_k)$ on line 17 now requires $\mathcal{O}(R)$ work if the shortest weight path values have been calculated up front.

Therefore, the total work required for any enumerated subpath in the fractional support graph $G^* = (V^*, A^*)$ is $\mathcal{O}(R|V^*||V|)$.

The final precedence inequality actually provides an opportunity to reduce the overall complexity of separation. For any s-t cut $S \subset V$ and $(i,j) \in A$ such that $j \in S \setminus \{s\}$, the s-t cut precedence inequality (3.15) is given by

$$x_{ij} \leq x(F_{ij}^+(S)),$$

where Φ_{ij}^+ is the set of all arcs which can appear in a weight feasible s-t walk passing through arc (i,j) first, and $F_{ij}^+(S) = \Big\{(k,l) \in \delta^+(S) : (k,l) \in \Phi_{ij}^+,\ k \neq i,\ l \neq i\Big\}$. By redefining

$$\Phi_{ij}^+ = \bigcap_{r=1}^{R} \Phi_{ij}^{+w^r},$$

where

$$\Phi_{ij}^{+w^r} = \big\{(k,l) \in A : \sigma_{si}^{w^r} + w_{ij}^r + \sigma_{jk}^{w^r} + w_{kl}^r + \sigma_{lt}^{w^r} \leq W_r\big\}$$

for $r = 1, \ldots, R$, we obtain a jointly defined version of inequality (3.15) which is stronger than the original and can be separated by Algorithm 4 in Section 3.4 exactly as stated. If the shortest weight path values have been calculated up front, each test of $(k,l) \in \Phi_{ij}^+$ on line 5 now requires $\mathcal{O}(R)$ work. However, only a single minimum s-t cut problem needs to be solved for each fractional arc in the support graph $G^* = (V^*, A^*)$. Therefore, the total work required is $\mathcal{O}(R|A^*|^2 + |A^*||V^*|^3)$ which is less than the work required if the original s-t cut precedence inequality (3.15) was individually separated for each resource, i.e., $\mathcal{O}(R|A^*||V^*|^3)$.

Although not included here, derivations for jointly defined versions of the reverse node precedence inequality (3.9), the reverse subpath precedence inequality (3.13) and the reverse s-t cut precedence inequality (3.16) proceed in much the same way by redefining the sets ϕ_{ij}^-, $\phi_Q^-(i_k)$ and Φ_{ij}^-. It is also possible to derive jointly defined versions of the strengthened precedence inequalities introduced in Section 3.5. For example, let us consider the strengthened node precedence inequality (3.17). Let $(i,j) \in A$ such that $j \neq s,t$. In the case of multiple resources, we can redefine ∇_{ij}^- to be the set of arcs into j which have shortest weight paths through them for each resource r that are at least as long as the shortest

weight s-j path through (i,j), or

$$\nabla_{ij}^{-} = \bigcap_{r=1}^{R} \nabla_{ij}^{-w^r},$$

where

$$\nabla_{ij}^{-w^r} = \left\{(k,j) \in \delta^{-}(j) : \sigma_{sk}^{w^r} + w_{kj}^{r} \geq \sigma_{si}^{w^r} + w_{ij}^{r}\right\}$$

for $r = 1, \ldots, R$. If we use the redefined set ϕ_{ij}^{+} above, then

$$x(\nabla_{ij}^{-}) \leq x(\phi_{ij}^{+})$$

is a jointly defined version of inequality (3.17). It should be noted that a tradeoff exists when going to multiple resources for this inequality. As R grows large, membership in ∇_{ij}^{-} (and ϕ_{ij}^{+}) becomes less likely unless the arc weights are highly correlated among resources. This causes the left hand side of the inequality to get weaker, while the right hand side gets stronger.

The conflict graph introduced in Section 3.6.2 also provides an opportunity for reducing the overall complexity of separation for any conflict graph inequalities such as the clique inequality (3.19) or the lifted odd hole inequality (3.21). The set of all node packings in the original conflict graph defines a node packing polytope which is a relaxation of the convex hull of feasible solutions to the WCSPP. This polytope can be formulated by adding an inequality $x_{ij} + x_{kl} \leq 1$ for all $(i,j) \in A$ and $(k,l) \in A \setminus F_{ij}$ such that $(k,l) \neq (i,j)$, where

$$F_{ij} = \left\{(k,l) \in A : (k,l) \in \Phi_{ij}^{-} \cup \Phi_{ij}^{+},\ k \neq i,\ l \neq j,\ (k,l) \neq (j,i)\right\}.$$

If we have multiple resources, we can use the redefined sets Φ_{ij}^{-} and Φ_{ij}^{+} discussed above to obtain a jointly defined version of the conflict graph that corresponds to a new node packing polytope which provides a stronger relaxation for X^{RCP}. If the shortest weight path values have been calculated up front, testing if a particular $(k,l) \in \Phi_{ij}^{-} \cup \Phi_{ij}^{+}$ for some $(i,j) \in A$ now requires $\mathcal{O}(R)$ work. Therefore, rather than building R conflict graphs for each resource, we can build a single graph with the same worst case complexity. Moreover, we only need to run our separation heuristics for inequalities (3.19) and (3.21) on one conflict graph rather than R of them. Thus, the total work required for separating jointly defined inequalities

which are stronger than the originals is actually less than the work required to separate individually for each resource.

The final inequality with potential for reduction in overall separation complexity is the path subset cover inequality (4.1). Although we have no jointly defined version of this inequality, we can simultaneously separate for all resources with a single execution of Algorithm 7 in Section 4.3. Each test of $(i_k, j) \in \phi_Q^+(i_k)$ on line 11 or $(i_k, j) \in \phi^+_{(i_2,\ldots,i_k)}(i_k)$ on line 14 now requires $\mathcal{O}(R)$ work if we use the jointly defined version of $\phi_Q^+(i_k)$ above and the shortest weight path values have been calculated up front. The savings comes from the fact that whenever we discover a minimal infeasible s-t subpath, it is minimal with respect to some resource r, and we can call MINIMAL_PATH_SUBSET_COVER for that single resource. Therefore, the total work required for any enumerated subpath in the fractional support graph $G^* = (V^*, A^*)$ is $\mathcal{O}(R|V^*|^2 + |V^*|^3\mathcal{SP}(V, A))$, which is less than R times the original complexity.

Table 2 summarizes the complexity of jointly defined separation for each of the various inequalities mentioned in this section. The work required to calculate shortest weight paths up front, or the *initial complexity*, is presented in a separate column because it only needs to be done once when separating multiple fractional solutions.

The only inequality which remains to be discussed is the lifted path subset cover inequality (4.2) which was introduced for acyclic G. A jointly defined version of this inequality can be obtained by separating minimal s-t path subset covers with respect to a single resource (using Algorithm 5 in Section 4.4), and then using multiple resources in our lifting problem to compute stronger lifting coefficients. The drawback to this approach is that we lose the ability to solve the lifting problem exactly in polynomial time. Exact lifting is still possible in pseudopolynomial time with a typical labeling algorithm if all arc weights are integral and non-negative. Approximate lifting is also possible using LP relaxations, Lagrangean relaxations, or K shortest paths.

Table 2: Separation complexity for multiple resources

Inequality Class	*Initial Complexity*	*Separation Routine and Complexity*
node precedence (3.8) and (3.9)	$\mathcal{O}(\mathcal{SP}(V,A)R)$	$\mathcal{O}(R\|A^*\|\|V^*\|)$ for Algorithm 2
subpath precedence† (3.12) and (3.13)	$\mathcal{O}(\mathcal{SP}(V,A)R)$	$\mathcal{O}(R\|V^*\|\|V\|)$ per fractional s-t path for Algorithm 3
s-t cut precedence‡ (3.15) and (3.16)	$\mathcal{O}(\|V\|\mathcal{SP}(V,A)R)$	$\mathcal{O}(R\|A^*\|^2 + \|A^*\|\|V^*\|^3)$ for Algorithm 4
clique†‡ (3.19)	$\mathcal{O}(\|V\|\mathcal{SP}(V,A)R)$	$\mathcal{O}(R\|A^*\|^2)$ for conflict graph§ + Greedy max weight cliques
lifted odd hole†‡ (3.21)	$\mathcal{O}(\|V\|\mathcal{SP}(V,A)R)$	$\mathcal{O}(R\|A^*\|^2)$ for conflict graph§ + Hoffman and Padberg [66] scheme
path subset cover†‡ (4.1)	$\mathcal{O}(\mathcal{SP}(V,A)R)$	$\mathcal{O}(R\|V^*\|^2 + \|V^*\|^3\mathcal{SP}(V,A))$ per fractional s-t path for Algorithm 7

†Heuristic separation routine

‡Jointly defined separation has improved complexity

§Fractional conflict graph is used for separation

5.3.4 Locally Valid Inequalities

The subproblem at any node n in the branch-and-bound tree is itself an RCSPP with a subset of the arcs (variables) in A fixed to zero or one. This subproblem has its own set of valid inequalities which are also valid for any of its children. Most of the inequalities that have been introduced for the RCSPP make frequent use of the shortest weight paths in G with respect to some resource constraint. Therefore, we can derive inequalities that are *locally valid* for the subtree rooted at n by using the shortest weight paths in the subgraph of G implied by the arc fixings at n. This results in larger shortest weight path values and stronger inequalities that are not necessarily valid for X^{RCP}.

Fixings to zero are easily handled by removing arcs from G, but as mentioned in Section 4.2, fixings to one are more difficult to enforce (without additional constraints) unless G is acyclic. For these inequalities to be valid, however, it suffices to use lower bounds on the shortest path values. We can obtain these lower bounds by disregarding the implications of the arcs that are fixed to one. In other words, we can find shortest weight paths in $G \setminus A_n^0$,

where A_n^0 is the set of arcs that have been fixed to zero at n.

Of course separating these locally valid inequalities requires additional work. For inequalities that are (globally) valid for the entire tree, we need to calculate the shortest paths only once, which requires $\mathcal{O}(\mathcal{SP}(V, A)R)$ or $\mathcal{O}(|V|\mathcal{SP}(V, A)R)$ work up front. However, the locally valid inequalities require us to recalculate these shortest paths at each node in the branch-and-bound tree for which they are separated. Therefore, they should be used judiciously.

5.4 Branching Schemes

After cutting planes have been added to the LP relaxation at any node in the branch-and-bound tree, the conventional strategy branches on some fractional arc (variable) if the solution to the LP relaxation is not integral. In this section, we improve this strategy by using the subgraph of G implied by a set of arc fixings to preprocess the subproblem at each node in the tree. We also consider some alternative branching schemes that may result in branch-and-bound trees which are more balanced.

5.4.1 Arc Branching and Preprocessing

Let $G_n = (V_n, A_n)$ be the subgraph of G implied by the arc fixings at any node n in the branch-and-bound tree. If we branch on a fractional arc (i, j) at n, then we must create two new subproblems with x_{ij} fixed to zero and one. If we restrict ourselves to the acyclic case, this is equivalent to creating two new restricted RCSPPs with underlying graphs $G_n \setminus \{(i, j)\}$ (fixed to zero) and $G_n[(i, j)]$ (fixed to one).

If we fix x_{ij} to zero, $G_n \setminus \{(i, j)\}$ may contain nodes that can no longer be found in any s-t path after the removal of (i, j). If this is the case, then $G_n \setminus \{(i, j)\}$ violates (A1), and we can call the trivial routine presented in Algorithm 11 to restore this property with two breadth-first searches in $\mathcal{O}(|A|)$.

For each node i such that i cannot be reached from s, or t cannot be reached from i, the function ST_CONNECTED removes i from the graph along with all of its incident arcs. If t is unreachable from s (the condition tested on line 3), then the graph has no feasible s-t

Algorithm 11 Assuring $G = (V, A)$ is s-t connected

```
1: ST_CONNECTED(G):
2: find ρ⁺(s) for G
3: if t ∉ ρ⁺(s) then
4:    stop G has no feasible s-t path
5: end if
6: find ρ⁻(t) for G
7: for all i ∉ ρ⁺(s) ∩ ρ⁻(t) do
8:    V ← V \ {i}
9:    A ← A \ (δ⁻(i) ∪ δ⁺(i))
10: end for
```

path, and the corresponding RCSPP is infeasible. Any arc removed when ST_CONNECTED is called for $G_n \setminus \{(i,j)\}$ can also be set to zero along with (i,j) in the new subproblem.

If we fix x_{ij} to one, we can define $G_n[(i,j)] = (V_n[(i,j)], A_n[(i,j)])$ in $\mathcal{O}(|A|)$ and set all of the arcs in $A_n \setminus A_n[(i,j)]$ to zero in the new subproblem. Note that if the graph is not acyclic, we can still use the cyclic approximation for $G_n[(i,j)]$ discussed in Section 4.2. This is accomplished by initially removing any arcs $(k,l) \in \delta^+(i) \cup \{(j,i)\} \cup \delta^-(j) \subseteq A_n$ such that $(k,l) \neq (i,j)$ to arrive at the reduced graph $G'_n = (V_n, A'_n)$. Then, we can define $G'_n[(i,j)]$ and set all of the arcs in $A_n \setminus A'_n[(i,j)]$ to zero in the new subproblem.

Remark 18. *By definition, $G_n[(i,j)]$ will always satisfy (A1).*

The number of arcs in A_n which are set to zero in each of the new subproblems can be increased by repeatedly calling the function AAN_PREPROCESSING on $G_n \setminus \{(i,j)\}$ and $G_n[(i,j)]$ until the subproblems are shown to be infeasible or no other reductions are possible. This preprocessing can be expensive since each call to AAN_PREPROCESSING requires $\mathcal{O}(\mathcal{SP}(V,A)R)$ work. Therefore, it might be wise to use an iteration limit on the number of calls per graph. Not only do these preprocessing schemes strengthen the LP relaxation for any node in the subtree rooted at n, but they also increase the values of the shortest weight paths for the corresponding underlying graphs. As a result, this helps to improve any locally valid inequalities because longer shortest weight paths imply stronger inequalities.

5.4.2 Alternative Branching Schemes

A drawback of the arc branching scheme for the RCSPP is that one branch is much more restrictive than the other, and the branch-and-bound tree tends to be unbalanced. If we fix an arc to zero, the set of feasible s-t paths does not change much, and relatively little progress is made in our search. Thus, fixing an arc to one has a much stronger impact than fixing it to zero.

One way to make the tree more balanced is to use *node branching* and fix the flow through some fractional $i \in V \setminus \{s,t\}$ to zero or one. Fixing to zero becomes stronger because fixing node i implies that all of the arcs in the set $\delta^-(i) \cup \delta^+(i)$ are fixed to zero. Conversely, fixing to one becomes weaker since fixing arc (i,j) implies that nodes i and j are both fixed to one. It is easy to show that any solution $x^* \in \mathbb{R}^A$ is fractional if and only if there exists some $i \in V \setminus \{s,t\}$ such that $0 < x^*(\delta^+(i)) < 1$. Hence, this branching scheme is enough to enforce integrality.

Let G_n be the subgraph of G implied by the fixings at any node n in the branch-and-bound tree. If we restrict ourselves to the acyclic case, branching on a fractional node i is equivalent to creating two new restricted RCSPPs with underlying graphs $G_n \setminus \{i\}$ (fixing to zero) and $G_n[(i)]$ (fixing to one). This is convenient because additional constraints are not needed for either of the new subproblems. If the graph is not acyclic, we can still use the cyclic approximation for $G_n[(i)]$, but we must also add the constraint $x(\delta^+(i)) = 1$ to all subproblems in the subtree rooted at n to ensure that each feasible s-t path contains i. It should be noted that the same preprocessing discussed for arc branching applies here as well, and at the very least, we should call ST_Connected for $G_n \setminus \{i\}$.

In Chapter 6, we will evaluate node branching along with a few approaches for how to choose which fractional node in G to branch on. However, there are several other straightforward alternative branching schemes for the RCSPP. We mention three of them here, but postpone their investigation for future research.

The first alternative is the use of special ordered set (SOS) branching with the GUB inequalities (5.6) and (5.7). For an overview of the SOS branching scheme, see [89]. To implement this scheme, however, one needs to define a logical ordering on the arcs in each

GUB. One possibility is to sort them by their shortest path values, e.g., a weighted sum

$$\alpha_c(\sigma^c_{si} + c_{ij} + \sigma^c_{jt}) + \sum_{r=1}^{R} \alpha_r(\sigma^{w^r}_{si} + w^r_{ij} + \sigma^{w^r}_{jt})$$

for some $\alpha_c, \alpha_1, \ldots, \alpha_R \in \mathbb{R}$.

The next alternative is to fix s-t subpaths in G. Sometimes, it is beneficial to have an unbalanced tree. For example, we may want to dive in the tree and find a feasible solution with as few fixings as possible. If Q is a fractional s-t subpath in G such that the flow along Q is relatively large, the intuition is that Q might be contained in high quality feasible solutions. We can force all paths to contain Q in one subproblem by setting the arcs in $A_n \setminus A_n[Q]$ to zero and adding the constraint $x(Q) = |A(Q)|$ (needed only if G is not acyclic). To prohibit Q in the other subproblem, we add can add the infeasible subpath inequality $x(Q) \leq |A(Q)| - 1$. If the flow through Q is large enough, this inequality will cut off the current fractional solution.

The final alternative is one that has been used for the TSP. If the fractional support graph G^* contains cycles, we can branch on the value of $\delta^+(S)$ for some $S \subseteq V \setminus \{s, t\}$. Priority can also be given to fractional arcs which appear in disjoint subgraphs in G^* because branching on them first may help to keep G^* connected and eliminate subtours.

5.5 Primal Heuristics

Any feasible solution for the RCSPP provides an upper bound on the optimal value. The strength of this bound determines the likelihood of fathoming nodes in the branch-and-bound tree. Before the branch-and-cut algorithm begins, any RCSPP heuristic proposed in the literature can be used to find a good solution. As the search progresses, this bound is automatically strengthened each time a node in the branch-and-bound tree is fathomed by optimality and produces a new incumbent solution. In this section, we would like to incorporate RCSPP heuristic schemes which search for additional feasible solutions at any node in the search tree. These schemes utilize the information provided by the current fractional solution and the best integer solution found with the hope that both of them will give an indication as to which arcs will be present in high quality feasible solutions.

5.5.1 MIP-based Heuristics

Many general purpose heuristics have been proposed in the literature for MIPs with no specific structure. Most modern solvers use some variant of these heuristics to find new feasible solutions throughout the search tree. We mention two of the more recent schemes here which we intend to adapt for use on the RCSPP.

Fischetti and Lodi [43] introduced a variant of local search for MIPs called *local branching*. They use *soft fixing* to find solutions that are close to the current incumbent $\bar{x}$ by solving an auxiliary MIP with the additional constraint

$$\sum_{j \in B: \bar{x}_j = 1} (1 - x_j) + \sum_{j \in B: \bar{x}_j = 0} x_j \leq \Delta, \tag{5.12}$$

where B is the set of binary variables in the original MIP, and Δ is a neighborhood radius parameter. If Δ is small, the auxiliary MIP should only explore a small neighborhood of solutions which are similar to $\bar{x}$ because the left hand side of the soft fixing constraint (5.12) is a metric which measures the hamming distance from $\bar{x}$. The hope is that the neighborhood contains improved solutions, and the auxiliary MIPs are much easier to solve after this constraint has been added.

Danna et al. [24] proposed a similar scheme called *Relaxation Induced Neighborhood Search* (RINS) which also uses auxiliary MIPs to explore neighborhoods. RINS differs from local branching by using the current fractional solution to the LP relaxation (along with the incumbent) to guide the neighborhood selection. Rather than using a soft fixing constraint, the authors use *hard fixing*. At any node of the branch-and-cut tree, variables that have common integral values in both the LP solution and the incumbent are fixed. This results in a smaller MIP that is easier to solve because it involves fewer variables. While local branching only explores a new neighborhood each time a different incumbent is found, RINS defines a new neighborhood at each node in the branch-and-cut tree and explores a neighborhood of both the fractional solution and the incumbent. The ever changing fractional solution provides a built-in diversification mechanism that allows us to explore a variety of neighborhoods throughout the search tree.

5.5.2 MIP-based Heuristics for the RCSPP

One of the drawbacks of the general purpose heuristics discussed above is that solving an auxiliary MIP is also $\mathcal{NP}$-hard, and it is not a very reliable mechanism for rapid neighborhood searches. In practice, node limits are enforced to end these MIP subproblems prematurely. We would like to use the ideas of soft fixing and hard fixing combined with less exhaustive neighborhood searches that have improved complexity guarantees. This may reduce the chances of finding improved solutions for a given neighborhood, but we can be less selective about which nodes in the branch-and-bound tree we attempt the searches at and explore many more neighborhoods. If we can enforce either of the fixings without changing the structure of the RCSPP, we can use any pre-existing heuristics for the RCSPP on an auxiliary problem which should be easier to work on.

The soft fixing constraint (5.12) can be viewed as an additional resource. However, care must be taken when adding this resource constraint because most heuristics for the RCSPP assume non-negative arc weights. If the heuristic we intend to implement requires non-negativity, then we can add the weaker constraint

$$\sum_{(i,j)\in A:\bar{x}_{ij}=0} x_{ij} \leq \Delta.$$

This inequality only considers deviations on arcs not used in the incumbent, and Δ should be adjusted accordingly. If G is acyclic, hard fixing is equivalent to a new restricted RCSPP with the underlying graph $G'[A_1]$, where $G' = G \setminus A_0$, and A_δ is the set of arcs in both the fractional solution and the current incumbent with common value δ for $\delta = 0, 1$. If G is not acyclic, then we must approximate $G'[A^1]$.

Once we have introduced the additional resource for soft fixing or defined the underlying subgraph for hard fixing, we can repeatedly call the function AAN_Preprocessing to preprocess the auxiliary RCSPP and make the corresponding neighborhood even smaller. If we can assume that no negative cost cycles exist in G, the function DB_Preprocessing is preferable because it might also identify a new incumbent solution.

After the auxiliary problem has been preprocessed, we can use any RCSPP heuristic to search for a solution, e.g., find the K shortest s-t paths for some fixed K and test each for

feasibility. However, this heuristic and most others assume that no negative cost cycles exist in G. If we cannot make this assumption, we present an alternative scheme which implicitly does hard fixing while generating random s-t paths in G. This can be accomplished by starting at node s and randomly following any arc out of the current node i using the probabilities defined by

$$p_{ij} = \frac{x^*_{ij} + \bar{x}_{ij}}{x^*(\delta^+(i)) + \bar{x}(\delta^+(i))}, \ \forall \ (i,j) \in \delta^+(i),$$

where x^* is the current fractional solution and $\bar{x}$ is the incumbent. These probabilities ensure that any arc which is fixed at zero in both solutions will never be followed, and any arc which is fixed at one in both solutions will always be followed. If t is reached (without revisiting any nodes) and our s-t path P satisfies $c(P) < c\bar{x}$ and $w^r(P) \leq W_r$ for all $r = 1, \ldots, R$, then we have found an improving solution and we update $\bar{x}$.

It should be noted that at least one feasible solution is necessary to use both soft and hard fixing. If no solution has been found, we can use the current fractional support graph G^* as the underlying graph for our auxiliary problem. The hope is that one of the longest paths in G^* might be weight feasible for all resources. While this is a somewhat naive approach, this path may allow us to begin using the more sophisticated schemes discussed above.

CHAPTER 6

COMPUTATIONAL EXPERIMENTS FOR THE RESOURCE CONSTRAINED SHORTEST PATH PROBLEM

6.1 Introduction

Now that the framework for our branch-and-cut algorithm has been described, we are ready to investigate its performance on various instances of the resource constrained shortest path problem (RCSPP). Since there are many different choices involved in the implementation of each component in the algorithm, comparing the performance for all possible combinations of these choices is practically impossible. Thus, we begin with a set of baseline computational results provided via CPLEX [69], a leading integer programming software package. Each instance is solved using CPLEX' branch-and-cut algorithm with the default parameter settings. Then, one at a time, we incorporate the four major components discussed in Chapter 5, i.e., preprocessing techniques, branching schemes, primal heuristics and cutting planes. For each of these components, we present computational experiments which compare a few combinations of the implementation choices. Once the best combination has been identified for the current component, those choices are fixed, and the corresponding results become the new baseline for the next component to be incorporated.

The ultimate goal of this computational study is to (1) justify the use of each component in our algorithm, (2) provide empirical guidance for the choices involved in the implementation of the algorithm, and (3) demonstrate that our algorithm can outperform a generic branch-and-cut implementation. While it is possible to provide comparisons with alternative solution approaches such as traditional labeling methods, we leave such investigations as a line of future research. In this chapter, we are focused on verifying a set of fundamental components that can be applied in situations where branch-and-cut becomes a viable

option, e.g., pricing problems in column generation algorithms with negative cost cycles. Moreover, branch-and-cut may be our only option for cases in which the RCSPP appears as a substructure in a more difficult problem like the extension in [7] or the dial-a-flight problem in [39, 40]. In these cases, we can solve the problem more effectively by applying extended versions of these components for an improved branch-and-cut algorithm.

All computational experiments were performed on a cluster of Sun X2100 machines. Each machine has a dual-core AMD Opteron 175 processor, running at 2.2 GHz, and 4 GB of RAM. The source code was written in the C++ programming language, compiled using GCC version 3.2 with code optimization -O3, and executed in Red Hat Enterprise Linux AS, release 4. All instances were solved using CPLEX 9.0, and each branch-and-cut component in our algorithm was incorporated using callback functions available in the CPLEX callable library. A time limit of 3600 CPU seconds was imposed in the solution of each instance, including any preliminary preprocessing prior to branch-and-cut. However, it should be noted that the time taken to read any problem input was not accounted for in the effective running times.

6.2 Problem Classes

The RCSPP instances used in this chapter are divided into six problem classes: A1-A3 and C1-C3. If an instance belongs to a class whose name begins with the letter 'A', then the underlying graph is acyclic; otherwise, the graph contains cycles. In this section, we describe each of the classes considered.

Beasley and Christofides [11] developed a set of RCSPP instances, and we partitioned them into our first two problem classes, A1 and C1. Of the 24 instances in [11], instances 3-4, 7-8, 11-12, 15-16, 19-20 and 23-24 have acyclic graphs and are included in class A1; the rest belong to class C1. As we will see, these problem classes are easily solved with branch-and-cut. One of the reasons for this is that all arc costs are non-negative. If we modify the instances in [11] by multiplying all of the arc costs by -1, we get our next two problem classes, A2 and C2, which prove to be more challenging.

The attributes for problem classes A1-A2 and C1-C2 can be found in Tables 3 and 4.

For reference purposes, we use instance numbers which correspond to the numbers used in [11]. For each instance, we report the size of the node and arc sets in the underlying graph $G = (V, A)$, the number of resource constraints (R), the value of the unconstrained shortest cost s-t path in G (z_{SP}), the value of the original linear programming relaxation with no cutting planes added (z_{LP}), the optimal value (z^*), and the gap between z^* and z_{LP} (LP Gap%) which is computed as $100 \times |(z^* - z_{LP})/z^*|$.

Table 3: RCSPP problem classes A1 and A2

				A1 (positive arc costs)				A2 (negative arc costs)			
Instance	$\|V\|$	$\|A\|$	R	z_{SP}	z_{LP}	z^*	LP Gap%	z_{SP}	z_{LP}	z^*	LP Gap%
3	100	959	1	1	1.5	2	25.0	-179	-78.8	-78	1.0
4					2.0	2	0.0		-74.3	-74	0.3
7		999	10	3	4.2	6	30.7	-184	-32.7	-29	12.6
8					5.4	14	61.6		-28.2	-23	22.5
11	200	1,971	1	6	6.0	6	0.0	-226	-125.6	-125	0.5
12					6.0	6	0.0		-119.5	-119	0.4
15		1,960	10	5	6.9	9	23.8	-223	-33.2	-24	38.4
16					9.0	17	47.1		-28.8	-17	69.5
19	500	4,978	1	6	6.0	6	0.0	-314	-99.0	-99	0.0
20					6.0	6	0.0		-92.0	-92	0.0
23		4,868	10	3	3.5	4	12.7	-277	-45.2	-36	25.4
24					4.3	5	14.8		-38.2	-29	31.7
Average	267	2,623	6				18.0				16.9

Table 4: RCSPP problem classes C1 and C2

				C1 (positive arc costs)				C2 (negative arc costs)†		
Instance	$\|V\|$	$\|A\|$	R	z_{SP}	z_{LP}	z^*	LP Gap%	z_{LP}	z^*	LP Gap%
1	100	955	1	80	89.0	131	32.0	-2,256.8	-2,119	6.5
2					98.0	131	25.2	-2,018.8	-1,868	8.1
5		990	10	79	83.9	100	16.1	-1,748.8	-1,515	15.4
6					88.6	100	11.4	-1,531.4	-1,279	19.7
9	200	2,040	1	230	356.7	420	15.1	-924.7	-808	14.4
10					420.0	420	0.0	-808.0	-808	0.0
13		2,080	10	200	292.4	448	34.7	-1,099.0	-593	85.3
14					403.5	◇	◇	-869.7	◇	◇
17	500	4,858	1	455	488.6	652	25.1	-41,519.0	-40,731	1.9
18					522.1	652	19.9	-37,331.4	-36,549	2.1
21		4,847	10	611	678.4	858	20.9	-5,374.8	-4,221	27.3
22					768.2	858	10.5	-4,571.0	-3,657	25.0
Average	267	2,628	6				17.6			17.2

† Shortest path with negative cost cycles is strongly $\mathcal{NP}$-hard
◇ Problem is infeasible

The degree of difficulty for any RCSPP is also highly dependent on the size of the underlying graph. To challenge our branch-and-cut algorithm with larger graphs, we randomly

generated our final two problem classes, A3 and C3, with schemes that were very similar to what was used for class A2. Like A2, all arc costs were randomly generated integers between -5 and 0, and all arc weights were randomly generated integers between 0 and 5. The only discrepancy between the schemes used for classes A3 and C3 was in their graph generation approaches.

All of the graphs in class A3 were generated using the approach in [11] for acyclic graphs. Let n be the total number of desired nodes, and $V = \{1, 2, \ldots, n\}$ with $s = 1$ and $t = n$. To enforce a minimum cardinality on any s-t path, we randomly include arc $(i, j) \in A$ for all $i = 1, \ldots, n-1$ and $j = i+1, \ldots, \min\{n, i + \lfloor n/4 \rfloor\}$. The probability of including arc (i, j) is strictly a function of n and chosen such that the expected value of $|A|$ is $10n$. By construction, the graph $G = (V, A)$ will always be acyclic.

The graphs in class C3 were generated using a slightly modified version of the approach in [11] for cyclic graphs which was originally outlined by Handler and Zang [61]. In our scheme, each node $i \in V = \{1, 2, \ldots, n\}$ is associated with coordinates x_i and y_i inside an $n \times n$ square in $\mathbb{R}^2$. The source and sink nodes are always located at opposite corners in this square, i.e., $x_1 = y_1 = 0$ and $x_n = y_n = n$. For all other nodes $i = 2, \ldots, n-1$, we randomly generate coordinates from independent uniform $(0, n)$ distributions. A minimum cardinality on any s-t path is enforced by randomly including arc $(i, j) \in A$ for all $i = 1, \ldots, n$ and $j = 1, \ldots, n$ such that $i \neq j$ and $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \leq n\sqrt{2}/4$. Once again, the probability of including arc (i, j) is chosen so that the expected value of $|A|$ is $10n$.

In [11], Beasley and Christofides determine the resource capacities for each instance by calculating the amount of resources used by the shortest cost s-t path and reducing them by a certain percentage. By increasing this percentage, they get "tighter" resource constraints. For each of the instances in classes A3 and C3, we used a fixed capacity W for all resources. In an attempt to guarantee feasibility while varying W for tighter constraints, we chose $W \geq 20$ since any s-t path in either graph type has at least 4 arcs, and the maximum weight on any arc is 5.

To obtain the instances in class A3, we randomly generated two sets of graphs and arc costs for $n \in \{1000, 1250, 1500\}$. Arc weights for $R \in \{5, 10\}$ were generated for each of

these six sets, and by varying $W \in \{20, 30, 40\}$, we arrived at a total of 36 instances. The same process was repeated for class C3 with $n \in \{500, 750, 1000\}$. The attributes for both classes can be found in Tables 5 and 6. Note that there are infeasible instances in class C3.

Table 5: RCSPP problem class A3

Instance	$\|V\|$	$\|A\|$	R	W	z_{SP}	z_{LP}	z^*	LP Gap%
1	1,000	10,010	5	20	-241	-50.4	-44	14.5
2				30		-71.6	-64	11.8
3				40		-90.7	-85	6.7
4			10	20		-42.1	-32	31.6
5				30		-62.8	-54	16.2
6				40		-82.1	-73	12.4
7		10,012	5	20	-239	-52.3	-44	18.9
8				30		-75.8	-70	8.4
9				40		-94.3	-89	6.0
10			10	20		-46.4	-33	40.6
11				30		-69.0	-60	15.1
12				40		-88.3	-79	11.8
13	1,250	12,451	5	20	-262	-50.5	-42	20.3
14				30		-73.7	-67	10.0
15				40		-94.2	-89	5.9
16			10	20		-44.0	-32	37.6
17				30		-67.3	-55	22.3
18				40		-86.8	-78	11.3
19		12,497	5	20	-256	-56.2	-50	12.4
20				30		-83.9	-74	13.4
21				40		-105.7	-97	8.9
22			10	20		-46.4	-36	28.9
23				30		-68.8	-59	16.6
24				40		-88.2	-79	11.7
25	1,500	14,993	5	20	-252	-50.4	-46	9.6
26				30		-71.3	-65	9.7
27				40		-91.3	-86	6.1
28			10	20		-44.4	-35	27.0
29				30		-64.9	-55	18.0
30				40		-83.6	-74	12.9
31		15,016	5	20	-250	-53.4	-45	18.7
32				30		-72.6	-68	6.8
33				40		-91.0	-86	5.8
34			10	20		-46.1	-36	28.0
35				30		-67.4	-57	18.2
36				40		-85.4	-77	10.9
Average	1,250	12,497	8					15.7

By construction, all of the instances in problem classes A2-A3 and C2-C3 have non-positive arc costs. Moreover, all of the instances in classes C2 and C3 are strongly $\mathcal{NP}$-hard because they contain negative cost cycles. These instances are meant to emulate challenging column generation pricing problems with a lot of negative reduced costs, or routing problems with incentives for visiting nodes in the graph.

Table 6: RCSPP problem class C3

Instance	$\|V\|$	$\|A\|$	R	W	z_{LP}	z^*	LP Gap%
1	500	4,979	5	20	-55.7	-43	29.5
2				30	-91.8	-81	13.3
3				40	-125.7	-114	10.3
4			10	20	-40.3	◇	◇
5				30	-71.1	-59	20.5
6				40	-98.4	-88	11.8
7		4,974	5	20	-50.3	-41	22.8
8				30	-90.8	-82	10.8
9				40	-128.4	-119	7.9
10			10	20	-37.0	◇	◇
11				30	-66.1	-56	18.0
12				40	-93.0	-83	12.0
13	750	7,532	5	20	-41.5	-35	18.5
14				30	-80.4	-73	10.1
15				40	-113.7	-105	8.3
16			10	20	-29.0	◇	◇
17				30	-69.2	-57	21.5
18				40	-97.8	-85	15.0
19		7,602	5	20	-51.1	-38	34.4
20				30	-90.2	-80	12.7
21				40	-126.5	-117	8.1
22			10	20	-36.1	◇	◇
23				30	-68.9	-62	11.2
24				40	-98.1	-85	15.5
25	1,000	9,975	5	20	-36.6	◇	◇
26				30	-80.4	-69	16.5
27				40	-117.8	-106	11.1
28			10	20	-29.8	◇	◇
29				30	-62.1	-49	26.8
30				40	-90.4	-77	17.4
31		10,110	5	20	-50.2	-40	25.4
32				30	-85.8	-75	14.4
33				40	-120.4	-111	8.5
34			10	20	-39.4	◇	◇
35				30	-69.6	-59	17.9
36				40	-97.2	-86	13.0
Average	750	7,529	8				12.9

◇ Problem is infeasible

6.3 Default CPLEX Experiments

In search of a benchmark, we solved all of the RCSPP instances using CPLEX with the default parameter settings. With these settings, CPLEX preprocesses each instance before invoking branch-and-cut by applying a presolver and aggregator designed for general mixed integer programs (MIPs). By default, the branch-and-cut algorithm generates cutting planes that are valid for general MIPs such as Gomory fractional cuts, knapsack cover cuts, and generalized upper bound (GUB) cover cuts. However, these CPLEX cuts are not enough to guarantee feasibility for the instances in problem classes C2 and C3 since they contain negative cost cycles. To forbid these cycles, we added the trivial GUB inequalities (5.6) and the subtour elimination inequalities (5.8) discussed in Section 5.3.

There are only $\mathcal{O}(|V|)$ GUB inequalities, and they were implicitly added a priori with the callable library function `CPXaddlazyconstraints`. CPLEX assures that any inequality added with this function is always satisfied by all integer solutions. They are placed in a cut pool and only added to the formulation if they are violated.

The subtour elimination inequalities were separated for each solution x^* to the linear programming (LP) relaxations in the branch-and-bound tree, including integral ones, using the cut callback functionality in the callable library. Let $G^* = (V^*, A^*)$ be the corresponding support graph for some solution x^*. We add an inequality (5.8) with $S = V(H)$ for all disjoint subgraphs H of G^* such that $V(H) \cap \{s, t\} = \emptyset$ and $x^*(A(H)) > |V(H)| - 1$. All of the disjoint subgraphs can be identified using a single depth-first search on G^*. Therefore, the total work required for this separation routine is $\mathcal{O}(|A^*|)$ for each solution x^*, and it is not difficult to show that this is enough to guarantee an elementary s-t path.

For a comparison, we also solved the instances with all CPLEX cuts turned off, i.e., `CPX_PARAM_CUTPASS` set to -1. The full results of both experiments can be found in Appendix A in Table 21. The CPX setting refers to default CPLEX, and the CPX-C setting refers to CPLEX with its cuts turned off. For each instance and setting, we report several performance metrics such as the number of constraints in the formulation after preprocessing (Rows), the number of variables in the formulation after preprocessing (Cols), the

number of nonzeros in the constraint matrix after preprocessing (Nzs), the number of non-CPLEX cutting planes added (Cuts), the value of the best integer solution found (z_{IP}), the gap between z_{IP} and the value of the LP relaxation at the root node (Root Gap%) which is computed as $100 \times |(z_{IP} - z_{root})/z_{IP}|$, the gap between z_{IP} and the value of the best lower bound in the branch-and-bound tree (Final Gap%) which is computed as $100 \times |(z_{IP} - z_{LB})/z_{IP}|$, the number of nodes evaluated in the branch-and-bound tree before the best integer solution was found (z_{IP} Node), the total number of evaluated nodes in the branch-and-bound tree (B&B Nodes), the time spent separating non-CPLEX cuts (t_C), and the effective running time of the branch-and-cut algorithm (t). If an instance was not solved to proven optimality, then the time limit of 3600 CPU seconds was reached. The reader can deduce which of the instances timed out by checking the final integrality gap.

We summarize these results in Table 7. For each problem class and setting, we report the averages for most of the performance metrics. We also report the total difference between the true optimal value and the value of the best integer solution found (Δz^*), the number of instances that were solved at the root node (#rt), and the number of instances which timed out (#to). These results confirm that the instances developed by Beasley and Christofides in [11] are easily solved with CPLEX, and introducing negative arc costs provides more of a challenge. The largest classes, A3 and C3, prove to be the most difficult, with two instances timing out for class C3.

We see in Table 7 that the CPLEX cuts decreased the integrality gaps at the root node for all of the problem classes, yet this led to a marginal decrease in the number of branch-and-bound nodes evaluated for only two of the classes. Although the CPLEX cut statistics are not reported in the tables, it is worth mentioning that there were a total of 512 CPLEX cuts added for all of the instances, and 506 of them were Gomory fractional cuts. This indicates that the general purpose cover cuts were not very useful for these RCSPPs. Turning CPLEX cuts off also resulted in less cycle elimination inequalities for classes C2 and C3. However, it should be noted that the number of non-CPLEX cuts reported is not entirely accurate. If an instance was proven to be infeasible, CPLEX did not provide the number of lazy GUB cuts added to the formulation.

Table 7: Summary of the RCSPP default CPLEX results

Class	Setting	Avg Rows	Avg Cols	Avg Nzs	Avg Cuts	Avg Root Gap%	Avg Final Gap% (Δz^*)	Avg z_{IP} Node	Avg B&B Nodes (#rt)	Avg t_C	Avg t (#to)
A1	CPX	255	2,465	16,567	0	10.1	0.0 (0)	26	36 (9)	0.0	0.58 (0)
	CPX-C	255	2,465	16,567	0	18.0	0.0 (0)	22	27 (7)	0.0	0.36 (0)
A2	CPX	255	2,465	16,567	0	14.8	0.0 (0)	139	270 (6)	0.0	10.22 (0)
	CPX-C	255	2,465	16,567	0	16.9	0.0 (0)	60	166 (6)	0.0	6.10 (0)
A3	CPX	1,077	10,452	85,003	0	14.0	0.0 (0)	1,871	4,000 (0)	0.0	595.99 (0)
	CPX-C	1,077	10,452	85,003	0	15.7	0.0 (0)	1,355	2,710 (0)	0.0	375.80 (0)
C1	CPX	272	2,363	17,085	0	0.4	0.0 (0)	0	1 (10)	0.0	0.13 (0)
	CPX-C	272	2,363	17,085	0	17.6	0.0 (0)	2	6 (4)	0.0	0.15 (0)
C2	CPX	272	2,363	17,085	177	12.6	0.0 (0)	1,997	2,654 (2)	0.2	8.26 (0)
	CPX-C	272	2,363	17,085	161	15.7	0.0 (0)	1,882	2,681 (2)	0.3	10.85 (0)
C3	CPX	757	7,526	61,252	47	11.7	0.4 (2)	6,272	12,517 (1)	4.6	696.95 (2)
	CPX-C	757	7,526	61,252	41	12.8	0.4 (2)	4,749	9,860 (0)	4.3	578.89 (2)

When comparing results in this chapter, we will use running time as our primary performance metric. Turning off all CPLEX cuts led to smaller run times for the majority of the problem classes, and the improvement was most drastic for the hardest classes. Therefore, we turned off CPLEX cuts for all subsequent experiments in this chapter.

6.4 Preprocessing and Probing Experiments

CPLEX' presolver and aggregator were both successful in reducing the size of many of the formulations, but only a single RCSPP instance had its initial LP relaxation value (before any cuts were added) changed as a result of this preprocessing. In this section, we investigate the impact of the RCSPP specific preprocessing and probing techniques discussed in Section 5.2.

In the first set of preprocessing experiments, we utilized the scheme of Dumitrescu and Boland [35] for problem classes A1-A3 and C1. After removing any arcs in $\delta^-(s) \cup \delta^+(t)$, and before preprocessing with CPLEX, we repeatedly called the function DB_Preprocessing from Algorithm 10 in Section 5.2 until no other reductions to G were possible. The function was initially called with no incumbent, and if an incumbent P_I was found with value $U = c(P_I)$, then the upper cutoff value for the branch-and-cut algorithm was updated, i.e., `CPX_PARAM_CUTUP` was set to U. CPLEX assures that any node in the branch-and-bound tree with an LP relaxation value above this cutoff is fathomed. The instances in the remaining classes, C2 and C3, contain negative cost cycles. Therefore, we used the weaker preprocessing scheme of Aneja et al. [2] instead and repeatedly called the function AAN_Preprocessing from Algorithm 9 in Section 5.2 until no other reductions to G were possible. Note that the shortest weight paths with respect to each resource were found in polynomial time since the arc weights are non-negative for all of the problem classes.

In addition to the initial RCSPP preprocessing from [35] and [2], the second set of experiments also included one of the acyclic preprocessing enhancements introduced in Section 5.2.2. Because the resource constraints are very dense for all of the instances, and the arc weights are very similar in magnitude, we did not explore the first two acyclic enhancements (identification of redundant constraints and matrix coefficient reduction).

However, we did use preprocessing on $G[Q]$, the subgraph induced by all s-t paths in G containing a given s-t subpath Q, to remove infeasible nodes and arcs. After the initial RCSPP preprocessing, but before CPLEX' presolver and aggregator, we preprocessed $G[(i)]$ for each node i. If no feasible s-t path existed in $G[(i)]$ after it was preprocessed, then i was removed from G. Preprocessing $G[(i,j)]$ for each arc (i,j) is more expensive, hence, we only considered a subset of the arcs after branch-and-cut had begun. More precisely, we preprocessed $G[(i,j)]$ for any arc (i,j) such that $x^*_{ij} > 0$ in some fractional solution x^* to the LP relaxations at the root node or either of its two child nodes in the branch-and-bound tree. If no feasible s-t path existed in $G[(i,j)]$ after it was preprocessed, then x_{ij} was fixed to zero.

To define any $G[Q]$ for classes C1-C3, we used the cyclic approximation discussed in Section 4.2. All $G[Q]$ were preprocessed by repeatedly calling AAN_Preprocessing until no further reductions were possible. If an incumbent solution was found in the initial RCSPP preprocessing for classes A1-A3 and C1, we used the extra optimality constraint (5.11) to strengthen the preprocessing of each $G[Q]$.

The full results of both sets of experiments can be found in Appendix A in Table 22. The DB and AAN settings refer to initial RCSPP preprocessing with DB_Preprocessing and AAN_Preprocessing, respectively. If +GQ is appended to the setting, then the results correspond to our second set of experiments which also incorporated the acyclic preprocessing enhancement. For these results, we introduce two new performance metrics: the number of positive arcs in fractional LP solutions that were fixed to zero during branch-and-cut using preprocessing on some $G[(i,j)]$ (Fixes), and the time spent performing non-CPLEX preprocessing (t_P).

We summarize these results in Table 8. As in [35], we see that the scheme of Dumitrescu and Boland solved all of the instances in class C1 and all but two of the instances in class A1 at the root node. When the scheme was combined with the acyclic preprocessing enhancement, the two remaining instances in class A1 were also solved at the root node. Likewise, this combination was an improvement for classes A2 and A3. In both classes, the integrality gaps at the root node were at least as good as the ones obtained using

Table 8: Summary of the RCSPP preprocessing and probing results

Class	Setting	Avg Rows	Avg Cols	Avg Nzs	Avg Fixes	Avg Cuts	Avg Root Gap%	Avg Final Gap% (Δz^*)	Avg z_{IP} Node	Avg B&B Nodes (#rt)	Avg t_P	Avg t_C	Avg t (#to)
A1	DB	24	192	2,008	0	0	9.1	0.0 (0)	10	17 (10)	0.0	0.0	0.21 (0)
	DB+GQ	4	13	122	4	0	0.0	0.0 (0)	0	0 (12)	0.0	0.0	0.02 (0)
A2	DB	253	2,409	15,927	0	0	16.8	0.0 (0)	79	186 (5)	0.0	0.0	6.75 (0)
	DB+GQ	224	2,089	12,462	156	0	4.3	0.0 (0)	26	73 (7)	0.6	0.0	4.94 (0)
A3	DB	1,071	10,328	83,902	0	0	15.7	0.0 (0)	1,378	2,726 (0)	0.1	0.0	330.05 (0)
	DB+GQ	1,064	10,257	83,213	291	0	14.0	0.0 (0)	1,309	2,553 (1)	5.5	0.0	296.74 (0)
C1	DB	3	5	40	0	0	0.0	0.0 (0)	0	0 (12)	0.0	0.0	0.00 (0)
C2	AAN	135	1,131	4,639	0	159	5.7	0.0 (0)	1,896	2,377 (4)	0.0	0.2	6.06 (0)
	AAN+GQ	135	1,130	4,626	1	159	5.7	0.0 (0)	1,897	2,378 (4)	0.2	0.2	6.37 (0)
C3	AAN	757	7,517	61,173	0	42	12.7	0.3 (0)	5,042	10,188 (0)	0.0	3.9	519.96 (2)
	AAN+GQ	757	7,517	61,173	0	42	12.7	0.3 (0)	5,042	10,157 (0)	8.0	3.9	528.67 (2)

default CPLEX with the cuts on. The scheme of Aneja et al. decreased the integrality gaps at the root node for class C2 and enabled all of the instances in class C3 to find the actual optimal solution (i.e., $\Delta z^* = 0$) without increasing the number of cycle elimination inequalities. However, combining the cyclic approximation for the enhancement with the scheme of Aneja et al. did not provide any improvement for classes C2 and C3.

The best preprocessing and probing settings for each problem class led to smaller run times when compared to CPLEX with all cuts turned off. For all subsequent experiments in this chapter, we use the scheme of Dumitrescu and Boland combined with the acyclic preprocessing enhancement for classes A2 and A3, and we use only the scheme of Aneja et al. for classes C2 and C3. Since all of the instances in classes A1 and C1 were solved at the root node with preprocessing, they are excluded from the rest of the experiments.

6.5 Branching Scheme Experiments

By default, CPLEX branches on fractional arc variables. In this section, we investigate the alternate branching schemes proposed in Section 5.4. We begin with a comparison between enhanced arc branching and branching on fractional nodes in G.

In the enhanced arc branching experiments, the fractional arc (i, j) was always selected by CPLEX at any node n in the branch-and-bound tree. However, we used the branch callback functionality in the callable library to fix additional arcs in both branches to zero. Let G_n be the subgraph of G implied by the previous fixings at n. For the downward branch ($x_{ij} = 0$), we used the trivial function ST_CONNECTED from Algorithm 11 in Section 5.4 to fix all arcs which are incident to nodes that can no longer be found in any s-t path in $G_n \setminus \{(i, j)\}$. For the upward branch ($x_{ij} = 1$), we defined $G_n[(i, j)] = (V_n[(i, j)], A_n[(i, j)])$ (approximately in the cyclic case) and fixed all arcs in $A_n \setminus A_n[(i, j)]$. Since we made no estimates on the optimal value of either branch, we used the scheme suggested by Land and Powell [85] and asked CPLEX to follow the downward branch first if $x^*_{ij} < 0.5$ and the upward branch first otherwise.

We also used the branch callback functionality for the node branching experiments. To select a fractional node in the graph to branch upon, we considered four different strategies:

1. choose a fractional node i that minimizes $\min\{x^*(\delta^+(i)), 1 - x^*(\delta^+(i))\}$,

2. choose a fractional node i that maximizes $\min\{x^*(\delta^+(i)), 1 - x^*(\delta^+(i))\}$,

3. choose the fractional node corresponding to the flow balance constraint dual variable in the LP relaxation with the largest absolute value, and

4. choose the first fractional node in a topological ordering (acyclic G only).

Once the fractional node i was chosen, we defined the fixings for both of the branches. For the downward branch ($x_{kl} = 0$ for all $(k, l) \in \delta^-(i) \cup \delta^+(i)$), we called ST_CONNECTED on $G_n \setminus \{i\}$ to fix additional arcs to zero. For the upward branch ($x(\delta^+(i)) = 1$), we defined $G_n[(i)] = (V_n[(i)], A_n[(i)])$ (approximately in the cyclic case) and fixed all arcs in $A_n \setminus A_n[(i)]$ to zero. The constraint $x(\delta^+(i)) = 1$ was not added to the subproblem in the upward branch if the underlying graph was acyclic because $G_n[(i)]$ is not an approximation in this case and each s-t path in $G_n[(i)]$ must contain i. Then, we asked CPLEX to follow the downward branch first if $x^*(\delta^+(i)) < 0.5$ and the upward branch first otherwise.

The full results of these experiments can be found in Appendix A in Table 23. The ARC setting refers to enhanced arc branching, and the NDs setting for $s = 1, 2, 3, 4$ refers to node branching with fractional node selection strategy s in the list above. For these results, we introduce a new performance metric: the non-CPLEX time spent on branching (t_B).

These results are summarized in Table 9. We see that the formulation sizes and the integrality gaps at the root node have slightly increased from the best preprocessing and probing results in Table 8 for all of the problem classes. This is because we used the callable library function `CPXcopyprotected` to specify that any arcs remaining after the initial RCSPP preprocessing should not be substituted out of the problem by CPLEX' presolver and aggregator. This allows us to explicitly update bounds in the branch callbacks for all of the additional arcs which are fixed to zero. The majority of the branching schemes for classes A2 and A3 led to smaller run times when compared to the best preprocessing and probing results, but none of the branching schemes provided an improvement for classes C2 and C3.

Table 9: Summary of the RCSPP branching scheme results

Class	Setting	Avg Rows	Avg Cols	Avg Nzs	Avg Fixes	Avg Cuts	Avg Root Gap%	Avg Final Gap% (Δz^*)	Avg z_{IP} Node	Avg B&B Nodes (#rt)	Avg t_P	Avg t_C	Avg t_B	Avg t (#to)
A2	ARC	228	2,092	12,528	174	0	5.0	0.0 (0)	47	100 (7)	0.7	0.0	0.2	5.52 (0)
	ND1	228	2,092	12,528	220	0	5.0	0.0 (0)	76	176 (7)	0.6	0.0	0.3	4.92 (0)
	ND2	228	2,092	12,528	170	0	5.0	0.0 (0)	11	34 (7)	0.6	0.0	0.1	2.78 (0)
	ND3	228	2,092	12,528	253	0	5.0	0.0 (0)	61	77 (7)	0.6	0.0	0.1	2.94 (0)
	ND4	228	2,092	12,528	263	0	5.0	0.0 (0)	11	43 (7)	0.6	0.0	0.1	2.67 (0)
A3	ARC	1,065	10,258	83,260	292	0	14.0	0.0 (0)	2,045	3,711 (1)	6.3	0.0	13.3	176.10 (0)
	ND1	1,065	10,258	83,260	253	0	14.0	0.0 (0)	4,933	8,853 (1)	4.9	0.0	30.1	319.75 (0)
	ND2	1,065	10,258	83,260	291	0	14.0	0.0 (0)	541	1,438 (1)	5.5	0.0	5.0	117.15 (0)
	ND3	1,065	10,258	83,260	291	0	14.0	0.0 (0)	1,717	3,262 (1)	5.4	0.0	10.9	170.56 (0)
	ND4	1,065	10,258	83,260	283	0	14.0	0.0 (0)	1,198	2,454 (1)	5.4	0.0	8.2	143.57 (0)
C2	ARC	149	1,145	4,775	0	142	5.8	0.0 (0)	1,317	1,887 (4)	0.0	0.2	1.1	6.54 (0)
	ND1	149	1,145	4,775	0	120	5.8	0.0 (0)	1,716	2,802 (4)	0.0	0.2	1.9	13.34 (0)
	ND2	149	1,145	4,775	0	170	5.8	0.0 (0)	2,365	2,577 (4)	0.0	0.3	2.1	14.81 (0)
	ND3	149	1,145	4,775	0	133	5.8	0.0 (0)	1,685	2,419 (4)	0.0	0.2	1.7	10.61 (0)
C3	ARC	757	7,518	61,210	0	52	13.1	1.1 (10)	7,707	10,984 (0)	0.0	3.5	30.6	723.62 (3)
	ND1	757	7,518	61,210	0	64	13.9	3.3 (30)	20,105	37,114 (0)	0.0	13.5	127.2	1,401.75 (8)
	ND2	757	7,518	61,210	0	45	12.8	0.4 (2)	4,402	7,917 (0)	0.0	2.6	23.1	541.59 (2)
	ND3	757	7,518	61,210	0	46	13.1	1.0 (8)	6,321	11,349 (0)	0.0	4.2	38.9	710.90 (3)

In a second set of experiments, we improved the branching schemes by applying the preprocessing scheme of Aneja et al. to fix an even greater number of arcs to zero in the branching subproblems. For problem classes A2 and A3, we repeatedly called the function AAN_Preprocessing on the corresponding subgraphs for each branch until no other reductions were possible. If an incumbent solution had been found, we used the extra optimality constraint (5.11) to strengthen the preprocessing. Since the complexity required to find shortest weight paths is larger for cyclic graphs, and finding shortest cost paths is strongly $\mathcal{NP}$-hard with negative cost cycles, we used only a single call to AAN_Preprocessing for classes C2 and C3 with no optimality constraint. We chose to use fractional node selection strategy 2 for node branching in these experiments, i.e., we branch upon the fractional node with the maximum integer infeasibility, because it was the most effective for the larger problem classes.

The full results of this second set of experiments can be found in Appendix A in Table 24, and a summary is presented in Table 10. We see that the additional preprocessing with the scheme of Aneja et al. decreased the number of evaluated branch-and-bound nodes for all of the problem classes and branching schemes. When combined with additional preprocessing, enhanced arc branching led to the quickest run times for the smaller problem classes A2 and C2, and node branching performed best for the larger problem classes A3 and C3. In fact, only one of the instances in C3 timed out with node branching, and the actual optimal solution was obtained for all of the instances (i.e., $\Delta z^* = 0$).

Despite the increase in non-CPLEX time spent on branching, the best branching scheme settings for each problem class led to smaller run times when compared to the best preprocessing and probing results in Table 8. For all subsequent experiments in this chapter, we use enhanced arc branching with additional preprocessing for classes A2 and C2, and we use node branching with fractional node selection strategy 2 and additional preprocessing for classes A3 and C3.

Table 10: Summary of the RCSPP branching scheme results with preprocessing

Class	Setting	Avg Rows	Avg Cols	Avg Nzs	Avg Fixes	Avg Cuts	Avg Root Gap%	Avg Final Gap% (Δz^*)	Avg z_{IP} Node	Avg B&B Nodes (#rt)	Avg t_P	Avg t_C	Avg t_B	Avg t (#to)
A2	ARC+AAN	228	2,092	12,528	162	0	5.0	0.0 (0)	7	11 (7)	0.6	0.0	0.1	1.88 (0)
	ND2+AAN	228	2,092	12,528	171	0	5.0	0.0 (0)	7	15 (7)	0.6	0.0	0.1	1.95 (0)
A3	ARC+AAN	1,065	10,258	83,260	260	0	14.0	0.0 (0)	647	985 (1)	5.2	0.0	17.1	112.40 (0)
	ND2+AAN	1,065	10,258	83,260	289	0	14.0	0.0 (0)	392	764 (1)	5.5	0.0	7.2	89.76 (0)
C2	ARC+AAN	149	1,145	4,775	0	134	5.8	0.0 (0)	1,341	1,827 (4)	0.0	0.1	1.4	5.42 (0)
	ND2+AAN	149	1,145	4,775	0	173	5.8	0.0 (0)	2,005	2,520 (4)	0.0	0.2	2.0	12.72 (0)
C3	ARC+AAN	757	7,518	61,210	0	49	13.4	1.3 (15)	4,716	7,006 (0)	0.0	2.4	69.2	607.87 (3)
	ND2+AAN	757	7,518	61,210	0	37	12.7	0.2 (0)	2,842	6,759 (0)	0.0	2.1	37.7	490.34 (1)

6.6 Primal Heuristic Experiments

In preliminary experiments with the MIP-based heuristics proposed in Section 5.5.2 for the RCSPP, we were unable to find an approach that led to a decrease in run times. In fact, we found the opposite was often the case. The earlier we found the optimal solution in the searches, the larger the run times for these instances. In this section, we present computational results which illustrate this claim.

In our preliminary experiments, we also explored various combinations of CPLEX parameter settings that determine the order in which the branch-and-bound tree is traversed. The parameter `CPX_PARAM_MIPEMPHASIS` controls the balance between the two main goals in a branch-and-cut algorithm: (1) feasibility – the search for new incumbent solutions, and (2) optimality – the search for a proof that there does not exist an undiscovered feasible solution with a better objective value. The parameters `CPX_PARAM_BTTOL`, `CPX_PARAM_NODESEL` and `CPX_PARAM_DIVETYPE` control the balance between diving and backtracking. We also implemented several alternative strategies for selecting the next node to be evaluated in the branch-and-bound tree during a backtrack using the node callback functionality in the callable library. The best results were achieved with nothing more than emphasizing feasibility over optimality, i.e., setting `CPX_PARAM_MIPEMPHASIS` to `CPX_MIPEMPHASIS_FEASIBILITY`. We used this parameter setting for all of the experiments presented in this section. Moreover, we turned off CPLEX' relaxation induced neighborhood search (RINS) heuristic so as not to bias the results.

Using the heuristic callback functionality in the callable library, we implemented a crude hybrid neighborhood search heuristic for the RCSPP. This approach combines elements of the RINS heuristic of Danna et al. [24] and the local branching heuristic of Fischetti and Lodi [43]. At any node in the branch-and-bound tree, we used the incumbent solution $\bar{x}$ (if it exists) and the fractional solution x^* to the current LP relaxation to define $x^H \in \mathbb{R}^A$ as follows

$$x^H = \begin{cases} \bar{x} + x^*, & \text{if } \bar{x} \text{ exists} \\ x^*, & \text{otherwise.} \end{cases}$$

Then, we constructed an auxiliary RCSPP by adding the soft fixing constraint

$$\sum_{(i,j)\in A: x_{ij}^H = 0} x_{ij} \leq \Delta$$

to the original formulation. This inequality stipulates that at most Δ arcs which are not found in either of the support graphs for $\bar{x}$ and x^* can be included in the new s-t path, and it is not difficult to show that the neighborhood defined by this inequality contains any solution in the corresponding RINS and local branching neighborhoods when $\bar{x}$ exists. The auxiliary problem was preprocessed by repeatedly calling the function AAN_Preprocessing until no other reductions were possible. If an incumbent solution $\bar{x}$ was available for problem classes A2 and A3, we used the extra optimality constraint (5.11) with $U = c\bar{x}$ to strengthen this preprocessing. To solve the auxiliary problem, we simply enumerated all feasible s-t paths and limited the computational burden by setting $\Delta = 1$. Since this heuristic does not require an incumbent solution, we ran it for every node in the branch-and-bound tree.

For a comparison, we solved all of the instances with and without the hybrid heuristic. The full results of these experiments can be found in Appendix A in Table 25. The HYBRID setting refers to the use of our heuristic, and the NORINS setting refers to the use of no MIP-based heuristics. For these results, we introduce our final performance metric: the time spent on non-CPLEX heuristics (t_H).

The results are summarized in Table 11. We see that the hybrid heuristic decreased the number of branch-and-bound nodes evaluated before the actual optimal values were found (i.e., z_{IP} Node) for all of the problem classes. However, this led to longer run times for all of the classes and three additional time outs for class C3. Furthermore, finding solutions earlier in the search actually led to an increase in the number of branch-and-bound nodes evaluated for classes A2 and A3. Class C3 experienced the most significant decrease in the number of evaluated nodes, but this statistic was skewed by the additional instances that timed out.

Despite the fact that auxiliary problems were solved at every node in the branch-and-bound tree with explicit enumeration, the time spent on the hybrid heuristic was a relatively small fraction of the overall run time for all of the problem classes except C2. Therefore,

Table 11: Summary of the RCSPP primal heuristic results

Class	Setting	Avg Rows	Avg Cols	Avg Nzs	Avg Fixes	Avg Cuts	Avg Root Gap%	Avg Final Gap% (Δz^*)	Avg z_{IP} Node	Avg B&B Nodes (#rt)	Avg t_P	Avg t_C	Avg t_H	Avg t_B	Avg t (#to)
A2	HYBRID	228	2,092	12,528	154	0	4.9	0.0 (0)	5	13 (6)	0.6	0.0	0.0	0.1	1.32 (0)
	NORINS	228	2,092	12,528	151	0	4.9	0.0 (0)	10	12 (4)	0.5	0.0	0.0	0.1	1.10 (0)
A3	HYBRID	1,065	10,258	83,260	140	0	15.1	0.0 (0)	81	1,110 (0)	5.2	0.0	4.4	11.3	110.99 (0)
	NORINS	1,065	10,258	83,260	44	0	15.1	0.0 (0)	182	599 (0)	5.5	0.0	0.0	6.3	46.94 (0)
C2	HYBRID	149	1,145	4,775	0	111	5.8	0.0 (0)	382	2,020 (4)	0.0	0.1	5.1	1.5	9.75 (0)
	NORINS	149	1,145	4,775	0	127	5.8	0.0 (0)	1,659	2,319 (4)	0.0	0.2	0.0	2.2	6.29 (0)
C3	HYBRID	757	7,518	61,210	0	28	12.7	0.6 (0)	405	5,720 (0)	0.0	1.7	11.4	41.9	719.30 (4)
	NORINS	757	7,518	61,210	0	34	12.7	0.2 (0)	2,418	6,608 (0)	0.0	2.1	0.0	39.6	420.96 (1)

it is not likely that a faster implementation of the heuristic would enable us to solve the instances more quickly than using no MIP-based heuristic at all. Even if we used the improved efficiency to move to larger values of Δ, it seems as though finding solutions earlier in the search delays the proof of optimality. This behavior might be caused by reduction schemes, such as reduced cost fixing, which are used internally by CPLEX to fix additional variables to one of their bounds when an upper bound on the optimal value is available. Better bounds lead to more fixings, and this impacts the LP relaxation solutions, the branching variable(s) selections, the overall branching strategy, and the convergence of the best lower bound in the branch-and-bound tree. However, this hypothesis cannot be tested because there are no specific parameter settings for reduction schemes such as reduced cost fixing in CPLEX 9.0.

Since our primary performance metric is running time, we do not use the hybrid heuristic for the remaining experiments in this chapter, but we do set the emphasis parameter to feasibility and turn off CPLEX' RINS heuristic. These settings led to smaller run times for all of the problem classes except C2 when compared to the best branching scheme results in Table 10, and the improvement was most dramatic for the hardest classes. Note that if our primary goal was to find good feasible solutions in a limited amount of time without solving the instances to optimality (e.g., column generation pricing problems), it could be advantageous to investigate more efficient versions of the MIP-based heuristics in Section 5.5.2 that do not solve the auxiliary problems to optimality. This crude implementation found all of the optimal solutions for class C3 approximately four times faster than the best branching scheme results which had CPLEX' RINS heuristic turned on.

6.7 Cutting Plane Experiments

We have already established that the cuts generated by CPLEX for general MIPs were not very effective for these particular RCSPP instances. Up until this point, only a minimal set of valid inequalities have been considered for problem classes C2 and C3 which guarantee elementary s-t paths in the presence of negative cost cycles. In this section, we incorporate more of the valid inequality classes for the RCSPP outlined in Section 5.3 and evaluate

their performance.

A number of choices must be made about when to generate cutting planes and how to manage them when implementing a branch-and-cut algorithm. In preliminary experiments with the RCSPP inequality classes, many violated inequalities were found at each node in the branch-and-bound tree, and this led to much larger LP relaxations and longer overall running times. When faced with a similar situation, several researchers have successfully used deletion to limit the size of the LP relaxations by frequently removing all cutting planes that have been inactive for recent LP solutions. Even though the same cut may be generated several times in this approach, it is likely to work well with the RCSPP inequalities because separation is relatively inexpensive. However, CPLEX 9.0 does not allow the user to delete a cutting plane after it has been added with the cut callback functionality in the callable library. Thus, with the exception of the cycle elimination inequalities mentioned above for classes C2 and C3, cutting planes were only generated at the root node for all of the experiments presented in this section. Moreover, we used tolerances and probabilities during separation to limit the number of generated cuts.

Let $x^* \in [0,1]^A$ be a (possibly integral) solution to the LP relaxation at the root node. If the corresponding support graph $G^* = (V^*, A^*)$ contains disjoint subgraphs, and there exists subtour elimination inequalities (5.8) violated by these subgraphs (see Section 6.3), then we add all of the inequalities to the LP relaxation, reoptimize, and repeat our search for cuts with the new solution. If no violated inequalities (5.8) are found and x^* is fractional (integral x^* is optimal), then we solve the separation problem for each of the remaining RCSPP inequality classes in one *round* of cut generation. If violated inequalities are found during the current round of cut generation, then we add all of the inequalities to the LP relaxation, reoptimize, and repeat our search for cuts with the new solution. Otherwise, we branch on a fractional node or arc in G^*. Typically, the change in the value of the LP relaxation decreases with each round of cut generation. Therefore, we branch prematurely to avoid a tailing off effect if the change in value is less than 10^{-3} between successive rounds.

We now present the separation details for all of the RCSPP inequality classes considered in each round of cut generation, most of which are jointly defined versions of the original

inequalities introduced for the weight constrained shortest path problem. For more details on the jointly defined versions, which are derived using multiple resources, refer to Section 5.3.3. Note that if an incumbent solution was found at the root node for problem classes A2 and A3, we used the optimality constraint (5.11) as an extra resource for the RCSPP inequalities.

Node Precedence Inequalities Jointly defined versions of the node precedence inequality (3.8), were separated using Algorithm 2 in Section 3.2. The maximum violation for any of the precedence inequalities (node, subpath and s-t cut) is bounded by the value of the left hand side, and a small value of x^*_{ij} implies a small likelihood of violation. Thus, we skipped any arc $(i,j) \in A^*$ such that $x^*_{ij} < 0.25$ in the outermost loop of the algorithm to limit the number of generated cuts. We also used a similar scheme to separate jointly defined versions of the reverse node precedence inequality (3.9).

Subpath Precedence Inequalities Jointly defined versions of the subpath precedence inequality (3.12) were separated using Algorithm 3 in Section 3.3. To limit the number of generated cuts and reduce the computational burden of enumerating over all fractional s-t subpaths, we skipped any subpath $Q = (i_1, i_2, \ldots, i_k)$ such that $x^*_{i_1,i_2} < 0.75$ since they are less likely to result in a violation. We also used a similar scheme to separate jointly defined versions of the reverse subpath precedence inequality (3.13).

s-t Cut Precedence Inequalities Jointly defined versions of the s-t cut precedence inequality (3.15) were separated using Algorithm 4 in Section 3.4. We used CPLEX' network optimizer to solve each minimum s-t cut problem as an LP (refer to Section 1.2.3). To limit the number of generated cuts and reduce the number of minimum s-t cut problems solved, we skipped any arc $(i,j) \in A^*$ such that $x^*_{ij} < 0.75$ in the outermost loop of the algorithm since they are less likely to result in a violation. We also used a similar scheme to separate jointly defined versions of the reverse s-t cut precedence inequality (3.16).

Clique Inequalities A jointly defined version of the conflict graph described in Section 3.6.2 was used to separate the clique inequalities (3.19). A fractional conflict graph $G_C^* = (V_C^*, E_C^*)$ was constructed for each solution x^* to the LP relaxation, where each node in V_C^* corresponded to an arc (i, j) in the original graph such that $0 < x_{ij}^* < 1$. The simple greedy heuristic outlined in Section 3.6.1 was used to find maximum weight cliques on G_C^*. To limit the number of generated cuts and reduce the effort of generating a clique for each $(i, j) \in V_C^*$, we randomly chose only 25% of the arcs in the set $\{(i, j) \in V_C^* : x_{ij}^* \geq 0.25\}$ to initialize the greedy cliques.

Lifted Odd Hole Inequalities The same fractional conflict graph that was used to separate the clique inequalities (3.19) was also used to separate the lifted odd hole inequalities (3.21). We implemented the approach of Hoffman and Padberg [66] and used the shortest paths in an auxiliary layered graph to find maximum weight odd holes on G_C^*. Each time we found an odd hole H, we lifted the remaining $(i, j) \in V_C^* \setminus H$ into the corresponding odd hole inequality (3.20) in random order. The lifting problems were approximated with an integer version of the dual to the node packing problem and solved using a greedy algorithm. To limit the number of generated cuts and reduce the effort of finding a maximum weight odd hole for each $(i, j) \in V_C^*$, we randomly chose only 25% of the arcs in the set $\{(i, j) \in V_C^* : x_{ij}^* \geq 0.25\}$ to initialize the layered graphs. Furthermore, we abandoned our lifting procedure and searched for a new odd hole if a violation had not been found after solving for the first 10 lifting coefficients.

Lifted Path Subset Cover Inequalities The original (not jointly defined) lifted path subset cover inequalities (4.2) were separated with respect to each individual resource. Given any resource constraint, we found a minimal path subset cover E, which was not necessarily violated, using Algorithm 5 in Section 4.3. Then, only the arcs in

$$F = \left\{(i, j) \in A \setminus E : x^*(\delta^+(i)) > 0\right\}$$

were lifted into the corresponding path subset cover inequality (4.1) in topological order using Algorithm 8 in Section 4.4. Since only arcs in $E \cup F$ can contribute to the violation

of inequality (4.2), we abandoned our lifting procedure and moved on to the next resource if a violation had not been found after all of the arcs in F had been lifted. Otherwise, we lifted the remaining $(i,j) \in A \setminus (E \cup F)$ with a second pass of the topological lifting routine. Inequalities (4.2) were only generated for problem classes A2 and A3 because the underlying graph for the RCSPP must be acyclic for topological lifting.

Generalized Subtour Elimination Inequalities Even though the GUB inequalities (5.6) and the subtour elimination inequalities (5.8) are enough to forbid negative cost cycles for problem classes C2 and C3, we also separated the stronger generalized subtour elimination inequalities (5.10) for fractional solutions to the LP relaxation at the root node. For each node $i \in V^* \setminus \{s,t\}$, we found a maximally violated inequality (5.10) by solving a minimum i-t cut problem on G^* with the arc capacities given by x^*. Each minimum i-t cut problem was solved as an LP using CPLEX' network optimizer (refer to Section 1.2.3). Out of all of our cycle elimination inequalities, (5.10) is the most expensive to separate (i.e., $\mathcal{O}(|V^*|^4)$ with the most efficient algorithm), and it typically increases the tailing off effect for the change in value of the LP relaxation. Thus, we skipped any node $i \in V^* \setminus \{s,t\}$ such that $x^*(\delta^+(i)) < 0.5$ to limit the number of generated cuts and reduce the number of LPs solved.

In the first set of cutting plane experiments, we generated all of the RCSPP inequality classes discussed above. Yet, adding each inequality class may not always be advantageous because separation takes time and is not always successful. Therefore, we also turned off each inequality class one at a time to see if their exclusion led to improved performance. The full results of these experiments can be found in Appendix A in Table 26. The ALL setting refers to the use of all inequalities. If -c is appended to the setting for c = NDP (node precedence inequalities), SPP (subpath precedence inequalities), CTP (s-t cut precedence inequalities), CLQ (clique inequalities), LOH (lifted odd hole inequalities), LPS (lifted path subset cover inequalities), GSE (generalized subtour elimination inequalities), then inequality class c was turned off for those results. Recall that most of the RCSPP inequalities

make use of the shortest weight paths with respect to each resource vector. Thus, the time spent calculating $\sigma_{ij}^{w^r}$ for all $i \in V \setminus \{s\}$, $j \in V \setminus \{t\}$ and $r = 1, \ldots, R$ was reported in the separation time for non-CPLEX cuts (t_C). However, $\sigma_{si}^{w^r}$ and $\sigma_{it}^{w^r}$ are provided by the initial preprocessing routines for all $i \in V$ and $r = 1, \ldots, R$.

These results are summarized in Table 12. We see that the integrality gaps at the root node have decreased for all of the problem classes when compared to the best primal heuristic results in Table 11. Also, the number of instances solved at the root node have increased for problem classes A2 and C2. Additional improvement is possible for both performance metrics, at the expense of larger LP relaxations, if a less conservative cut generation approach is used. For example, by separating all of the inequality classes without any tolerances or probabilities to limit the number of generated cuts, we were able to reduce the average integrality gap at the root node to 10% for problem class A3 in our preliminary experiments, with two of the instances solving at the root node. However, the average number of non-CPLEX cuts generated for each instance increased to 3,419 (more than three times the number of original constraints) which led to much longer run times.

It is difficult to draw definite conclusions about which individual inequality classes should be turned off. The clique inequalities (3.19) seemed to be the most critical inequality class because turning them off led to longer run times for all of the problem classes. Conversely, turning off the lifted odd hole inequalities (3.21) led to improved run times for all but one of the problem classes. Note that ALL-NDP was not presented for problem class A3 because there were no violated node precedence inequalities (3.8) and (3.9) found when all of the inequalities were generated.

Generating all of the RCSPP inequalities led to smaller run times when compared to the best primal heuristic results for all of the problem classes except A2. Since the preprocessing time was a relatively large fraction of the overall run time for A2, we turned off part of the acyclic preprocessing enhancement for problem classes A2 and A3 in a second set of experiments. More precisely, we did not fix any positive arcs in the fractional LP solutions to zero using preprocessing on some $G[(i, j)]$.

The full results of this second set of experiments can be found in Appendix A in Table 27,

Table 12: Summary of the RCSPP cutting plane results

Class	Setting	Avg Rows	Avg Cols	Avg Nzs	Avg Fixes	Avg Cuts	Avg Root Gap%	Avg Final Gap% (Δz^*)	Avg z_{IP} Node	Avg B&B Nodes (#rt)	Avg t_P	Avg t_C	Avg t_B	Avg t (#to)
A2	ALL	228	2,092	12,528	214	9	2.2	0.0 (0)	10	12 (8)	0.5	0.1	0.1	1.27 (0)
	ALL-NDP	228	2,092	12,528	214	8	2.2	0.0 (0)	10	12 (8)	0.4	0.1	0.1	1.28 (0)
	ALL-SPP	228	2,092	12,528	220	6	2.4	0.0 (0)	10	12 (7)	0.5	0.1	0.1	1.29 (0)
	ALL-CTP	228	2,092	12,528	217	8	2.2	0.0 (0)	10	12 (8)	0.5	0.1	0.1	1.27 (0)
	ALL-CLQ	228	2,092	12,528	216	8	2.2	0.0 (0)	7	12 (8)	0.5	0.1	0.1	1.38 (0)
	ALL-LOH	228	2,092	12,528	218	7	2.2	0.0 (0)	7	9 (8)	0.5	0.1	0.1	1.20 (0)
	ALL-LPS	228	2,092	12,528	303	6	2.6	0.0 (0)	11	12 (7)	0.5	0.1	0.1	1.32 (0)
A3	ALL	1,065	10,258	83,260	64	17	14.6	0.0 (0)	244	637 (0)	3.5	2.8	5.6	44.18 (0)
	ALL-SPP	1,065	10,258	83,260	48	11	14.9	0.0 (0)	130	576 (0)	3.4	2.7	5.1	41.85 (0)
	ALL-CTP	1,065	10,258	83,260	63	16	14.6	0.0 (0)	245	638 (0)	3.5	2.8	5.6	44.34 (0)
	ALL-CLQ	1,065	10,258	83,260	76	17	14.6	0.0 (0)	233	654 (0)	3.6	2.8	5.8	47.00 (0)
	ALL-LOH	1,065	10,258	83,260	76	5	14.6	0.0 (0)	183	610 (0)	3.6	2.8	5.4	43.84 (0)
	ALL-LPS	1,065	10,258	83,260	58	16	14.6	0.0 (0)	252	647 (0)	3.5	2.3	5.7	44.24 (0)
C2	ALL	149	1,145	4,775	0	221	2.1	0.0 (0)	295	442 (6)	0.0	0.7	0.7	3.79 (0)
	ALL-NDP	149	1,145	4,775	0	218	2.1	0.0 (0)	277	413 (6)	0.0	0.7	0.6	3.36 (0)
	ALL-SPP	149	1,145	4,775	0	193	2.2	0.0 (0)	319	464 (6)	0.0	0.7	0.7	3.77 (0)
	ALL-CTP	149	1,145	4,775	0	561	2.1	0.0 (0)	531	657 (6)	0.0	1.0	1.1	15.60 (0)
	ALL-CLQ	149	1,145	4,775	0	191	2.2	0.0 (0)	280	430 (6)	0.0	1.6	0.7	4.76 (0)
	ALL-LOH	149	1,145	4,775	0	192	2.1	0.0 (0)	347	462 (6)	0.0	0.8	0.7	3.83 (0)
	ALL-GSE	149	1,145	4,775	0	185	2.1	0.0 (0)	295	387 (6)	0.0	0.6	0.5	3.01 (0)
C3	ALL	757	7,518	61,210	0	67	11.5	0.2 (0)	2,556	6,313 (0)	0.0	5.3	34.5	361.85 (1)
	ALL-NDP	757	7,518	61,210	0	67	11.6	0.3 (0)	2,872	6,462 (0)	0.0	5.5	40.7	433.88 (2)
	ALL-SPP	757	7,518	61,210	0	47	12.0	0.3 (2)	2,574	6,483 (0)	0.0	5.3	40.4	428.44 (2)
	ALL-CTP	757	7,518	61,210	0	61	11.5	0.2 (0)	2,163	6,048 (0)	0.0	5.3	35.8	379.19 (1)
	ALL-CLQ	757	7,518	61,210	0	75	11.5	0.2 (0)	2,172	5,877 (0)	0.0	6.4	40.8	443.86 (1)
	ALL-LOH	757	7,518	61,210	0	65	11.5	0.2 (0)	1,895	5,819 (0)	0.0	5.3	33.4	355.27 (1)
	ALL-GSE	757	7,518	61,210	0	54	11.7	0.2 (0)	2,231	6,334 (0)	0.0	5.2	35.7	384.50 (1)

and a summary is presented in Table 13. We see that turning off the acyclic preprocessing enhancement increased the integrality gaps at the root node and the number of evaluated nodes in the branch-and-bound tree for both of the problem classes and all of the settings. Furthermore, the number of instances solved at the root node decreased for class A2. Despite these negative trends, we see improvement in our primary performance metric for both of the problem classes and all of the settings. Moreover, the combination of all of the RCSPP inequalities with no acyclic preprocessing enhancement led to smaller run times for class A2 when compared to the best primal heuristic results in Table 11.

While these very conservative cut generation settings are by no means the best way to use these inequalities, it does indicate their potential. Generating all of the inequality classes led to improved performance for each problem class when compared to the best primal heuristic results, and each inequality class contributed to a decline in performance for at least one problem class when excluded. Of course, additional experiments which evaluate more aggressive implementation choices might lead to better performance, but we believe that a platform which allows for more sophisticated cut management schemes is critical for their success. The ability to remove cuts after they have been added is crucial if inequalities are to be generated throughout the branch-and-bound tree. We expect that removing cuts will also allow the locally valid versions of the RCSPP inequalities mentioned in Section 5.3.4 to be successful. After all, these inequalities enhance the preprocessing done at each node in the branch-and-bound tree by our branching schemes, and many of the shortest weight path values necessary to separate these inequalities will already be provided by the preprocessing.

6.8 Summary

In this chapter we have investigated various implementation choices for each component in our branch-and-cut algorithm for the RCSPP. The best results from each section of experiments are presented in Table 14. For each problem class and setting, we report the average gap between the value of the best integer solution found and the value of the best lower bound in the branch-and-bound tree (Avg Final Gap%), the total difference between

Table 13: Summary of the RCSPP cutting plane results without arc fixing

Class	Setting	Avg Rows	Avg Cols	Avg Nzs	Avg Cuts	Avg Root Gap%	Avg Final Gap% (Δz^*)	Avg z_{IP} Node	Avg B&B Nodes (#rt)	Avg t_P	Avg t_C	Avg t_B	Avg t (#to)
A2	NFX+ALL	228	2,092	12,528	39	7.4	0.0 (0)	9	12 (4)	0.2	0.3	0.1	0.88 (0)
	NFX+ALL-NDP	228	2,092	12,528	38	7.4	0.0 (0)	9	12 (4)	0.2	0.3	0.1	0.88 (0)
	NFX+ALL-SPP	228	2,092	12,528	47	8.6	0.0 (0)	19	22 (5)	0.2	0.5	0.2	1.40 (0)
	NFX+ALL-CTP	228	2,092	12,528	35	7.4	0.0 (0)	11	15 (5)	0.2	0.3	0.1	0.90 (0)
	NFX+ALL-CLQ	228	2,092	12,528	35	7.5	0.0 (0)	14	17 (5)	0.2	0.3	0.2	1.22 (0)
	NFX+ALL-LOH	228	2,092	12,528	31	7.5	0.0 (0)	7	12 (5)	0.2	0.3	0.1	0.92 (0)
	NFX+ALL-LPS	228	2,092	12,528	39	7.9	0.0 (0)	9	13 (5)	0.2	0.1	0.1	0.73 (0)
A3	NFX+ALL	1,065	10,258	83,260	23	15.2	0.0 (0)	251	643 (0)	0.9	3.1	5.7	43.03 (0)
	NFX+ALL-SPP	1,065	10,258	83,260	18	15.5	0.0 (0)	136	582 (0)	0.9	3.0	5.2	39.68 (0)
	NFX+ALL-CTP	1,065	10,258	83,260	19	15.2	0.0 (0)	245	639 (0)	0.9	2.9	5.6	41.16 (0)
	NFX+ALL-CLQ	1,065	10,258	83,260	18	15.2	0.0 (0)	248	667 (0)	0.9	2.8	6.0	46.72 (0)
	NFX+ALL-LOH	1,065	10,258	83,260	6	15.2	0.0 (0)	193	621 (0)	0.9	2.9	5.5	41.09 (0)
	NFX+ALL-LPS	1,065	10,258	83,260	17	15.2	0.0 (0)	254	651 (0)	0.9	2.3	5.8	43.85 (0)

the true optimal value and the value of the best integer solution found (Δz^*), the average number of nodes evaluated in the branch-and-bound tree before the best integer solution was found (Avg z_{IP} Node), the total number of instances that were solved at the root node (#rt), the total effective running time of the branch-and-cut algorithm over all instances (Total t), the total number of instances which timed out (#to), and the performance ratio which is computed by dividing the total running time for default CPLEX by the total running time for that particular setting.

Table 14: Final summary of the RCSPP results

Class	Setting	Avg Final Gap% (Δz^*)	Avg z_{IP} Node (#rt)	Total t (#to)	Performance Ratio
A1	CPX	0.0 (0)	26 (9)	6.90 (0)	—
	CPX-C	0.0 (0)	22 (7)	4.35 (0)	1.6X
	DB+GQ	0.0 (0)	0 (12)	0.22 (0)	31.4X
A2	CPX	0.0 (0)	139 (6)	122.66 (0)	—
	CPX-C	0.0 (0)	60 (6)	73.18 (0)	1.7X
	DB+GQ	0.0 (0)	26 (7)	59.25 (0)	2.1X
	ARC+AAN	0.0 (0)	7 (7)	22.56 (0)	5.4X
	NORINS	0.0 (0)	10 (4)	13.21 (0)	9.3X
	NFX+ALL-LPS	0.0 (0)	9 (5)	8.80 (0)	13.9X
A3	CPX	0.0 (0)	1,871 (0)	21,455.46 (0)	—
	CPX-C	0.0 (0)	1,355 (0)	13,528.81 (0)	1.6X
	DB+GQ	0.0 (0)	1,309 (1)	10,682.63 (0)	2.0X
	ND2+AAN	0.0 (0)	392 (1)	3,231.47 (0)	6.6X
	NORINS	0.0 (0)	182 (0)	1,689.80 (0)	12.7X
	NFX+ALL-SPP	0.0 (0)	136 (0)	1,428.58 (0)	15.0X
C1	CPX	0.0 (0)	0 (10)	1.52 (0)	—
	CPX-C	0.0 (0)	2 (4)	1.79 (0)	0.9X
	DB	0.0 (0)	0 (12)	0.04 (0)	38.0X
C2	CPX	0.0 (0)	1,997 (2)	99.12 (0)	—
	CPX-C	0.0 (0)	1,882 (2)	130.22 (0)	0.8X
	AAN	0.0 (0)	1,896 (4)	72.67 (0)	1.4X
	ARC+AAN	0.0 (0)	1,341 (4)	65.01 (0)	1.5X
	NORINS	0.0 (0)	1,659 (4)	75.49 (0)	1.3X
	ALL-GSE	0.0 (0)	295 (6)	36.13 (0)	2.7X
C3	CPX	0.4 (2)	6,272 (1)	25,090.02 (2)	—
	CPX-C	0.4 (2)	4,749 (0)	20,839.87 (2)	1.2X
	AAN	0.3 (0)	5,042 (0)	18,718.56 (2)	1.3X
	ND2+AAN	0.2 (0)	2,842 (0)	17,652.40 (1)	1.4X
	NORINS	0.2 (0)	2,418 (0)	15,154.63 (1)	1.7X
	ALL-LOH	0.2 (0)	1,895 (0)	12,789.62 (1)	2.0X

We see positive trends for most of the performance metrics as each combination of implementation choices is fixed. Even though our goal was not to discover the "best" combination of implementation choices, we did outline a successful branch-and-cut implementation which significantly outperforms default CPLEX for all of the problem classes.

The only component which hindered the algorithm's performance with respect to running time was the MIP-based primal heuristic for the RCSPP; therefore, we turned it off along with CPLEX' RINS heuristic. However, the RCSPP heuristic did drastically decrease the number of branch-and-bound nodes evaluated before the actual optimal values were found for all of the problem classes, and this might be favorable if good feasible solutions are required in a limited amount of time.

CHAPTER 7

APPLICATION: THE DIAL-A-FLIGHT PROBLEM

7.1 Introduction

Recently, the DayJet Corporation (www.dayjet.com) began providing *per-seat*, *on-demand* air transportation services in the southeast region of the United States. Using a fleet of relatively cheap small jet aircraft with no fixed schedule, DayJet services regional travelers who request transportation one day or a few days in advance. In addition to reserving an entire aircraft (per-plane on-demand), each traveler is allowed to reserve individual seats (per-seat on-demand) with the potential for much lower fares.

To effectively manage day-to-day operations, DayJet employs two essential scheduling components: (1) an online accept/reject system to quickly inform passengers whether their transportation requests can be serviced and at what price, and (2) an offline scheduling system to construct minimum cost pilot and jet itineraries for the next day of operations once the reservation deadline has passed. The online accept/reject system is proprietary and involves a very rapid (less than 15 seconds) heuristic search. The implementation of an offline scheduling system, developed for and in collaboration with DayJet, is discussed by Espinoza et al. [39, 40]. In [39], the authors formulate the offline version of the problem as an integer multicommodity network flow model with side constraints. When solved with CPLEX [69], a leading integer programming software package, this formulation allows the solution of small instances (about 80 requests and no more than 8 aircraft). In [40], the core optimization technology is embedded in a parallel local search scheme which provides high quality solutions for very large practical instances (about 3,000 requests and over 300 aircraft). All of the neighborhoods employed in the parallel local search scheme involve subsets of jets and define smaller instances of the offline problem because passengers never switch aircraft. Thus, when the subsets are chosen small enough, the neighborhoods can be completely explored using the multicommodity network flow model.

In practice, DayJet allows the offline scheduling system to run for a fixed amount of time (typically four hours) after the reservation deadline has passed for the next day of operations. Therefore, the most straightforward way to improve the performance of the system developed by Espinoza et al. is to reduce the solution times for the multicommodity network flow model. If we can outperform CPLEX' branch-and-cut algorithm, then the parallel local search scheme can investigate a greater portion of the feasibility space in the same amount of time by exploring an increased number of neighborhoods or by exploring neighborhoods involving larger subsets of jets. The multicommodity network flow model consists of several resource constrained shortest paths (one for each jet) linked together by constraints which guarantee that all accepted traveler requests are satisfied. Therefore, we can apply extended versions of the branch-and-cut components for the resource constrained shortest path problem (RCSPP) described in Chapter 5.

In the remainder of this chapter, we focus our efforts on solving small instances of the offline problem more efficiently. We begin with a detailed description of the problem and present an abstract version of the multicommodity network flow model introduced in [39]. Then, we outline a straightforward extension of the branch-and-cut algorithm for the RCSPP which can be used to solve the problem. Finally, we investigate the algorithm's performance with practical instances provided by DayJet.

7.2 Problem Description and Formulation

In [39], Espinoza et al. define the *dial-a-flight problem* (DAFP), a new vehicle routing problem that arises in the context of on-demand air transportation and is concerned with the scheduling of a (static) set of requests for a single day of operations. Each request specifies an origin airport, a destination airport, an earliest acceptable departure time at the origin, a latest acceptable arrival time at the destination, and the number of passengers along with their total weight. A fleet of jet airplanes is available to provide the requested air transportation. Each jet has a home base (airport), a seating capacity limiting the number of passengers that can be accommodated, and a weight capacity limiting the weight that can be accommodated. Each jet is available for a certain period during the day, and has to

return to its home base at the end of the day. A set of pilots, stationed at the home bases of the airplanes, is available to fly the jets. A pilot departs from the home base where he is domiciled at the start of his duty and returns to the home base at the end of his duty. A pilot schedule has to satisfy FAA regulations governing flying hours and duty period, and pilots do not change aircraft during their duty. To ensure acceptable service, an itinerary for a passenger will involve at most two flights; i.e., at most one intermediate stop is allowed. Furthermore, if there is an intermediate stop, both flights have to be on the same aircraft. An airplane can deadhead (fly without passengers) to pick up passengers at another airport or to return home. The objective is to specify minimum cost pilot and jet itineraries, i.e., the flight legs and associated departure times, while satisfying all requests and respecting all constraints. Note that the DAFP can be viewed as a generalization of the dial-a-ride problem (see [20] for an overview) with the additional restriction that passengers make at most one intermediate stop between their origin and destination.

After experimenting with a couple of different formulations for the DAFP, including a column generation/branch-and-price approach, Espinoza et al. developed an integer multicommodity network flow model with side constraints. For each jet in the fleet, they construct a discretized (in minutes), time-activity graph with a node set that represents the key decisions taken from the perspective of the jet as it travels during the day between airports, e.g., which flights to make, when to make them, and which passengers to carry in each flight. Consequently, any itinerary for a single jet can be expressed as a path from the source to the sink in the corresponding graph for that jet.

Definition 7.1. A *feasible itinerary* for a single jet is defined as a sequence of flights satisfying the following conditions:

(R1) the jet begins and ends the itinerary at its home base,

(R2) the jet is available for the entire duration of the itinerary,

(R3) the jet never carries more passengers on a flight than its seating capacity,

(R4) the jet never carries more weight on a flight than its weight capacity,

(R5) the jet does not fly more than the maximum number of flying hours allowed by the FAA for a pilot during the day, and

(R6) the service requirements of all requests satisfied by the jet are met; that is, passengers are picked up at their origin and dropped off at their destination within their specified time window with at most one intermediate stop, and without changing airplanes during their trip.

By construction, any path from the source to the sink in one of the time-activity graphs corresponds to an itinerary for a single jet that satisfies conditions (R1), (R2) and (R6). However, knapsack inequalities are necessary to impose conditions (R3)-(R5) and ensure a feasible itinerary for each jet. All of the itineraries are linked together by set partitioning constraints which guarantee that all requests are satisfied, and flying time is used as a proxy for the operational cost of the itineraries since fuel consumption is not a linear function of the flying time (e.g., planes consume more fuel during takeoff and landing). For a rigorous treatment of the multicommodity network flow model, along with a detailed description of the graph construction procedure, we refer the reader to [39]. We now present a more concise version of the formulation below which is sufficient for our discussion.

Let $\mathcal{J}$ be the set of all jets in the fleet and $\mathcal{R}$ be the set of all transportation requests. For each $j \in \mathcal{J}$, let $G_j = (V_j, A_j)$ be the corresponding time-activity graph with source node s_j and sink node t_j. For each $(u, v) \in A_j$, assign a flying time $f_{uv} \in \mathbb{Z}_+$ and define a binary variable

$$x_{uv} = \begin{cases} 1, & \text{if jet } j \text{ uses arc } (u, v) \\ 0, & \text{otherwise.} \end{cases}$$

For each $v \in V_j$ and $r \in \mathcal{R}$, assign an indicator

$$\lambda_v^r = \begin{cases} 1, & \text{if request } r \text{ is loaded on board jet } j \text{ at node } v \\ 0, & \text{otherwise.} \end{cases}$$

Define $K_j \in \mathbb{Z}_+$ to be the number of knapsack inequalities added for each $j \in \mathcal{J}$. We can

formulate the DAFP as follows

$$\min \quad \sum_{j\in\mathcal{J}} \sum_{(u,v)\in A_j} f_{uv}x_{uv}$$

$$\text{s.t.} \quad x(\delta^+(v)) - x(\delta^-(v)) = \begin{cases} 1, & v = s_j,\ \forall\ j \in \mathcal{J} \\ -1, & v = t_j,\ \forall\ j \in \mathcal{J} \\ 0, & \forall\ v \in V_j \setminus \{s_j, t_j\},\ \forall\ j \in \mathcal{J} \end{cases} \tag{7.1}$$

$$\sum_{(u,v)\in A_j} a^k_{uv}x_{uv} \leq b_k,\ \forall\ k = 1,\ldots,K_j,\ \forall\ j \in \mathcal{J} \tag{7.2}$$

$$\sum_{j\in\mathcal{J}} \sum_{(u,v)\in A_j} \lambda^r_v x_{uv} = 1,\ \forall\ r \in \mathcal{R} \tag{7.3}$$

$$x_{uv} \in \{0,1\},\ \forall\ (u,v) \in A_j,\ \forall\ j \in \mathcal{J}, \tag{7.4}$$

where $a^k \in \mathbb{Z}_+^{A_j}$ and $b^k \in \mathbb{Z}_+$ for all $k = 1,\ldots,K_j$ and $j \in \mathcal{J}$. The flow balance constraints (7.1) guarantee that x defines $|\mathcal{J}|$ paths, one s_j-t_j path for each $j \in \mathcal{J}$, satisfying conditions (R1), (R2) and (R6). The knapsack inequalities (7.2) ensure that each path respects conditions (R3)-(R5). For each k, the knapsack coefficient a^k_{uv} may represent the number of passengers transported, the weight carried, or the flying time for $(u,v) \in A_j$, and, correspondingly, b_k may represent the seating capacity, the weight capacity, or the maximum number of flying hours allowed by the FAA. The set partitioning constraints (7.3) force all requests to be satisfied exactly once, and the integrality constraints (7.4) forbid fractional paths. It should be noted that the time-activity graphs are acyclic [39], and there is no need for inequalities which forbid cycles.

7.3 Branch-and-Cut Algorithm

Espinoza et al. [39] solve the DAFP using CPLEX' branch-and-cut algorithm which is designed for general mixed integer programs (MIPs). However, the feasibility set of the DAFP,

$$X^{DAFP} = \{x : \text{subject to (7.1), (7.2), (7.3) and (7.4)}\},$$

has structure that allows us to solve the problem more effectively. If the set partitioning constraints (7.3) are relaxed, the problem decomposes into $|\mathcal{J}|$ RCSPPs with feasibility sets

$$X^{RCP}_j = \left\{x \in \mathbb{B}^{A_j} : \text{subject to (7.1) and (7.2) for jet } j\right\},\ \forall\ j \in \mathcal{J}.$$

We have examined various reformulation techniques for the RCSPP which can be used to strengthen the linear programming (LP) relaxation for each individual X_j^{RCP}, i.e., preprocessing techniques, cutting planes and branching schemes. These techniques can also be applied in a branch-and-cut algorithm for the DAFP with no major modifications, and we briefly discuss the three fundamental components below.

Preprocessing and Probing A variety of techniques were developed in [39] to control the size of the underlying graphs and tighten the integrality gap for each X_j^{RCP}. While it is possible to use the preprocessing scheme of Aneja et al. [2] for each $j \in \mathcal{J}$ to eliminate nodes and arcs that cannot appear in a feasible s_j-t_j path in G_j, the network aggregation algorithm described by Espinoza et al. is capable of providing a stronger LP relaxation bound. The aggregation algorithm, which can be used for any RCSPP and is especially useful when the resource constraints are very tight, contracts individual nodes and uses bounds obtained from shortest paths with respect to each resource to eliminate some of the newly created arcs. Moreover, these new arcs, which correspond to partial itineraries flown by the plane, allow us to discard some of the resource constraints by enforcing them within the underlying graph. Therefore, we begin to experience some of the advantages of a column generation approach in which the variables correspond to a full itinerary for a single plane. Note that a feasible path in some G_j does not necessarily define an incumbent solution for the DAFP because it may not satisfy all of the requests exactly once. Consequently, the preprocessing scheme of Dumitrescu and Boland [35] is no longer valid for each X_j^{RCP}. However, we can utilize all three of the acyclic preprocessing enhancements introduced in Section 5.2.2, such as the identification of redundant resource constraints, since the underlying graphs do not contain cycles.

Cutting Planes Clearly, the more nodes we contract with the network aggregation algorithm in [39], the closer we get to convexifying each set X_j^{RCP}. It should be noted, however, that the number of newly created arcs grows exponentially with respect to the number of contracted nodes, and the improved integrality gap comes at the expense of a larger formulation with many more variables. Espinoza et al. control the size of the formulation

by contracting only those nodes with a relatively small number of incident arcs, but this implicitly limits the strength of the LP relaxation. To continue improving the integrality gap, we can generate cutting planes at any node in the branch-and-bound tree. Any valid inequality for some X_j^{RCP} is also valid for $X^{DAFP} \subseteq X_1^{RCP} \times \cdots \times X_{|\mathcal{J}|}^{RCP}$. Therefore, all of the inequality classes for the RCSPP outlined in Section 5.3 can be incorporated (cycle elimination inequalities will be redundant in this case).

Branching Schemes If the solution to the LP relaxation at some node n in the branch-and-bound tree is not integral, the conventional strategy for the DAFP is to branch on a fractional arc (variable) in one of the underlying graphs, G_j. Once the two new restricted subproblems have been created by fixing the arc to zero and one, we can use the subgraph of G_j implied by each of the arc fixings along with the RCSPP preprocessing scheme of Aneja et al. [2] to set additional arcs to zero as described in Section 5.4. Not only does this strengthen the LP relaxation for any node in the subtree rooted at n, but it also reduces the computational effort required to solve these LPs. In addition, it is possible to rerun the network aggregation algorithm in [39] at each node in the branch and bound tree, but the algorithm may be computationally expensive and requires us to generate columns within the branch-and-bound tree. Of course, all of the alternative branching schemes for the RCSPP discussed in Section 5.4, including node branching, are also applicable.

7.4 Computational Experiments

In this section, we investigate the performance of our branch-and-cut algorithm on various instances of the DAFP. All of the instances were formulated with the software developed by Espinoza et al. for DayJet using strategies that were shown to be effective in [39]. This includes preprocessing with the network aggregation algorithm. Thus, we need only justify the use of the other two major components, i.e., branching schemes and cutting planes.

We begin with a set of baseline computational results in which each instance is solved using CPLEX' branch-and-cut algorithm with the default parameter settings. Then, for each component, we present computational experiments which compare a few combinations

of the implementation choices explored in Chapter 6. Once the best combination has been identified for the branching scheme component, those choices are fixed, and the corresponding results become the new baseline for the cutting plane experiments. The ultimate goal of this computational study is to demonstrate that incorporating each of the components leads to an algorithm that can outperform a generic branch-and-cut implementation for small instances of the DAFP.

All computational experiments were performed on a cluster of Sun X2100 machines. Each machine has a dual-core AMD Opteron 175 processor, running at 2.2 GHz, and 4 GB of RAM. The source code was written in the C++ programming language, compiled using GCC version 3.2 with code optimization -O3, and executed in Red Hat Enterprise Linux AS, release 4. All instances were solved using CPLEX 9.0, and each branch-and-cut component in our algorithm was incorporated using callback functions available in the CPLEX callable library. A time limit of 3600 CPU seconds was imposed in the solution of each instance. However, it should be noted that the time spent formulating the instances (building the underlying graphs and running the network aggregation algorithm) prior to branch-and-cut was not accounted for in the effective running times.

7.4.1 Problem Classes

For testing purposes, DayJet simulated 30 days of flight operations and provided the set of feasible schedules returned by their online accept/reject system before any offline optimization had occurred. By randomly selecting a subset of 5 planes from each of these schedules, we defined 30 small instances of the DAFP. Each instance consists of a list of transportation requests corresponding to only a few of the airports in the southeastern United States. Hence, we divided the instances into two problem classes, A and B, based on the number of corresponding airports. The requests in class A (B) involve transportation between 6 (8) different airports.

The software developed by Espinoza et al. for DayJet was used to generate an integer multicommodity network flow model for each instance. Using the results in [39] to guide our

selection, all of the underlying graphs were constructed with the most basic time discretization which contains only the "special" time instants described in the paper. In addition, each graph was preprocessed with the network aggregation algorithm at a level of $\kappa = 10$, i.e., only nodes $v \in V_j$ such that $|\delta^-(v)||\delta^+(v)| \leq |\delta^-(v)| + |\delta^+(v)| + 10$ are eligible for contraction.

The attributes for problem classes A and B can be found in Table 15. For each instance, we report the number of scheduled requests ($|\mathcal{R}|$), the size of the largest node set in the underlying graphs after aggregation (Max $|V_j|$), the size of the largest arc set in the underlying graphs after aggregation (Max $|A_j|$), the maximum number of knapsack constraints added for any jet after aggregation (Max K_j), the number of constraints in the formulation (Rows), the number of variables in the formulation (Cols), the number of nonzeros in the constraint matrix (Nzs), the value of the original linear programming relaxation with no cutting planes added (z_{LP}), the optimal value (z^*), the gap between z^* and z_{LP} (LP Gap%) which is computed as $100 \times |(z^* - z_{LP})/z^*|$, and the flying time incurred in the original feasible schedule provided by the online accept/reject system (U) which defines an upper bound on z^*.

In preliminary experiments with the acyclic preprocessing enhancements introduced in Section 5.2.2, neither matrix coefficient reduction, nor preprocessing on $G_j[(u, v)]$ (the subgraph induced by all s_j-t_j paths in G_j containing (u, v)) for all $(u, v) \in A_j$ and $j \in \mathcal{J}$, significantly reduced the integrality gaps. However, we were able to identify many of the redundant knapsack inequalities (7.2) by calculating the value of the 'longest' s_j-t_j path in G_j with arc lengths given by the vector a^k for all $k \in \{1, \ldots, K_j\}$ and $j \in \mathcal{J}$. For some instances, almost 75% of the knapsack constraints reported in Table 15 were identified as redundant. Therefore, we used this enhancement for all of the experiments in this chapter. Discarding these inequalities has no affect on the LP relaxation value, but it does minimize the computational effort required to solve the problem by reducing the size of the formulation.

Table 15: DAFP problem classes A and B

Class	Instance	$\|\mathcal{R}\|$	Max $\|V_j\|$	Max $\|A_j\|$	Max K_j	Rows	Cols	Nzs	z_{LP}	z^*	LP Gap%	U
A	1	40	204	2,428	503	2,469	11,972	60,006	2,578.0	2,608	1.2	2,700
	2	47	207	2,460	452	2,373	11,979	57,228	2,779.0	2,888	3.8	2,978
	3	59	348	4,466	814	4,807	22,123	124,326	3,003.5	3,106	3.3	3,402
	4	62	434	5,733	1,040	6,106	28,187	156,485	3,001.0	3,105	3.3	3,429
	5	60	378	5,471	1,145	6,264	27,029	163,338	2,699.9	2,797	3.5	3,345
	6	54	235	2,907	551	3,053	14,088	72,552	3,094.6	3,226	4.1	3,689
	7	58	325	4,503	1,149	5,768	21,883	125,409	3,200.2	3,314	3.4	3,628
	8	49	262	3,636	722	3,778	17,790	102,798	2,855.5	2,894	1.3	3,240
	9	58	384	4,882	992	5,658	23,720	121,921	2,846.0	2,994	4.9	3,403
	10	49	331	4,372	807	4,500	21,212	112,185	2,599.3	2,827	8.1	3,201
	11	47	255	3,272	732	3,893	16,096	84,837	2,259.5	2,410	6.2	2,894
	12	45	242	2,972	622	3,091	14,404	72,008	2,479.0	2,587	4.2	3,210
	13	62	463	6,185	1,206	6,723	30,369	173,226	3,237.3	3,371	4.0	3,537
	14	53	260	3,452	805	3,715	16,555	85,456	2,835.5	2,976	4.7	3,060
	15	63	398	5,196	1,035	5,443	25,899	137,868	2,965.5	3,133	5.3	3,176
	16	54	314	3,807	830	4,036	17,697	89,154	3,352.0	3,445	2.7	3,524
	17	56	352	4,790	1,187	6,015	23,810	132,456	2,715.7	2,894	6.2	3,111
	18	52	334	4,421	792	4,652	21,993	122,155	2,731.0	2,959	7.7	3,263
	19	60	308	4,106	764	4,316	20,160	106,867	3,180.7	3,356	5.2	3,410
	20	54	308	3,936	788	4,245	19,224	103,953	2,762.0	2,879	4.1	2,932
B	1	67	516	7,405	1,565	8,305	35,727	192,360	2,998.0	3,128	4.2	3,128
	2	60	475	6,661	1,425	6,888	31,538	166,191	2,819.8	2,971	5.1	3,129
	3	74	713	9,743	2,071	11,269	46,690	252,956	3,104.3	3,251	4.5	3,396
	4	70	680	9,825	2,131	11,820	48,525	284,600	3,004.6	3,165	5.1	3,226
	5	59	401	5,648	1,049	5,374	27,162	140,793	2,975.0	3,063	2.9	3,275
	6	65	339	4,411	808	4,138	21,771	103,240	3,472.7	3,557	2.4	3,772
	7	78	672	9,830	2,109	11,132	48,610	279,054	2,655.5	2,865	7.3	3,059
	8	72	502	7,157	1,408	7,496	35,569	183,342	3,416.0	3,510	2.7	3,678
	9	67	593	7,775	1,704	9,007	38,659	209,241	2,893.5	3,105	6.8	3,174
	10	69	665	9,348	2,020	10,598	44,410	246,117	2,781.5	3,042	8.6	3,216

7.4.2 Default CPLEX Experiments

As our first benchmark, we solved all of the DAFP instances using CPLEX with the default parameter settings. With these settings, the branch-and-cut algorithm generates cutting planes that are valid for general MIPs such as Gomory fractional cuts, knapsack cover cuts, and generalized upper bound cover cuts. For a comparison, we also solved the instances with all CPLEX cuts turned off, i.e., `CPX_PARAM_CUTPASS` set to -1.

Given that a feasible solution is available for each instance, we always initialized the upper cutoff value for the branch-and-cut algorithm (i.e., `CPX_PARAM_CUTUP`) with the flying time incurred in the original schedule provided by the online accept/reject system. CPLEX assures that any node in the branch-and-bound tree with an LP relaxation value above this cutoff is fathomed. As a result, any integral solutions that do not improve upon the original schedule never become incumbent candidates. Moreover, the cutoff helps to reduce the size of the branch-and-bound trees.

The full results of both experiments can be found in Appendix B in Table 28. The CPX setting refers to default CPLEX, and the CPX-C setting refers to CPLEX with its cuts turned off. For each instance and setting, we report several performance metrics such as the value of the best integer solution found (z_{IP}), the gap between z_{IP} and the value of the LP relaxation at the root node (Root Gap%) which is computed as $100 \times |(z_{IP} - z_{root})/z_{IP}|$, the gap between z_{IP} and the value of the best lower bound in the branch-and-bound tree (Final Gap%) which is computed as $100 \times |(z_{IP} - z_{LB})/z_{IP}|$, the number of nodes evaluated in the branch-and-bound tree before the best integer solution was found (z_{IP} Node), the total number of evaluated nodes in the branch-and-bound tree (B&B Nodes), and the effective running time of the branch-and-cut algorithm (t). If an instance was not solved to proven optimality, then the time limit of 3600 CPU seconds was reached. The reader can deduce which of the instances timed out by checking the final integrality gap.

We summarize these results in Table 16. For each problem class and setting, we report the averages for most of the performance metrics. We also report the total difference between the true optimal value and the value of the best integer solution found (Δz^*), the number of instances that were solved at the root node (#rt), and the number of instances

which timed out (#to). The fact that almost half of the experiments timed out confirms that these instances are not easily solved with CPLEX. The instances in class B have 33% more airports and 25% more requests (on average) than class A, which leads to formulations that are close to double in size. Therefore, it is not surprising that class B proves to be more challenging.

Table 16: Summary of the DAFP default CPLEX results

Class	Setting	Avg Root Gap%	Avg Final Gap% (Δz^*)	Avg z_{IP} Node	Avg B&B Nodes (#rt)	Avg t (#to)
A	CPX	2.9	0.9 (24)	4,243	12,735 (0)	1,532.69 (6)
	CPX-C	4.4	1.2 (1)	5,372	18,061 (0)	1,766.25 (8)
B	CPX	3.7	2.1 (286)	2,067	3,492 (1)	1,933.58 (5)
	CPX-C	6.0	3.7 (355)	3,896	4,871 (0)	2,299.25 (6)

We see in Table 16 that the CPLEX cuts decreased the integrality gaps at the root node for both of the problem classes, which led to a decrease in the number of branch-and-bound nodes evaluated. Moreover, the final integrality gaps were also improved for those instances that timed out. When comparing results in this chapter, we will use running time as our primary performance metric. Since the use of CPLEX cuts led to smaller run times and fewer timeouts, we turned on CPLEX cuts for all subsequent experiments in this chapter.

7.4.3 Branching Scheme Experiments

In our next series of experiments, we investigate the alternate branching schemes proposed in Section 5.4. Initially, we compare enhanced arc branching and branching on fractional nodes in the underlying graphs.

By default, CPLEX branches on a fractional arc (u, v) in some G_j at any node in the branch-and-bound tree. However, we used the branch callback functionality in the callable library for the enhanced arc branching experiments. Once (u, v) was set to zero or one in the two new subproblems, additional arcs in G_j were fixed to zero with the trivial function ST_CONNECTED from Algorithm 11 in Section 5.4 for the downward branch and by defining the subgraph induced by all s_j-t_j paths containing (u, v) for the upward branch. Since we made no estimates on the optimal value of either branch, we used the scheme suggested by

Land and Powell [85] and asked CPLEX to follow the downward branch first if $x^*_{ij} < 0.5$ and the upward branch first otherwise.

The branch callback functionality was also used for the node branching experiments. By choosing a fractional arc at any node in the branch-and-bound tree, CPLEX implicitly identifies an underlying graph G_j with fractional flow. Instead of branching on that arc, we considered three strategies to select a fractional node in G_j to branch upon:

1. choose a fractional node v that minimizes $\min\{x^*(\delta^+(v)), 1 - x^*(\delta^+(v))\}$,
2. choose a fractional node v that maximizes $\min\{x^*(\delta^+(v)), 1 - x^*(\delta^+(v))\}$,
3. choose the first fractional node in a topological ordering.

Once the fractional node v was chosen, we defined the fixings for both of the branches. For the downward branch, all of the arcs in $\delta^-(v) \cup \delta^+(v)$ were set to zero, and ST_CONNECTED was used to fix additional arcs in G_j to zero. For the upward branch, the constraint $x(\delta^+(v)) = 1$ was not added to the subproblem because the subgraph induced by all s_j-t_j paths containing v can be precisely defined for the acyclic case. Thus, we fixed all arcs in G_j that were not members of this subgraph to zero. Finally, we asked CPLEX to follow the downward branch first if $x^*(\delta^+(v)) < 0.5$ and the upward branch first otherwise.

The full results of these experiments can be found in Appendix B in Table 29. The ARC setting refers to enhanced arc branching, and the NDs setting for $s = 1, 2, 3$ refers to node branching with fractional node selection strategy s in the list above. For these results, we introduce a new performance metric: the non-CPLEX time spent on branching (t_B). The results are summarized in Table 17. None of the branching schemes led to smaller run times when compared to the default CPLEX results. Still, it is clear that strategy 3 (topological ordering) outperforms the other two fractional node selection strategies.

In a second set of experiments, we tried to improve the two most effective branching schemes above, i.e., enhanced arc branching and node branching with fractional node selection strategy 3, by applying the preprocessing scheme of Aneja et al. [2] to fix an even greater number of arcs to zero in the branching subproblems. We repeatedly called the

Table 17: Summary of the DAFP branching scheme results

Class	Setting	Avg Root Gap%	Avg Final Gap% (Δz^*)	Avg z_{IP} Node	Avg B&B Nodes (#rt)	Avg t_B	Avg t (#to)
A	ARC	3.3	1.3 (302)	1,238	22,178 (0)	31.7	1,594.28 (7)
	ND1	3.2	2.0 (284)	3,111	22,507 (0)	41.5	2,078.52 (10)
	ND2	2.8	1.2 (1)	1,813	24,983 (0)	50.6	2,011.95 (9)
	ND3	2.8	0.9 (0)	537	21,578 (0)	42.2	1,702.44 (8)
B	ARC	3.4	1.6 (181)	649	5,291 (1)	15.0	1,948.36 (4)
	ND1	4.3	3.5 (484)	647	2,997 (1)	11.0	2,223.69 (6)
	ND2	3.5	2.2 (209)	640	3,333 (1)	13.6	2,069.82 (5)
	ND3	2.9	1.3 (7)	485	3,928 (1)	14.9	2,057.78 (5)

function AAN_Preprocessing from Algorithm 9 in Section 5.2 on the corresponding subgraph for each branch until no other reductions were possible. The full results of this second set of experiments can be found in Appendix B in Table 30, and a summary is presented in Table 18.

Table 18: Summary of the DAFP branching scheme results with preprocessing

Class	Setting	Avg Root Gap%	Avg Final Gap% (Δz^*)	Avg z_{IP} Node	Avg B&B Nodes (#rt)	Avg t_B	Avg t (#to)
A	ARC+AAN	3.1	1.0 (203)	1,607	13,312 (0)	113.1	1,288.59 (6)
	ND3+AAN	2.8	0.8 (0)	660	18,639 (0)	140.9	1,606.93 (7)
B	ARC+AAN	3.4	1.8 (174)	833	2,786 (1)	159.9	1,913.67 (4)
	ND3+AAN	3.1	1.2 (70)	418	3,733 (1)	128.7	1,675.32 (2)

We see that the additional preprocessing with the scheme of Aneja et al. decreased the number of evaluated branch-and-bound nodes for all of the problem classes and branching schemes. When combined with additional preprocessing, enhanced arc branching led to the fewest timeouts and the quickest run times for problem class A, despite the increase in non-CPLEX time spent on branching. Conversely, node branching performed best for problem class B. For consistency, we use enhanced arc branching with additional preprocessing for all of the remaining experiments in this chapter since it is the only scheme that led to smaller run times for both classes when compared to the default CPLEX results in Table 16.

7.4.4 Cutting Plane Experiments

We have already established that the cuts generated by CPLEX for general MIPs are effective for these particular DAFP instances. In our final series of experiments, we also incorporate the majority of the valid inequality classes for the RCSPP outlined in Section 5.3 and evaluate their performance.

We know from Chapter 6 that many violated RCSPP inequalities can be found at each node in the branch-and-bound tree, but adding too many cutting planes in a branch-and-cut algorithm can lead to prohibitively large LP relaxations and longer overall running times. In the absence of a more sophisticated cut management scheme, e.g., CPLEX 9.0 does not allow the user to delete cuts after they have been added with the cut callback functionality in the callable library, we decided to generate cutting planes only at the root node for all of the DAFP cutting plane experiments. If the solution to the LP relaxation at the root node was not integer feasible, the set of jets $J^* \subseteq \mathcal{J}$ associated with the fractional flow was identified. Then, for every X_j^{RCP} such that $j \in J^*$, we solved the separation problem for each of the following RCSPP inequality classes in a round of cut generation:

- node precedence inequalities (3.8) and (3.9)
- subpath precedence inequalities (3.12) and (3.13)
- s-t cut precedence inequalities (3.15) and (3.16)
- clique inequalities (3.19) based on the conflict graph described in Section 3.6.2
- lifted odd hole inequalities (3.21) based on the conflict graph described in Section 3.6.2
- lifted path subset cover inequalities (4.2).

With the exception of the lifted path subset cover inequalities (4.2), we applied jointly defined versions of these inequalities which are derived using multiple resources (refer to Section 5.3.3). Each inequality class was separated with the same conservative schemes described in Section 6.7 which involve tolerances and probabilities used to limit the number of generated cuts. If any violated inequalities were found during the current round of cut

generation, then we added all of the inequalities to the LP relaxation, reoptimized, and repeated our search for cuts with the new solution; otherwise, we branched. However, if the change in value of the LP relaxation was less than 10^{-3} between successive rounds, we branched to avoid a tailing off effect.

The full results of these experiments can be found in Appendix B in Table 31. The ALL setting refers to the use of all of the RCSPP inequality classes listed above. We also turned off each inequality class one at a time to see if their exclusion led to improved performance. If -c is appended to the setting for c = NDP (node precedence inequalities), SPP (subpath precedence inequalities), CTP (s-t cut precedence inequalities), CLQ (clique inequalities), LOH (lifted odd hole inequalities), LPS (lifted path subset cover inequalities), then inequality class c was turned off for those experiments. The final two performance metrics that must be introduced for these results are the number of non-CPLEX cutting planes added (Cuts), and the time spent separating non-CPLEX cuts (t_C).

A summary of the results is presented in Table 19. We see that generating all of the RCSPP inequalities (ALL) decreased the integrality gaps at the root node to under 1% for both problem classes, which is extremely tight and resembles the strength of typical column generation formulations. Furthermore, the total number of instances solved without having to branch (#rt) increased to five. Only two of the instances resulted in a time out, but in both cases, the best integer solution was optimal (i.e., $\Delta z^* = 0$). These are significant improvements over the best branching scheme results in Table 18.

It is difficult to draw definite conclusions about which individual inequality class should be turned off. The lifted path subset cover inequalities (4.2) seemed to be the most critical inequality class because turning them off led to significantly longer run times for both problem classes. Conversely, turning off the clique inequalities (3.19) led to improved run times for both problem classes. Turning off each of the remaining inequality classes improved the performance for class B, but led to worse results for class A. Nevertheless, all of the settings outperformed the best branching scheme results for both problem classes.

Table 19: Summary of the DAFP cutting plane results

Class	Setting	Avg Cuts	Avg Root Gap%	Avg Final Gap% (Δz^*)	Avg z_{IP} Node	Avg B&B Nodes (#rt)	Avg t_C	Avg t_B	Avg t (#to)
A	ALL	405	0.8	0.0 (0)	116	13,188 (3)	3.1	60.6	662.55 (1)
	ALL-NDP	297	0.8	0.1 (0)	300	11,931 (2)	3.1	55.1	783.32 (2)
	ALL-SPP	331	0.9	0.1 (0)	162	14,911 (3)	3.0	71.5	859.18 (3)
	ALL-CTP	343	0.8	0.2 (0)	216	14,989 (3)	2.7	73.0	846.51 (4)
	ALL-CLQ	406	0.8	0.0 (0)	203	11,427 (3)	3.0	51.4	604.26 (1)
	ALL-LOH	349	0.8	0.1 (0)	477	13,574 (5)	3.0	63.4	749.57 (3)
	ALL-LPS	404	1.1	0.2 (0)	112	14,391 (2)	2.3	71.6	936.74 (3)
B	ALL	533	0.7	0.0 (0)	99	2,022 (2)	17.0	62.3	734.72 (1)
	ALL-NDP	396	0.7	0.0 (0)	37	1,798 (2)	17.4	57.6	574.29 (0)
	ALL-SPP	425	0.7	0.0 (0)	104	2,010 (2)	17.1	59.6	670.29 (1)
	ALL-CTP	456	0.7	0.0 (0)	47	2,324 (3)	16.4	73.2	657.79 (0)
	ALL-CLQ	512	0.7	0.1 (0)	68	2,381 (3)	17.0	71.2	695.34 (1)
	ALL-LOH	446	0.7	0.0 (0)	128	1,882 (3)	16.6	66.7	561.07 (0)
	ALL-LPS	520	1.7	0.9 (194)	170	2,398 (2)	14.0	91.0	1,257.15 (2)

7.5 Summary

In this chapter we have discussed a branch-and-cut algorithm for the DAFP and investigated various implementation choices for each component. The best results from each series of experiments are presented in Table 20. For each problem class and setting, we report the average gap between the value of the best integer solution found and the value of the best lower bound in the branch-and-bound tree (Avg Final Gap%), the total difference between the true optimal value and the value of the best integer solution found (Δz^*), the average number of nodes evaluated in the branch-and-bound tree before the best integer solution was found (Avg z_{IP} Node), the total number of instances that were solved at the root node (#rt), the total effective running time of the branch-and-cut algorithm over all instances (Total t), the total number of instances which timed out (#to), and the performance ratio which is computed by dividing the total running time for default CPLEX by the total running time for that particular setting.

We see positive trends for all of the performance metrics in Table 20 as each combination of implementation choices is fixed. It is clear that incorporating each of the components led to an algorithm that outperforms default CPLEX for small instances of the DAFP. Naturally, the branch-and-cut algorithm can also be used to find improved solutions for much larger instances by integrating it within the parallel local search scheme of Espinoza

Table 20: Final summary of the DAFP results

Class	Setting	Avg Final Gap% (Δz^*)	Avg z_{IP} Node (#rt)	Total t (#to)	Performance Ratio
A	CPX	0.9 (24)	4,243 (0)	30,653.71 (6)	—
	ARC+AAN	1.0 (203)	1,607 (0)	25,771.82 (6)	1.2X
	ALL-CLQ	0.0 (0)	203 (3)	12,085.24 (1)	2.5X
B	CPX	2.1 (286)	2,067 (1)	19,335.75 (5)	—
	ND3+AAN	1.2 (70)	418 (1)	16,753.22 (2)	1.2X
	ALL-LOH	0.0 (0)	128 (3)	5,610.69 (0)	3.5X

et al. [40] which is the state of the art for very large DAFP instances. It should be noted that Espinoza et al. impose a time limit on the solution of each integer program because they only want to spend a limited amount of time exploring each neighborhood. If our algorithm is to enhance their scheme, it must be able to find good feasible solutions very early in the search. Fortunately, the improvement in the number of nodes evaluated in the branch-and-bound tree before the best integer solution was found was even more impressive than the improvement in effective run times for both problem classes when both branch-and-cut components were incorporated.

CHAPTER 8

CONCLUSIONS

In this thesis, we used integer programming techniques to solve the resource constrained shortest path problem (RCSPP) and the dial-a-flight problem (DAFP) with branch-and-cut. We now summarize the main contributions and offer suggestions for future research.

8.1 Main Contributions

Precedence Inequalities for the WCSPP Up to now, research on lower bounds for the RCSPP has focused on Lagrangean relaxation. To the best of our knowledge, the branch-and-cut algorithm of Spoorendonk et al. [106] is the only case in which cutting planes, other than some variant of the subtour elimination inequalities, were used to solve the problem. In Chapter 3, we introduced three new precedence inequality classes (node, subpath and s-t cut) which are valid for the single resource relaxation, i.e., the weight constrained shortest path problem (WCSPP). Reverse versions of these precedence inequality classes were also introduced. Separation routines were provided, and we established a result that leads to strengthened versions of all of the new precedence inequalities.

Conflict Graph for the WCSPP For any binary integer program (BIP), we can define an auxiliary conflict graph used to obtain a node packing polytope which describes a relaxation of the convex hull of feasible solutions. Therefore, any valid inequalities for the node packing polytope, including the clique and odd cycle inequalities, are also valid for the BIP. In Chapter 3, we outlined a polynomial time algorithm used to construct a conflict graph for the WCSPP which yields a stronger relaxation than conflict graphs approximated with typical BIP probing techniques such as those described by Atamturk et al. [5].

Fixing Subpaths in G There are occasions in a branch-and-cut algorithm, e.g., during preprocessing and branching, when we might want to consider only those s-t paths in the

underlying graph G which contain a given subpath Q. In Chapter 4, we established a necessary condition for any s-t path P containing Q by defining a subgraph of G which must contain P. As a consequence, we get an efficient preprocessing scheme (the subgraph can be defined in polynomial time) which removes incompatible nodes and arcs from G. Moreover, we established that this condition is sufficient when G is acyclic, which allows us to forbid all s-t paths that do not contain Q without introducing additional constraints.

Path Subset Cover Inequalities for the WCSPP The ability to fix a set of disjoint s-t subpaths in the underlying graph led to a new valid inequality class for the WCSPP. In Chapter 4, we introduced the path subset cover inequalities which are stronger than the typical cover inequalities used for general BIPs. We proved that the separation problem is $\mathcal{NP}$-hard, thus, two separation heuristics were provided.

Lifted Path Subset Cover Inequalities for the WCSPP Valid inequalities are often strengthened using sequential lifting techniques. In Chapter 4, we considered sequential lifting for the path subset cover inequalities when the underlying graph is acyclic. We introduced a full-dimensional relaxation for the WCSPP, known as the path subset polytope, and proved that a minimal path subset cover is facet-defining for a lower dimensional subpolytope of the relaxation. The lifting problem was discussed and shown to be equivalent to a WCSPP which is solvable in polynomial time with typical node labeling methods. We developed a faster lifting scheme which exploits the relationship between consecutive lifting problems by considering only those lifting sequences such that all outgoing arcs for the same node are lifted consecutively, and the nodes are evaluated in topological order.

Strength of the Node Precedence Inequalities for the WCSPP Due to the infrequent use of branch-and-cut, very little is known about the actual RCSPP polytope. In Chapter 4, we gave the dimension of the convex hull of feasible solutions for the WCSPP when the underlying graph is acyclic. Then, we studied a projection of the convex hull onto a lower dimensional subspace and established sufficient conditions for which the strengthened version of the node precedence inequalities defines a facet of the projection.

Branch-and-Cut Algorithm for the RCSPP In Chapter 5, we discussed several of the fundamental components involved in a branch-and-cut algorithm for the RCSPP. We enhanced three general BIP preprocessing techniques for when the underlying graph is acyclic. Most of the new inequality classes for the WCSPP were generalized to consider multiple resources without increasing the overall complexity of separation, and we introduced stronger versions of these inequalities which are locally valid for the subtree rooted at some node in the branch-and-bound tree. Alternative branching strategies were developed which lead to strengthened linear programming (LP) relaxations and balanced search trees, and we described a primal heuristic scheme that uses fractional solutions, along with the current incumbent, to search for new feasible solutions throughout the branch-and-bound tree. We conducted a computational study to evaluate several implementation choices for these components and presented the results in Chapter 6. The results indicate that our algorithm outperforms CPLEX' default branch-and-cut algorithm.

Branch-and-Cut Algorithm for the DAFP In Chapter 7, we extended our RCSPP branch-and-cut algorithm to solve the DAFP, which can be formulated as an integer multicommodity network flow model consisting of several RCSPPs linked together by set partitioning constraints which guarantee that all travel requests are satisfied. We performed computational experiments with practical instances provided by the DayJet Corporation, and the results verified that the extended algorithm also outperforms CPLEX' default branch-and-cut algorithm.

8.2 Future Research

Clearly, the reformulation techniques used in our branch-and-cut algorithms (i.e., preprocessing techniques, cutting planes and branching schemes) offer a promising approach for cases in which the RCSPP appears as a substructure in a more difficult optimization problem that cannot be solved with the typical node labeling methods or Lagrangean relaxation approaches. We believe that our branch-and-cut algorithm is also a viable alternative to

these typical approaches for some of the more difficult instances of the RCSPP. For example, there are RCSPP pricing problems in various column generation approaches which are strongly $\mathcal{NP}$-hard due to the presence of negative cost cycles. We have preliminary computational results which show that our algorithm is two orders of magnitude faster than default CPLEX for some instances of a vehicle routing pricing problem with negative cost cycles. We intend to implement additional computational tests to compare the performance of our algorithm for these instances with node labeling methods from the literature which guarantee elementary paths.

CPLEX 9.0 did not allow us to delete inactive cutting planes after they had been added to the LP relaxation with the cut callback functionality in the callable library. Therefore, we controlled the size of the LP relaxations in our computational experiments by generating cutting planes only at the root node of the branch-and-bound tree with very conservative settings. We would like to implement a sophisticated cut management scheme using a different platform which will allow us to evaluate more aggressive cut generation approaches, e.g., adding the (stronger) locally valid versions of the RCSPP inequalities throughout the search tree.

If cutting planes are to be added frequently, then we must find ways to generate them more efficiently. In particular, the sequential lifting techniques used for the lifted path subset cover inequalities call for the solution of multiple optimization problems to determine the lifting coefficients, and the resulting inequalities depend on the order in which the variables were lifted. We limited some of the effort by using a topological lifting scheme that exploits the relationship between consecutive lifting problems. However, it might be possible to apply the ideas of sequence independent lifting for general BIPs (see [59]) which require the solution of a single optimization problem and reduce the computational burden.

Further investigation of the RCSPP polytope may lead to additional valid inequality classes. Yet, the ability to completely characterize the convex hull of feasible solutions for the RCSPP would not suffice for the DAFP. The best one could hope for is an LP relaxation bound which is equivalent to the bound obtained by a column generation approach in which the variables correspond to a full itinerary for a single plane. Future work for the DAFP

should incorporate valid inequalities which are derived by combining the RCSPPs from multiple planes.

It should be noted that we have used an additive notion of path feasibility for the RCSPP throughout this thesis, but researchers have considered other, more general, resource constraints. For example, time window constraints are more complicated than a simple knapsack constraint since waiting time is allowed before processing begins at each node. In future work, we plan to extend our branch-and-cut algorithm to consider such non-additive resources.

APPENDIX A

COMPUTATIONAL RESULTS FOR THE RESOURCE CONSTRAINED SHORTEST PATH PROBLEM

Table 21: RCSPP default CPLEX results

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A1	3	CPX	98	935	2,656	0	0	0.0	0.0	2	0	0	0.0	0.0	0.0	0.0	0.01
		CPX-C	98	935	2,656	0	0	25.0	0.0	2	0	0	0.0	0.0	0.0	0.0	0.01
A1	4	CPX	98	935	2,656	0	0	0.0	0.0	2	0	0	0.0	0.0	0.0	0.0	0.01
		CPX-C	98	935	2,656	0	0	0.0	0.0	2	0	0	0.0	0.0	0.0	0.0	0.01
A1	7	CPX	100	895	9,434	0	0	15.4	0.0	6	0	0	0.0	0.0	0.0	0.0	0.07
		CPX-C	100	895	9,434	0	0	30.7	0.0	6	10	14	0.0	0.0	0.0	0.0	0.11
A1	8	CPX	100	895	9,434	0	0	54.7	0.0	14	109	153	0.0	0.0	0.0	0.0	0.76
		CPX-C	100	895	9,434	0	0	61.6	0.0	14	215	233	0.0	0.0	0.0	0.0	1.04
A1	11	CPX	192	1,911	5,462	0	0	0.0	0.0	6	0	0	0.0	0.0	0.0	0.0	0.02
		CPX-C	192	1,911	5,462	0	0	0.0	0.0	6	0	0	0.0	0.0	0.0	0.0	0.02
A1	12	CPX	192	1,911	5,462	0	0	0.0	0.0	6	0	0	0.0	0.0	0.0	0.0	0.02
		CPX-C	192	1,911	5,462	0	0	0.0	0.0	6	0	0	0.0	0.0	0.0	0.0	0.02
A1	15	CPX	194	1,795	18,987	0	0	14.2	0.0	9	32	32	0.0	0.0	0.0	0.0	0.63
		CPX-C	194	1,795	18,987	0	0	23.8	0.0	9	0	4	0.0	0.0	0.0	0.0	0.16
A1	16	CPX	194	1,795	18,987	0	0	37.3	0.0	17	170	245	0.0	0.0	0.0	0.0	3.78
		CPX-C	194	1,795	18,987	0	0	47.1	0.0	17	36	67	0.0	0.0	0.0	0.0	1.18
A1	19	CPX	469	4,681	13,377	0	0	0.0	0.0	6	0	0	0.0	0.0	0.0	0.0	0.05
		CPX-C	469	4,681	13,377	0	0	0.0	0.0	6	0	0	0.0	0.0	0.0	0.0	0.05
A1	20	CPX	469	4,681	13,377	0	0	0.0	0.0	6	0	0	0.0	0.0	0.0	0.0	0.05
		CPX-C	469	4,681	13,377	0	0	0.0	0.0	6	0	0	0.0	0.0	0.0	0.0	0.05
A1	23	CPX	479	4,572	49,485	0	0	0.0	0.0	4	0	0	0.0	0.0	0.0	0.0	0.92
		CPX-C	479	4,572	49,485	0	0	12.7	0.0	4	0	0	0.0	0.0	0.0	0.0	0.47
A1	24	CPX	479	4,572	49,485	0	0	0.0	0.0	5	0	0	0.0	0.0	0.0	0.0	0.58
		CPX-C	479	4,572	49,485	0	0	14.8	0.0	5	6	6	0.0	0.0	0.0	0.0	1.23
A2	3	CPX	98	935	2,656	0	0	0.0	0.0	-78	0	0	0.0	0.0	0.0	0.0	0.02
		CPX-C	98	935	2,656	0	0	1.0	0.0	-78	0	0	0.0	0.0	0.0	0.0	0.02
A2	4	CPX	98	935	2,656	0	0	0.0	0.0	-74	0	0	0.0	0.0	0.0	0.0	0.02
		CPX-C	98	935	2,656	0	0	0.3	0.0	-74	0	0	0.0	0.0	0.0	0.0	0.02
A2	7	CPX	100	895	9,434	0	0	9.8	0.0	-29	0	17	0.0	0.0	0.0	0.0	0.27
		CPX-C	100	895	9,434	0	0	12.6	0.0	-29	30	39	0.0	0.0	0.0	0.0	0.35
A2	8	CPX	100	895	9,434	0	0	17.0	0.0	-23	20	68	0.0	0.0	0.0	0.0	0.43
		CPX-C	100	895	9,434	0	0	22.5	0.0	-23	72	74	0.0	0.0	0.0	0.0	0.34
A2	11	CPX	192	1,911	5,462	0	0	0.0	0.0	-125	0	0	0.0	0.0	0.0	0.0	0.05
		CPX-C	192	1,911	5,462	0	0	0.5	0.0	-125	0	0	0.0	0.0	0.0	0.0	0.05

◇ Problem is infeasible

Table 21 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A2	12	CPX	192	1,911	5,462	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.04
		CPX-C	192	1,911	5,462	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.04
A2	15	CPX	194	1,795	18,987	0	0	35.8	0.0	-24	240	715	0.0	0.0	0.0	0.0	9.22
		CPX-C	194	1,795	18,987	0	0	38.4	0.0	-24	40	385	0.0	0.0	0.0	0.0	4.20
A2	16	CPX	194	1,795	18,987	0	0	62.9	0.0	-17	244	267	0.0	0.0	0.0	0.0	7.23
		CPX-C	194	1,795	18,987	0	0	69.5	0.0	-17	0	197	0.0	0.0	0.0	0.0	2.35
A2	19	CPX	469	4,681	13,377	0	0	0.0	0.0	-99	0	0	0.0	0.0	0.0	0.0	0.15
		CPX-C	469	4,681	13,377	0	0	0.0	0.0	-99	0	0	0.0	0.0	0.0	0.0	0.15
A2	20	CPX	469	4,681	13,377	0	0	0.0	0.0	-92	0	0	0.0	0.0	0.0	0.0	0.12
		CPX-C	469	4,681	13,377	0	0	0.0	0.0	-92	0	0	0.0	0.0	0.0	0.0	0.12
A2	23	CPX	479	4,572	49,485	0	0	23.9	0.0	-36	1,093	1,679	0.0	0.0	0.0	0.0	73.69
		CPX-C	479	4,572	49,485	0	0	25.4	0.0	-36	270	895	0.0	0.0	0.0	0.0	40.28
A2	24	CPX	479	4,572	49,485	0	0	27.9	0.0	-29	70	496	0.0	0.0	0.0	0.0	31.42
		CPX-C	479	4,572	49,485	0	0	31.7	0.0	-29	311	401	0.0	0.0	0.0	0.0	25.26
A3	1	CPX	876	8,596	52,337	0	0	12.1	0.0	-44	480	787	0.0	0.0	0.0	0.0	59.30
		CPX-C	876	8,596	52,337	0	0	14.5	0.0	-44	45	236	0.0	0.0	0.0	0.0	29.27
A3	2	CPX	876	8,596	52,337	0	0	10.4	0.0	-64	1,749	2,547	0.0	0.0	0.0	0.0	134.45
		CPX-C	876	8,596	52,337	0	0	11.8	0.0	-64	30	982	0.0	0.0	0.0	0.0	58.36
A3	3	CPX	876	8,596	52,337	0	0	5.4	0.0	-85	0	216	0.0	0.0	0.0	0.0	15.35
		CPX-C	876	8,596	52,337	0	0	6.7	0.0	-85	0	171	0.0	0.0	0.0	0.0	11.81
A3	4	CPX	881	8,596	87,365	0	0	27.5	0.0	-32	360	2,237	0.0	0.0	0.0	0.0	355.73
		CPX-C	881	8,596	87,365	0	0	31.6	0.0	-32	240	1,778	0.0	0.0	0.0	0.0	259.28
A3	5	CPX	881	8,596	87,365	0	0	15.4	0.0	-54	2,000	9,225	0.0	0.0	0.0	0.0	1,320.47
		CPX-C	881	8,596	87,365	0	0	16.2	0.0	-54	1,230	5,853	0.0	0.0	0.0	0.0	546.30
A3	6	CPX	881	8,596	87,365	0	0	11.5	0.0	-73	620	7,025	0.0	0.0	0.0	0.0	518.91
		CPX-C	881	8,596	87,365	0	0	12.4	0.0	-73	730	5,647	0.0	0.0	0.0	0.0	452.77
A3	7	CPX	882	8,711	53,009	0	0	15.9	0.0	-44	40	710	0.0	0.0	0.0	0.0	58.50
		CPX-C	882	8,711	53,009	0	0	18.9	0.0	-44	490	736	0.0	0.0	0.0	0.0	61.99
A3	8	CPX	882	8,711	53,009	0	0	7.1	0.0	-70	560	772	0.0	0.0	0.0	0.0	61.46
		CPX-C	882	8,711	53,009	0	0	8.4	0.0	-70	280	473	0.0	0.0	0.0	0.0	66.77
A3	9	CPX	882	8,711	53,009	0	0	5.3	0.0	-89	1,560	1,832	0.0	0.0	0.0	0.0	84.17
		CPX-C	882	8,711	53,009	0	0	6.0	0.0	-89	328	815	0.0	0.0	0.0	0.0	49.29
A3	10	CPX	888	8,712	88,550	0	0	37.1	0.0	-33	3,550	8,072	0.0	0.0	0.0	0.0	1,367.41
		CPX-C	888	8,712	88,550	0	0	40.6	0.0	-33	1,630	4,700	0.0	0.0	0.0	0.0	749.79

◇ Problem is infeasible

Table 21 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	11	CPX	888	8,712	88,550	0	0	14.1	0.0	-60	3,850	6,664	0.0	0.0	0.0	0.0	665.83
		CPX-C	888	8,712	88,550	0	0	15.1	0.0	-60	5,282	6,472	0.0	0.0	0.0	0.0	529.09
A3	12	CPX	888	8,712	88,550	0	0	11.3	0.0	-79	11,153	15,769	0.0	0.0	0.0	0.0	1,289.31
		CPX-C	888	8,712	88,550	0	0	11.8	0.0	-79	15,675	15,675	0.0	0.0	0.0	0.0	1,116.61
A3	13	CPX	1,097	10,670	64,763	0	0	18.4	0.0	-42	420	869	0.0	0.0	0.0	0.0	93.45
		CPX-C	1,097	10,670	64,763	0	0	20.3	0.0	-42	347	527	0.0	0.0	0.0	0.0	62.27
A3	14	CPX	1,097	10,670	64,763	0	0	8.6	0.0	-67	0	356	0.0	0.0	0.0	0.0	47.58
		CPX-C	1,097	10,670	64,763	0	0	10.0	0.0	-67	240	467	0.0	0.0	0.0	0.0	61.28
A3	15	CPX	1,097	10,670	64,763	0	0	4.8	0.0	-89	0	171	0.0	0.0	0.0	0.0	12.76
		CPX-C	1,097	10,670	64,763	0	0	5.9	0.0	-89	120	157	0.0	0.0	0.0	0.0	44.16
A3	16	CPX	1,103	10,671	108,673	0	0	33.2	0.0	-32	880	2,744	0.0	0.0	0.0	0.0	655.34
		CPX-C	1,103	10,671	108,673	0	0	37.6	0.0	-32	460	1,602	0.0	0.0	0.0	0.0	392.90
A3	17	CPX	1,103	10,671	108,673	0	0	20.3	0.0	-55	4,984	13,689	0.0	0.0	0.0	0.0	1,934.22
		CPX-C	1,103	10,671	108,673	0	0	22.3	0.0	-55	3,205	8,742	0.0	0.0	0.0	0.0	1,197.90
A3	18	CPX	1,103	10,671	108,673	0	0	10.6	0.0	-78	4,620	7,492	0.0	0.0	0.0	0.0	977.01
		CPX-C	1,103	10,671	108,673	0	0	11.3	0.0	-78	980	3,129	0.0	0.0	0.0	0.0	540.29
A3	19	CPX	1,053	10,240	62,169	0	0	10.8	0.0	-50	600	810	0.0	0.0	0.0	0.0	126.60
		CPX-C	1,053	10,240	62,169	0	0	12.4	0.0	-50	170	295	0.0	0.0	0.0	0.0	50.34
A3	20	CPX	1,053	10,240	62,169	0	0	11.5	0.0	-74	70	1,042	0.0	0.0	0.0	0.0	88.11
		CPX-C	1,053	10,240	62,169	0	0	13.4	0.0	-74	376	728	0.0	0.0	0.0	0.0	93.76
A3	21	CPX	1,053	10,240	62,169	0	0	8.0	0.0	-97	530	1,002	0.0	0.0	0.0	0.0	137.29
		CPX-C	1,053	10,240	62,169	0	0	8.9	0.0	-97	295	558	0.0	0.0	0.0	0.0	63.66
A3	22	CPX	1,059	10,241	104,275	0	0	25.6	0.0	-36	1,230	2,589	0.0	0.0	0.0	0.0	627.91
		CPX-C	1,059	10,241	104,275	0	0	28.9	0.0	-36	1,369	2,143	0.0	0.0	0.0	0.0	402.82
A3	23	CPX	1,059	10,241	104,275	0	0	15.0	0.0	-59	1,740	4,096	0.0	0.0	0.0	0.0	684.62
		CPX-C	1,059	10,241	104,275	0	0	16.6	0.0	-59	1,449	3,017	0.0	0.0	0.0	0.0	506.50
A3	24	CPX	1,059	10,241	104,275	0	0	11.1	0.0	-79	3,770	8,224	0.0	0.0	0.0	0.0	1,162.27
		CPX-C	1,059	10,241	104,275	0	0	11.7	0.0	-79	3,570	5,737	0.0	0.0	0.0	0.0	828.38
A3	25	CPX	1,251	11,990	72,932	0	0	7.7	0.0	-46	30	159	0.0	0.0	0.0	0.0	42.55
		CPX-C	1,251	11,990	72,932	0	0	9.6	0.0	-46	660	688	0.0	0.0	0.0	0.0	138.20
A3	26	CPX	1,251	11,990	72,932	0	0	8.8	0.0	-65	970	1,270	0.0	0.0	0.0	0.0	188.79
		CPX-C	1,251	11,990	72,932	0	0	9.7	0.0	-65	1,110	1,561	0.0	0.0	0.0	0.0	219.94
A3	27	CPX	1,251	11,990	72,932	0	0	5.2	0.0	-86	10	200	0.0	0.0	0.0	0.0	37.88
		CPX-C	1,251	11,990	72,932	0	0	6.1	0.0	-86	690	807	0.0	0.0	0.0	0.0	129.66

◇ Problem is infeasible

Table 21 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	28	CPX	1,257	11,991	122,393	0	0	24.4	0.0	-35	1,520	2,177	0.0	0.0	0.0	0.0	797.58
		CPX-C	1,257	11,991	122,393	0	0	27.0	0.0	-35	440	1,022	0.0	0.0	0.0	0.0	333.83
A3	29	CPX	1,257	11,991	122,393	0	0	16.9	0.0	-55	5,759	7,222	0.0	0.0	0.0	0.0	1,308.75
		CPX-C	1,257	11,991	122,393	0	0	18.0	0.0	-55	1,658	4,373	0.0	0.0	0.0	0.0	818.66
A3	30	CPX	1,257	11,991	122,393	0	0	11.9	0.0	-74	7,380	10,058	0.0	0.0	0.0	0.0	1,651.54
		CPX-C	1,257	11,991	122,393	0	0	12.9	0.0	-74	2,140	4,251	0.0	0.0	0.0	0.0	775.38
A3	31	CPX	1,288	12,502	76,009	0	0	15.0	0.0	-45	0	408	0.0	0.0	0.0	0.0	68.24
		CPX-C	1,288	12,502	76,009	0	0	18.7	0.0	-45	200	448	0.0	0.0	0.0	0.0	157.79
A3	32	CPX	1,288	12,502	76,009	0	0	6.1	0.0	-68	286	326	0.0	0.0	0.0	0.0	69.55
		CPX-C	1,288	12,502	76,009	0	0	6.8	0.0	-68	35	118	0.0	0.0	0.0	0.0	35.15
A3	33	CPX	1,288	12,502	76,009	0	0	5.3	0.0	-86	630	800	0.0	0.0	0.0	0.0	116.31
		CPX-C	1,288	12,502	76,009	0	0	5.8	0.0	-86	140	346	0.0	0.0	0.0	0.0	72.50
A3	34	CPX	1,293	12,502	127,555	0	0	25.3	0.0	-36	1,010	3,421	0.0	0.0	0.0	0.0	896.02
		CPX-C	1,293	12,502	127,555	0	0	28.0	0.0	-36	1,772	2,091	0.0	0.0	0.0	0.0	673.14
A3	35	CPX	1,293	12,502	127,555	0	0	16.6	0.0	-57	1,840	9,172	0.0	0.0	0.0	0.0	1,828.66
		CPX-C	1,293	12,502	127,555	0	0	18.2	0.0	-57	1,200	7,167	0.0	0.0	0.0	0.0	1,318.47
A3	36	CPX	1,293	12,502	127,555	0	0	10.4	0.0	-77	3,150	9,831	0.0	0.0	0.0	0.0	1,971.54
		CPX-C	1,293	12,502	127,555	0	0	10.9	0.0	-77	210	4,062	0.0	0.0	0.0	0.0	680.50
C1	1	CPX	101	938	2,695	0	0	0.0	0.0	131	0	0	0.0	0.0	0.0	0.0	0.02
		CPX-C	101	938	2,695	0	0	32.0	0.0	131	2	2	0.0	0.0	0.0	0.0	0.02
C1	2	CPX	101	930	2,671	0	0	0.0	0.0	131	0	0	0.0	0.0	0.0	0.0	0.02
		CPX-C	101	930	2,671	0	0	25.2	0.0	131	2	2	0.0	0.0	0.0	0.0	0.02
C1	5	CPX	110	966	10,568	0	0	0.0	0.0	100	0	0	0.0	0.0	0.0	0.0	0.04
		CPX-C	110	966	10,568	0	0	16.1	0.0	100	0	2	0.0	0.0	0.0	0.0	0.05
C1	6	CPX	110	957	10,458	0	0	0.0	0.0	100	0	0	0.0	0.0	0.0	0.0	0.04
		CPX-C	110	957	10,458	0	0	11.4	0.0	100	0	1	0.0	0.0	0.0	0.0	0.05
C1	9	CPX	200	1,359	3,851	0	0	0.0	0.0	420	0	0	0.0	0.0	0.0	0.0	0.03
		CPX-C	200	1,359	3,851	0	0	15.1	0.0	420	0	0	0.0	0.0	0.0	0.0	0.03
C1	10	CPX	200	1,284	3,631	0	0	0.0	0.0	420	0	0	0.0	0.0	0.0	0.0	0.02
		CPX-C	200	1,284	3,631	0	0	0.0	0.0	420	0	0	0.0	0.0	0.0	0.0	0.02
C1	13	CPX	210	1,799	20,631	0	0	4.3	0.0	448	0	1	0.0	0.0	0.0	0.0	0.22
		CPX-C	210	1,799	20,631	0	0	34.7	0.0	448	17	22	0.0	0.0	0.0	0.0	0.38
C1	14	CPX	210	1,739	19,897	0	0	◇	◇	◇	◇	8	0.0	0.0	0.0	0.0	0.48
		CPX-C	210	1,739	19,897	0	0	◇	◇	◇	◇	34	0.0	0.0	0.0	0.0	0.51

◇ Problem is infeasible

Table 21 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C1	17	CPX	501	4,838	13,976	0	0	0.0	0.0	652	0	0	0.0	0.0	0.0	0.0	0.13
		CPX-C	501	4,838	13,976	0	0	25.1	0.0	652	2	2	0.0	0.0	0.0	0.0	0.14
C1	18	CPX	501	4,836	13,970	0	0	0.0	0.0	652	0	0	0.0	0.0	0.0	0.0	0.12
		CPX-C	501	4,836	13,970	0	0	19.9	0.0	652	0	2	0.0	0.0	0.0	0.0	0.13
C1	21	CPX	510	4,414	52,065	0	0	0.0	0.0	858	0	0	0.0	0.0	0.0	0.0	0.20
		CPX-C	510	4,414	52,065	0	0	20.9	0.0	858	0	0	0.0	0.0	0.0	0.0	0.23
C1	22	CPX	510	4,292	50,608	0	0	0.0	0.0	858	0	0	0.0	0.0	0.0	0.0	0.20
		CPX-C	510	4,292	50,608	0	0	10.5	0.0	858	0	0	0.0	0.0	0.0	0.0	0.21
C2	1	CPX	101	938	2,695	0	989	5.6	0.0	-2,119	11,033	11,033	0.0	0.6	0.0	0.0	26.43
		CPX-C	101	938	2,695	0	518	6.4	0.0	-2,119	4,790	7,127	0.0	0.3	0.0	0.0	9.86
C2	2	CPX	101	930	2,671	0	635	7.2	0.0	-1,868	4,686	7,379	0.0	0.4	0.0	0.0	11.14
		CPX-C	101	930	2,671	0	763	8.1	0.0	-1,868	3,295	6,684	0.0	0.3	0.0	0.0	12.11
C2	5	CPX	110	966	10,568	0	91	12.7	0.0	-1,515	1,556	2,537	0.0	0.1	0.0	0.0	4.50
		CPX-C	110	966	10,568	0	135	15.2	0.0	-1,515	3,415	3,585	0.0	0.2	0.0	0.0	6.82
C2	6	CPX	110	957	10,458	0	134	16.5	0.0	-1,279	2,250	4,804	0.0	0.2	0.0	0.0	8.05
		CPX-C	110	957	10,458	0	151	19.3	0.0	-1,279	2,760	5,066	0.0	0.2	0.0	0.0	7.75
C2	9	CPX	200	1,359	3,851	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.03
		CPX-C	200	1,359	3,851	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.03
C2	10	CPX	200	1,284	3,631	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.02
		CPX-C	200	1,284	3,631	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.02
C2	13	CPX	210	1,799	20,631	0	0	70.0	0.0	-593	30	58	0.0	0.0	0.0	0.0	1.02
		CPX-C	210	1,799	20,631	0	0	85.3	0.0	-593	27	51	0.0	0.0	0.0	0.0	0.64
C2	14	CPX	210	1,739	19,897	0	0	◇	◇	◇	◇	14	0.0	0.0	0.0	0.0	0.65
		CPX-C	210	1,739	19,897	0	0	◇	◇	◇	◇	12	0.0	0.0	0.0	0.0	0.43
C2	17	CPX	501	4,838	13,976	0	108	0.8	0.0	-40,731	1,899	2,937	0.0	0.8	0.0	0.0	23.14
		CPX-C	501	4,838	13,976	0	238	1.1	0.0	-40,731	4,010	4,663	0.0	1.4	0.0	0.0	33.73
C2	18	CPX	501	4,836	13,970	0	153	0.9	0.0	-36,549	2,104	2,440	0.0	0.6	0.0	0.0	11.12
		CPX-C	501	4,836	13,970	0	114	1.1	0.0	-36,549	1,223	1,749	0.0	0.4	0.0	0.0	8.10
C2	21	CPX	510	4,414	52,065	0	8	21.4	0.0	-4,221	403	577	0.0	0.1	0.0	0.0	10.32
		CPX-C	510	4,414	52,065	0	14	27.3	0.0	-4,221	3,060	3,201	0.0	1.0	0.0	0.0	49.41
C2	22	CPX	510	4,292	50,608	0	1	16.0	0.0	-3,657	0	64	0.0	0.0	0.0	0.0	2.70
		CPX-C	510	4,292	50,608	0	0	25.0	0.0	-3,657	0	38	0.0	0.0	0.0	0.0	1.32
C3	1	CPX	505	4,979	30,232	0	5	24.3	0.0	-43	120	772	0.0	0.1	0.0	0.0	22.39
		CPX-C	505	4,979	30,232	0	7	29.5	0.0	-43	470	821	0.0	0.2	0.0	0.0	24.98

◇ Problem is infeasible

Table 21 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C3	2	CPX	505	4,979	30,232	0	39	11.0	0.0	-81	4,860	7,919	0.0	2.2	0.0	0.0	144.38
		CPX-C	505	4,979	30,232	0	46	12.7	0.0	-81	6,669	8,156	0.0	2.3	0.0	0.0	159.84
C3	3	CPX	505	4,979	30,232	0	221	7.6	0.0	-114	33,526	33,526	0.0	7.8	0.0	0.0	485.13
		CPX-C	505	4,979	30,232	0	182	9.7	0.0	-114	22,394	26,763	0.0	7.2	0.0	0.0	440.55
C3	4	CPX	510	4,979	50,616	0	0	◇	◇	◇	◇	243	0.0	0.1	0.0	0.0	40.42
		CPX-C	510	4,979	50,616	0	0	◇	◇	◇	◇	144	0.0	0.0	0.0	0.0	25.39
C3	5	CPX	510	4,979	50,616	0	39	18.7	0.0	-59	3,660	10,231	0.0	2.5	0.0	0.0	315.38
		CPX-C	510	4,979	50,616	0	28	20.5	0.0	-59	4,228	7,836	0.0	1.7	0.0	0.0	234.60
C3	6	CPX	510	4,979	50,616	0	76	10.6	0.0	-88	1,000	19,426	0.0	3.8	0.0	0.0	499.15
		CPX-C	510	4,979	50,616	0	46	11.8	0.0	-88	980	11,764	0.0	2.6	0.0	0.0	286.89
C3	7	CPX	505	4,974	30,175	0	2	17.7	0.0	-41	120	401	0.0	0.1	0.0	0.0	12.22
		CPX-C	505	4,974	30,175	0	1	22.8	0.0	-41	183	251	0.0	0.1	0.0	0.0	10.48
C3	8	CPX	505	4,974	30,175	0	38	9.1	0.0	-82	2,270	3,679	0.0	0.9	0.0	0.0	61.99
		CPX-C	505	4,974	30,175	0	25	10.1	0.0	-82	20	1,217	0.0	0.3	0.0	0.0	21.04
C3	9	CPX	505	4,974	30,175	0	56	6.4	0.0	-119	4,630	6,024	0.0	1.7	0.0	0.0	100.80
		CPX-C	505	4,974	30,175	0	82	7.3	0.0	-119	7,528	8,844	0.0	2.6	0.0	0.0	165.04
C3	10	CPX	510	4,974	50,506	0	0	◇	◇	◇	◇	180	0.0	0.0	0.0	0.0	30.20
		CPX-C	510	4,974	50,506	0	0	◇	◇	◇	◇	52	0.0	0.0	0.0	0.0	12.06
C3	11	CPX	510	4,974	50,506	0	28	16.1	0.0	-56	2,000	8,536	0.0	1.6	0.0	0.0	255.37
		CPX-C	510	4,974	50,506	0	61	18.0	0.0	-56	6,870	11,089	0.0	3.2	0.0	0.0	371.16
C3	12	CPX	510	4,974	50,506	0	161	10.9	0.0	-83	40,230	61,793	0.0	14.9	0.0	0.0	1,506.34
		CPX-C	510	4,974	50,506	0	86	12.0	0.0	-83	9,480	26,919	0.0	6.2	0.0	0.0	686.14
C3	13	CPX	754	7,531	45,910	0	0	11.6	0.0	-35	0	22	0.0	0.0	0.0	0.0	2.77
		CPX-C	754	7,531	45,910	0	0	18.5	0.0	-35	0	24	0.0	0.0	0.0	0.0	1.69
C3	14	CPX	754	7,531	45,910	0	12	9.4	0.0	-73	0	1,458	0.0	0.4	0.0	0.0	72.12
		CPX-C	754	7,531	45,910	0	12	10.1	0.0	-73	150	891	0.0	0.3	0.0	0.0	52.49
C3	15	CPX	754	7,531	45,910	0	100	7.7	0.0	-105	14,718	20,283	0.0	7.8	0.0	0.0	593.39
		CPX-C	754	7,531	45,910	0	84	8.3	0.0	-105	8,601	12,198	0.0	5.6	0.0	0.0	419.61
C3	16	CPX	759	7,531	76,763	0	0	◇	◇	◇	◇	0	0.0	0.0	0.0	0.0	3.61
		CPX-C	759	7,531	76,763	0	0	◇	◇	◇	◇	2	0.0	0.0	0.0	0.0	5.10
C3	17	CPX	759	7,531	76,763	0	28	19.1	0.0	-57	8,258	13,575	0.0	4.4	0.0	0.0	989.98
		CPX-C	759	7,531	76,763	0	14	21.5	0.0	-57	1,240	4,833	0.0	1.5	0.0	0.0	347.68
C3	18	CPX	759	7,531	76,763	0	111	13.1	0.0	-85	8,740	41,179	0.0	12.1	0.0	0.0	2,414.82
		CPX-C	759	7,531	76,763	0	92	14.8	0.0	-85	8,160	33,890	0.0	11.3	0.0	0.0	1,918.42

◇ Problem is infeasible

Table 21 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C3	19	CPX	755	7,602	46,323	0	10	34.4	0.0	-38	1,494	2,023	0.0	0.6	0.0	0.0	73.84
		CPX-C	755	7,602	46,323	0	10	34.4	0.0	-38	1,494	2,023	0.0	0.6	0.0	0.0	84.56
C3	20	CPX	755	7,602	46,323	0	15	12.7	0.0	-80	20	1,190	0.0	0.3	0.0	0.0	39.41
		CPX-C	755	7,602	46,323	0	15	12.7	0.0	-80	20	1,190	0.0	0.4	0.0	0.0	45.30
C3	21	CPX	755	7,602	46,323	0	68	8.0	0.0	-117	6,380	8,604	0.0	3.6	0.0	0.0	262.19
		CPX-C	755	7,602	46,323	0	68	8.0	0.0	-117	6,380	8,604	0.0	4.2	0.0	0.0	302.41
C3	22	CPX	760	7,602	77,431	0	0	◇	◇	◇	◇	180	0.0	0.1	0.0	0.0	72.12
		CPX-C	760	7,602	77,431	0	0	◇	◇	◇	◇	180	0.0	0.1	0.0	0.0	80.01
C3	23	CPX	760	7,602	77,431	0	8	11.2	0.0	-62	1,865	1,865	0.0	0.8	0.0	0.0	124.09
		CPX-C	760	7,602	77,431	0	8	11.2	0.0	-62	1,865	1,865	0.0	0.8	0.0	0.0	141.10
C3	24	CPX	760	7,602	77,431	0	113	15.2	0.0	-85	17,470	56,692	0.0	16.8	0.0	0.0	2,574.45
		CPX-C	760	7,602	77,431	0	131	15.2	0.0	-85	9,960	59,378	0.0	21.0	0.0	0.0	2,900.50
C3	25	CPX	1,003	9,965	60,808	0	0	◇	◇	◇	◇	50	0.0	0.0	0.0	0.0	41.48
		CPX-C	1,003	9,965	60,808	0	0	◇	◇	◇	◇	44	0.0	0.0	0.0	0.0	28.91
C3	26	CPX	1,003	9,965	60,808	0	13	14.1	0.0	-69	1,790	4,298	0.0	1.6	0.0	0.0	361.14
		CPX-C	1,003	9,965	60,808	0	12	16.5	0.0	-69	2,530	3,105	0.0	1.3	0.0	0.0	211.99
C3	27	CPX	1,003	9,965	60,808	0	160	9.6	0.0	-106	16,477	22,486	0.0	10.6	0.0	0.0	2,251.47
		CPX-C	1,003	9,965	60,808	0	104	11.1	0.0	-106	12,550	18,425	0.0	11.7	0.0	0.0	1,080.13
C3	28	CPX	1,008	9,965	101,905	0	0	◇	◇	◇	◇	4	0.0	0.0	0.0	0.0	25.43
		CPX-C	1,008	9,965	101,905	0	0	◇	◇	◇	◇	24	0.0	0.0	0.0	0.0	31.67
C3	29	CPX	1,008	9,965	101,905	0	32	24.9	0.0	-49	3,280	15,963	0.0	6.2	0.0	0.0	2,436.76
		CPX-C	1,008	9,965	101,905	0	17	26.8	0.0	-49	580	8,389	0.0	3.5	0.0	0.0	1,213.62
C3	30	CPX	1,008	9,965	101,905	0	82	16.5	8.7	-77	17,620	41,838	0.0	27.2	0.0	0.0	3,600.00
		CPX-C	1,008	9,965	101,905	0	75	17.4	7.4	-77	23,360	34,671	0.0	24.6	0.0	0.0	3,600.00
C3	31	CPX	1,003	10,105	61,540	0	1	25.4	0.0	-40	174	271	0.0	0.1	0.0	0.0	29.12
		CPX-C	1,003	10,105	61,540	0	1	25.4	0.0	-40	174	271	0.0	0.1	0.0	0.0	32.61
C3	32	CPX	1,003	10,105	61,540	0	17	13.5	0.0	-75	20	4,568	0.0	1.7	0.0	0.0	182.58
		CPX-C	1,003	10,105	61,540	0	17	13.5	0.0	-75	20	4,568	0.0	2.0	0.0	0.0	211.57
C3	33	CPX	1,003	10,105	61,540	0	60	7.7	0.0	-111	13,150	17,221	0.0	10.1	0.0	0.0	719.14
		CPX-C	1,003	10,105	61,540	0	60	7.7	0.0	-111	13,150	17,221	0.0	11.5	0.0	0.0	831.95
C3	34	CPX	1,008	10,105	102,816	0	0	◇	◇	◇	◇	212	0.0	0.1	0.0	0.0	144.25
		CPX-C	1,008	10,105	102,816	0	0	◇	◇	◇	◇	212	0.0	0.1	0.0	0.0	145.82
C3	35	CPX	1,008	10,105	102,816	0	46	17.9	0.0	-59	8,980	11,129	0.0	7.0	0.0	0.0	1,002.09
		CPX-C	1,008	10,105	102,816	0	46	17.9	0.0	-59	8,980	11,129	0.0	8.0	0.0	0.0	1,114.56

◇ Problem is infeasible

Table 21 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C3	36	CPX	1,008	10,105	102,816	0	168	15.6	6.1	-84	12,940	32,772	0.0	19.7	0.0	0.0	3,600.00
		CPX-C	1,008	10,105	102,816	0	153	15.6	6.6	-84	12,940	27,965	0.0	19.9	0.0	0.0	3,600.00

◇ Problem is infeasible

Table 22: RCSPP preprocessing and probing results

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A1	3	DB	0	0	0	0	0	0.0	0.0	2	0	0	0.0	0.0	0.0	0.0	0.00
		DB+GQ	0	0	0	0	0	0.0	0.0	2	0	0	0.0	0.0	0.0	0.0	0.00
A1	4	DB	0	0	0	0	0	0.0	0.0	2	0	0	0.0	0.0	0.0	0.0	0.00
		DB+GQ	0	0	0	0	0	0.0	0.0	2	0	0	0.0	0.0	0.0	0.0	0.00
A1	7	DB	0	0	0	0	0	0.0	0.0	6	0	0	0.0	0.0	0.0	0.0	0.00
		DB+GQ	0	0	0	0	0	0.0	0.0	6	0	0	0.0	0.0	0.0	0.0	0.00
A1	8	DB	99	706	7,295	0	0	61.6	0.0	14	19	87	0.0	0.0	0.0	0.0	0.36
		DB+GQ	49	156	1,465	42	0	0.0	0.0	14	0	0	0.0	0.0	0.0	0.0	0.08
A1	11	DB	0	0	0	0	0	0.0	0.0	6	0	0	0.0	0.0	0.0	0.0	0.00
		DB+GQ	0	0	0	0	0	0.0	0.0	6	0	0	0.0	0.0	0.0	0.0	0.00
A1	12	DB	0	0	0	0	0	0.0	0.0	6	0	0	0.0	0.0	0.0	0.0	0.00
		DB+GQ	0	0	0	0	0	0.0	0.0	6	0	0	0.0	0.0	0.0	0.0	0.00
A1	15	DB	0	0	0	0	0	0.0	0.0	9	0	0	0.0	0.0	0.0	0.0	0.01
		DB+GQ	0	0	0	0	0	0.0	0.0	9	0	0	0.0	0.0	0.0	0.0	0.01
A1	16	DB	185	1,597	16,797	0	0	47.1	0.0	17	95	116	0.0	0.0	0.0	0.0	2.08
		DB+GQ	0	0	0	0	0	0.0	0.0	17	0	0	0.1	0.0	0.0	0.0	0.06
A1	19	DB	0	0	0	0	0	0.0	0.0	6	0	0	0.0	0.0	0.0	0.0	0.01
		DB+GQ	0	0	0	0	0	0.0	0.0	6	0	0	0.0	0.0	0.0	0.0	0.01
A1	20	DB	0	0	0	0	0	0.0	0.0	6	0	0	0.0	0.0	0.0	0.0	0.01
		DB+GQ	0	0	0	0	0	0.0	0.0	6	0	0	0.0	0.0	0.0	0.0	0.00
A1	23	DB	0	0	0	0	0	0.0	0.0	4	0	0	0.0	0.0	0.0	0.0	0.02
		DB+GQ	0	0	0	0	0	0.0	0.0	4	0	0	0.0	0.0	0.0	0.0	0.03
A1	24	DB	0	0	0	0	0	0.0	0.0	5	0	0	0.0	0.0	0.0	0.0	0.02
		DB+GQ	0	0	0	0	0	0.0	0.0	5	0	0	0.0	0.0	0.0	0.0	0.03
A2	3	DB	97	933	2,651	0	0	1.0	0.0	-78	3	3	0.0	0.0	0.0	0.0	0.03
		DB+GQ	97	933	2,651	0	0	1.0	0.0	-78	3	3	0.0	0.0	0.0	0.0	0.05
A2	4	DB	96	924	2,626	0	0	0.3	0.0	-74	0	0	0.0	0.0	0.0	0.0	0.02
		DB+GQ	96	924	2,626	0	0	0.3	0.0	-74	0	0	0.0	0.0	0.0	0.0	0.04
A2	7	DB	100	839	8,804	0	0	12.6	0.0	-29	110	118	0.0	0.0	0.0	0.0	0.57
		DB+GQ	89	708	7,405	124	0	1.9	0.0	-29	4	4	0.3	0.0	0.0	0.0	0.44
A2	8	DB	99	706	7,295	0	0	22.3	0.0	-23	62	67	0.0	0.0	0.0	0.0	0.36
		DB+GQ	49	156	1,465	12	0	4.3	0.0	-23	0	0	0.1	0.0	0.0	0.0	0.07
A2	11	DB	192	1,911	5,462	0	0	0.5	0.0	-125	0	0	0.0	0.0	0.0	0.0	0.06
		DB+GQ	192	1,911	5,462	0	0	0.5	0.0	-125	0	0	0.0	0.0	0.0	0.0	0.10

◇ Problem is infeasible

Table 22 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A2	12	DB	192	1,911	5,462	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.05
		DB+GQ	192	1,911	5,462	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.08
A2	15	DB	189	1,700	17,958	0	0	38.4	0.0	-24	10	387	0.0	0.0	0.0	0.0	3.82
		DB+GQ	151	1,179	12,201	751	0	8.1	0.0	-24	1	2	0.5	0.0	0.0	0.0	1.10
A2	16	DB	185	1,597	16,797	0	0	69.5	0.0	-17	0	145	0.0	0.0	0.0	0.0	2.19
		DB+GQ	0	0	0	0	0	0.0	0.0	-17	0	0	0.1	0.0	0.0	0.0	0.07
A2	19	DB	469	4,677	13,365	0	0	0.0	0.0	-99	0	0	0.0	0.0	0.0	0.0	0.18
		DB+GQ	469	4,677	13,365	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.27
A2	20	DB	468	4,662	13,321	0	0	0.0	0.0	-92	0	0	0.0	0.0	0.0	0.0	0.16
		DB+GQ	468	4,662	13,321	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.25
A2	23	DB	479	4,559	49,360	0	0	25.4	0.0	-36	280	973	0.0	0.0	0.0	0.0	42.74
		DB+GQ	471	4,471	48,353	37	0	24.8	0.0	-36	300	852	3.5	0.0	0.0	0.0	49.77
A2	24	DB	475	4,485	48,017	0	0	31.7	0.0	-29	477	539	0.0	0.0	0.0	0.0	30.76
		DB+GQ	413	3,531	37,238	952	0	9.9	0.0	-29	9	12	2.5	0.0	0.0	0.0	7.01
A3	1	DB	865	8,370	50,935	0	0	14.5	0.0	-44	170	339	0.1	0.0	0.0	0.0	35.24
		DB+GQ	860	8,334	50,720	1	0	14.5	0.0	-44	161	439	2.1	0.0	0.0	0.0	26.67
A3	2	DB	876	8,588	52,287	0	0	11.8	0.0	-64	890	1,577	0.1	0.0	0.0	0.0	75.60
		DB+GQ	876	8,588	52,287	0	0	11.8	0.0	-64	890	1,577	1.7	0.0	0.0	0.0	79.45
A3	3	DB	876	8,596	52,337	0	0	6.7	0.0	-85	0	171	0.1	0.0	0.0	0.0	9.89
		DB+GQ	876	8,596	52,337	0	0	6.7	0.0	-85	0	171	1.2	0.0	0.0	0.0	11.26
A3	4	DB	866	8,339	84,719	0	0	31.6	0.0	-32	701	1,982	0.1	0.0	0.0	0.0	268.27
		DB+GQ	848	8,132	82,600	749	0	23.8	0.0	-32	250	573	15.3	0.0	0.0	0.0	123.59
A3	5	DB	881	8,588	87,286	0	0	16.2	0.0	-54	2,270	7,038	0.1	0.0	0.0	0.0	599.51
		DB+GQ	881	8,588	87,286	0	0	16.2	0.0	-54	2,270	7,038	4.5	0.0	0.0	0.0	613.38
A3	6	DB	881	8,596	87,365	0	0	12.4	0.0	-73	730	5,647	0.1	0.0	0.0	0.0	384.27
		DB+GQ	881	8,596	87,365	0	0	12.4	0.0	-73	730	5,647	2.9	0.0	0.0	0.0	395.49
A3	7	DB	878	8,588	52,241	0	0	18.9	0.0	-44	120	507	0.1	0.0	0.0	0.0	38.59
		DB+GQ	876	8,570	52,132	0	0	18.9	0.0	-44	0	534	2.1	0.0	0.0	0.0	30.80
A3	8	DB	882	8,711	53,009	0	0	8.4	0.0	-70	280	473	0.1	0.0	0.0	0.0	55.85
		DB+GQ	882	8,711	53,009	0	0	8.4	0.0	-70	280	473	1.3	0.0	0.0	0.0	59.23
A3	9	DB	882	8,711	53,009	0	0	6.0	0.0	-89	328	815	0.1	0.0	0.0	0.0	41.37
		DB+GQ	882	8,711	53,009	0	0	6.0	0.0	-89	328	815	1.2	0.0	0.0	0.0	43.33
A3	10	DB	883	8,578	87,123	0	0	40.6	0.0	-33	400	4,888	0.1	0.0	0.0	0.0	580.31
		DB+GQ	863	8,383	85,155	30	0	40.1	0.0	-33	2,160	5,693	7.7	0.0	0.0	0.0	593.81

◇ Problem is infeasible

Table 22 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	11	DB	888	8,712	88,550	0	0	15.1	0.0	-60	5,282	6,472	0.1	0.0	0.0	0.0	445.52
		DB+GQ	888	8,712	88,550	0	0	15.1	0.0	-60	5,282	6,472	3.8	0.0	0.0	0.0	462.23
A3	12	DB	888	8,712	88,550	0	0	11.8	0.0	-79	15,675	15,675	0.1	0.0	0.0	0.0	958.48
		DB+GQ	888	8,712	88,550	0	0	11.8	0.0	-79	15,675	15,675	2.5	0.0	0.0	0.0	971.34
A3	13	DB	1,091	10,409	63,117	0	0	20.3	0.0	-42	327	526	0.1	0.0	0.0	0.0	69.52
		DB+GQ	1,082	10,309	62,509	56	0	19.4	0.0	-42	530	575	4.2	0.0	0.0	0.0	67.19
A3	14	DB	1,097	10,670	64,763	0	0	10.0	0.0	-67	240	467	0.1	0.0	0.0	0.0	53.02
		DB+GQ	1,097	10,670	64,763	0	0	10.0	0.0	-67	240	467	2.0	0.0	0.0	0.0	55.00
A3	15	DB	1,097	10,670	64,763	0	0	5.9	0.0	-89	120	157	0.1	0.0	0.0	0.0	38.19
		DB+GQ	1,097	10,670	64,763	0	0	5.9	0.0	-89	120	157	1.4	0.0	0.0	0.0	39.20
A3	16	DB	1,094	10,354	105,082	0	0	37.6	0.0	-32	324	1,848	0.1	0.0	0.0	0.0	299.03
		DB+GQ	1,032	9,719	98,586	7,358	0	0.0	0.0	-32	0	0	17.8	0.0	0.0	0.0	52.08
A3	17	DB	1,103	10,671	108,673	0	0	22.3	0.0	-55	3,205	8,742	0.1	0.0	0.0	0.0	1,035.13
		DB+GQ	1,103	10,671	108,673	0	0	22.3	0.0	-55	3,205	8,742	4.8	0.0	0.0	0.0	1,073.19
A3	18	DB	1,103	10,671	108,673	0	0	11.3	0.0	-78	980	3,129	0.1	0.0	0.0	0.0	477.82
		DB+GQ	1,103	10,671	108,673	0	0	11.3	0.0	-78	980	3,129	4.2	0.0	0.0	0.0	493.94
A3	19	DB	1,053	10,207	61,942	0	0	12.4	0.0	-50	440	504	0.1	0.0	0.0	0.0	89.14
		DB+GQ	1,051	10,186	61,815	0	0	12.4	0.0	-50	440	487	3.0	0.0	0.0	0.0	89.65
A3	20	DB	1,053	10,240	62,169	0	0	13.4	0.0	-74	376	728	0.1	0.0	0.0	0.0	78.63
		DB+GQ	1,053	10,240	62,169	0	0	13.4	0.0	-74	376	728	1.6	0.0	0.0	0.0	79.82
A3	21	DB	1,053	10,240	62,169	0	0	8.9	0.0	-97	295	558	0.1	0.0	0.0	0.0	54.06
		DB+GQ	1,053	10,240	62,169	0	0	8.9	0.0	-97	295	558	1.3	0.0	0.0	0.0	56.24
A3	22	DB	1,057	10,158	103,166	0	0	28.9	0.0	-36	590	1,381	0.1	0.0	0.0	0.0	306.16
		DB+GQ	1,036	9,933	100,845	1,627	0	21.7	0.0	-36	109	192	29.4	0.0	0.0	0.0	93.58
A3	23	DB	1,059	10,241	104,275	0	0	16.6	0.0	-59	1,449	3,017	0.1	0.0	0.0	0.0	441.57
		DB+GQ	1,059	10,241	104,275	0	0	16.6	0.0	-59	1,449	3,017	4.9	0.0	0.0	0.0	456.90
A3	24	DB	1,059	10,241	104,275	0	0	11.7	0.0	-79	3,570	5,737	0.1	0.0	0.0	0.0	726.61
		DB+GQ	1,059	10,241	104,275	0	0	11.7	0.0	-79	3,570	5,737	3.7	0.0	0.0	0.0	742.06
A3	25	DB	1,190	11,017	66,855	0	0	9.6	0.0	-46	1,014	1,014	0.2	0.0	0.0	0.0	183.94
		DB+GQ	1,179	10,909	66,187	4	0	9.6	0.0	-46	620	673	3.4	0.0	0.0	0.0	208.69
A3	26	DB	1,249	11,946	72,640	0	0	9.7	0.0	-65	640	893	0.1	0.0	0.0	0.0	113.53
		DB+GQ	1,249	11,946	72,640	0	0	9.7	0.0	-65	640	893	2.9	0.0	0.0	0.0	119.44
A3	27	DB	1,251	11,990	72,932	0	0	6.1	0.0	-86	690	807	0.1	0.0	0.0	0.0	108.96
		DB+GQ	1,251	11,990	72,932	0	0	6.1	0.0	-86	690	807	2.0	0.0	0.0	0.0	114.28

◇ Problem is infeasible

Table 22 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	28	DB	1,175	10,744	109,106	0	0	27.0	0.0	-35	130	715	0.1	0.0	0.0	0.0	192.75
		DB+GQ	1,102	10,017	101,533	421	0	22.9	0.0	-35	580	802	13.4	0.0	0.0	0.0	267.75
A3	29	DB	1,254	11,931	121,785	0	0	18.0	0.0	-55	1,520	3,443	0.1	0.0	0.0	0.0	695.44
		DB+GQ	1,254	11,931	121,785	1	0	18.0	0.0	-55	1,570	3,914	8.4	0.0	0.0	0.0	564.36
A3	30	DB	1,257	11,991	122,393	0	0	12.9	0.0	-74	2,140	4,251	0.1	0.0	0.0	0.0	676.13
		DB+GQ	1,257	11,991	122,393	0	0	12.9	0.0	-74	2,140	4,251	6.4	0.0	0.0	0.0	672.76
A3	31	DB	1,278	12,222	74,256	0	0	18.7	0.0	-45	40	302	0.1	0.0	0.0	0.0	58.73
		DB+GQ	1,274	12,180	74,012	3	0	18.7	0.0	-45	20	345	3.4	0.0	0.0	0.0	53.70
A3	32	DB	1,288	12,501	76,002	0	0	6.8	0.0	-68	310	380	0.1	0.0	0.0	0.0	94.16
		DB+GQ	1,288	12,501	76,002	0	0	6.8	0.0	-68	310	380	3.0	0.0	0.0	0.0	99.18
A3	33	DB	1,288	12,502	76,009	0	0	5.8	0.0	-86	140	346	0.1	0.0	0.0	0.0	60.62
		DB+GQ	1,288	12,502	76,009	0	0	5.8	0.0	-86	140	346	2.5	0.0	0.0	0.0	64.95
A3	34	DB	1,274	12,097	122,940	0	0	28.0	0.0	-36	3,190	3,888	0.1	0.0	0.0	0.0	975.13
		DB+GQ	1,253	11,870	120,577	212	0	23.9	0.0	-36	30	863	16.6	0.0	0.0	0.0	232.73
A3	35	DB	1,293	12,498	127,513	0	0	18.2	0.0	-57	830	5,671	0.1	0.0	0.0	0.0	979.68
		DB+GQ	1,293	12,498	127,513	0	0	18.2	0.0	-57	830	5,671	6.3	0.0	0.0	0.0	989.06
A3	36	DB	1,293	12,502	127,555	0	0	10.9	0.0	-77	210	4,062	0.1	0.0	0.0	0.0	581.04
		DB+GQ	1,293	12,502	127,555	0	0	10.9	0.0	-77	210	4,062	5.5	0.0	0.0	0.0	586.31
C1	1	DB	0	0	0	0	0	0.0	0.0	131	0	0	0.0	0.0	0.0	0.0	0.00
C1	2	DB	0	0	0	0	0	0.0	0.0	131	0	0	0.0	0.0	0.0	0.0	0.00
C1	5	DB	0	0	0	0	0	0.0	0.0	100	0	0	0.0	0.0	0.0	0.0	0.00
C1	6	DB	0	0	0	0	0	0.0	0.0	100	0	0	0.0	0.0	0.0	0.0	0.00
C1	9	DB	0	0	0	0	0	0.0	0.0	420	0	0	0.0	0.0	0.0	0.0	0.00
C1	10	DB	0	0	0	0	0	0.0	0.0	420	0	0	0.0	0.0	0.0	0.0	0.00
C1	13	DB	0	0	0	0	0	0.0	0.0	448	0	0	0.0	0.0	0.0	0.0	0.00
C1	14	DB	39	56	474	0	0	◇	◇	◇	◇	0	0.0	0.0	0.0	0.0	0.01
C1	17	DB	0	0	0	0	0	0.0	0.0	652	0	0	0.0	0.0	0.0	0.0	0.01
C1	18	DB	0	0	0	0	0	0.0	0.0	652	0	0	0.0	0.0	0.0	0.0	0.00
C1	21	DB	0	0	0	0	0	0.0	0.0	858	0	0	0.0	0.0	0.0	0.0	0.01
C1	22	DB	0	0	0	0	0	0.0	0.0	858	0	0	0.0	0.0	0.0	0.0	0.01
C2	1	AAN	101	898	2,571	0	694	6.4	0.0	-2,119	6,658	8,380	0.0	0.4	0.0	0.0	15.20
		AAN+GQ	101	898	2,571	0	694	6.4	0.0	-2,119	6,658	8,380	0.0	0.4	0.0	0.0	15.47

◇ Problem is infeasible

Table 22 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C2	2	AAN	100	878	2,513	0	665	8.1	0.0	-1,868	5,972	6,164	0.0	0.3	0.0	0.0	9.52
		AAN+GQ	100	878	2,513	0	665	8.1	0.0	-1,868	5,972	6,164	0.0	0.3	0.0	0.0	9.72
C2	5	AAN	110	907	9,860	0	117	15.2	0.0	-1,515	4,076	4,668	0.0	0.2	0.0	0.0	7.62
		AAN+GQ	110	907	9,860	0	117	15.2	0.0	-1,515	4,076	4,668	0.3	0.2	0.0	0.0	7.99
C2	6	AAN	110	868	9,367	0	124	19.3	0.0	-1,279	2,410	4,931	0.0	0.2	0.0	0.0	7.04
		AAN+GQ	110	868	9,367	0	124	19.3	0.0	-1,279	2,410	4,931	0.2	0.2	0.0	0.0	7.37
C2	9	AAN	0	0	0	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		AAN+GQ	0	0	0	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
C2	10	AAN	0	0	0	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		AAN+GQ	0	0	0	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
C2	13	AAN	0	0	0	0	0	0.0	0.0	-593	0	0	0.0	0.0	0.0	0.0	0.01
		AAN+GQ	0	0	0	0	0	0.0	0.0	-593	0	0	0.0	0.0	0.0	0.0	0.01
C2	14	AAN	39	56	474	0	0	◇	◇	◇	◇	0	0.0	0.0	0.0	0.0	0.01
		AAN+GQ	39	56	474	0	0	◇	◇	◇	◇	0	0.0	0.0	0.0	0.0	0.01
C2	17	AAN	501	4,812	13,893	0	143	1.1	0.0	-40,731	1,357	1,944	0.0	0.5	0.0	0.0	14.69
		AAN+GQ	501	4,812	13,893	0	143	1.1	0.0	-40,731	1,357	1,944	0.7	0.5	0.0	0.0	15.67
C2	18	AAN	501	4,809	13,884	0	168	1.1	0.0	-36,549	2,269	2,402	0.0	0.6	0.0	0.0	18.39
		AAN+GQ	501	4,809	13,884	0	168	1.1	0.0	-36,549	2,269	2,402	0.7	0.6	0.0	0.0	19.37
C2	21	AAN	126	287	2,587	0	1	11.3	0.0	-4,221	10	34	0.0	0.0	0.0	0.0	0.17
		AAN+GQ	121	272	2,444	4	1	11.3	0.0	-4,221	26	50	0.6	0.0	0.0	0.0	0.75
C2	22	AAN	37	59	524	0	0	6.2	0.0	-3,657	2	3	0.0	0.0	0.0	0.0	0.02
		AAN+GQ	37	58	510	5	0	6.2	0.0	-3,657	1	2	0.0	0.0	0.0	0.0	0.06
C3	1	AAN	505	4,968	30,162	0	6	29.5	0.0	-43	210	690	0.0	0.1	0.0	0.0	15.33
		AAN+GQ	505	4,968	30,162	0	6	29.5	0.0	-43	210	690	2.1	0.1	0.0	0.0	17.70
C3	2	AAN	505	4,968	30,162	0	54	12.7	0.0	-81	7,568	10,221	0.0	2.6	0.0	0.0	165.49
		AAN+GQ	505	4,968	30,162	0	54	12.7	0.0	-81	7,568	10,221	2.1	2.7	0.0	0.0	170.49
C3	3	AAN	505	4,968	30,162	0	122	9.7	0.0	-114	11,380	17,877	0.0	3.9	0.0	0.0	214.04
		AAN+GQ	505	4,968	30,162	0	122	9.7	0.0	-114	11,380	17,877	2.3	3.9	0.0	0.0	219.41
C3	4	AAN	510	4,968	50,496	0	0	◇	◇	◇	◇	122	0.0	0.0	0.0	0.0	18.36
		AAN+GQ	510	4,968	50,496	0	0	◇	◇	◇	◇	122	4.3	0.0	0.0	0.0	22.35
C3	5	AAN	510	4,968	50,496	0	59	20.5	0.0	-59	4,900	9,279	0.0	2.2	0.0	0.0	266.62
		AAN+GQ	510	4,968	50,496	0	59	20.5	0.0	-59	4,900	9,279	4.7	2.2	0.0	0.0	270.91
C3	6	AAN	510	4,968	50,496	0	52	11.8	0.0	-88	2,000	11,498	0.0	2.2	0.0	0.0	253.14
		AAN+GQ	510	4,968	50,496	0	52	11.8	0.0	-88	2,000	11,498	4.8	2.2	0.0	0.0	259.70

◇ Problem is infeasible

Table 22 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C3	7	AAN	505	4,963	30,094	0	1	22.8	0.0	-41	30	151	0.0	0.0	0.0	0.0	6.42
		AAN+GQ	505	4,963	30,094	0	1	22.8	0.0	-41	30	151	2.1	0.0	0.0	0.0	8.58
C3	8	AAN	505	4,963	30,094	0	23	10.1	0.0	-82	1,020	2,077	0.0	0.5	0.0	0.0	43.99
		AAN+GQ	505	4,963	30,094	0	23	10.1	0.0	-82	1,020	2,077	2.1	0.5	0.0	0.0	46.83
C3	9	AAN	505	4,963	30,094	0	62	7.3	0.0	-119	5,000	6,418	0.0	1.6	0.0	0.0	103.00
		AAN+GQ	505	4,963	30,094	0	62	7.3	0.0	-119	5,000	6,418	2.2	1.6	0.0	0.0	106.98
C3	10	AAN	510	4,963	50,370	0	0	◇	◇	◇	◇	66	0.0	0.0	0.0	0.0	12.64
		AAN+GQ	510	4,963	50,370	0	0	◇	◇	◇	◇	66	4.1	0.0	0.0	0.0	16.45
C3	11	AAN	510	4,963	50,370	0	52	18.0	0.0	-56	6,050	10,958	0.0	2.6	0.0	0.0	298.54
		AAN+GQ	510	4,963	50,370	0	52	18.0	0.0	-56	6,050	10,958	4.5	2.7	0.0	0.0	311.06
C3	12	AAN	510	4,963	50,370	0	116	12.0	0.0	-83	22,160	38,791	0.0	9.5	0.0	0.0	837.87
		AAN+GQ	510	4,963	50,370	0	116	12.0	0.0	-83	22,160	38,791	4.6	9.5	0.0	0.0	853.72
C3	13	AAN	753	7,522	45,845	0	0	18.5	0.0	-35	10	21	0.0	0.0	0.0	0.0	5.24
		AAN+GQ	753	7,522	45,845	0	0	18.5	0.0	-35	10	21	4.6	0.0	0.0	0.0	9.72
C3	14	AAN	753	7,522	45,845	0	10	10.1	0.0	-73	10	1,061	0.0	0.3	0.0	0.0	35.47
		AAN+GQ	753	7,522	45,845	0	10	10.1	0.0	-73	10	1,061	5.0	0.3	0.0	0.0	40.40
C3	15	AAN	753	7,522	45,845	0	86	8.3	0.0	-105	10,703	14,174	0.0	5.0	0.0	0.0	394.22
		AAN+GQ	753	7,522	45,845	0	86	8.3	0.0	-105	10,703	14,174	5.2	5.1	0.0	0.0	402.12
C3	16	AAN	758	7,522	76,663	0	0	◇	◇	◇	◇	2	0.0	0.0	0.0	0.0	4.47
		AAN+GQ	758	7,522	76,663	0	0	◇	◇	◇	◇	2	8.5	0.0	0.0	0.0	13.03
C3	17	AAN	758	7,522	76,663	0	23	21.5	0.0	-57	4,650	8,738	0.0	2.8	0.0	0.0	503.57
		AAN+GQ	758	7,522	76,663	0	23	21.5	0.0	-57	4,650	8,738	9.8	2.8	0.0	0.0	526.50
C3	18	AAN	758	7,522	76,663	0	99	14.8	0.0	-85	6,120	27,230	0.0	8.0	0.0	0.0	1,395.87
		AAN+GQ	758	7,522	76,663	0	99	14.8	0.0	-85	6,120	27,230	10.3	8.2	0.0	0.0	1,414.98
C3	19	AAN	755	7,595	46,279	0	5	34.4	0.0	-38	1,318	1,999	0.0	0.6	0.0	0.0	64.15
		AAN+GQ	755	7,595	46,279	0	5	34.4	0.0	-38	1,318	1,999	4.9	0.5	0.0	0.0	69.04
C3	20	AAN	755	7,595	46,279	0	28	12.7	0.0	-80	4,440	5,746	0.0	2.5	0.0	0.0	182.20
		AAN+GQ	755	7,595	46,279	0	28	12.7	0.0	-80	4,440	5,746	4.8	2.7	0.0	0.0	194.34
C3	21	AAN	755	7,595	46,279	0	70	8.0	0.0	-117	2,229	4,228	0.0	1.5	0.0	0.0	150.20
		AAN+GQ	755	7,595	46,279	0	70	8.0	0.0	-117	2,229	4,228	4.7	1.5	0.0	0.0	157.34
C3	22	AAN	760	7,595	77,359	0	0	◇	◇	◇	◇	156	0.0	0.0	0.0	0.0	63.15
		AAN+GQ	760	7,595	77,359	0	0	◇	◇	◇	◇	156	9.4	0.0	0.0	0.0	73.33
C3	23	AAN	760	7,595	77,359	0	5	11.2	0.0	-62	1,163	1,196	0.0	0.4	0.0	0.0	82.81
		AAN+GQ	760	7,595	77,359	0	5	11.2	0.0	-62	1,163	1,196	10.1	0.4	0.0	0.0	93.31

◇ Problem is infeasible

Table 22 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C3	24	AAN	760	7,595	77,359	0	105	15.2	0.0	-85	5,940	53,629	0.0	16.1	0.0	0.0	2,428.04
		AAN+GQ	760	7,595	77,359	0	105	15.2	0.0	-85	5,940	53,629	10.1	16.5	0.0	0.0	2,482.29
C3	25	AAN	1,003	9,960	60,766	0	0	◇	◇	◇	◇	36	0.0	0.0	0.0	0.0	21.19
		AAN+GQ	1,003	9,960	60,766	0	0	◇	◇	◇	◇	36	8.7	0.0	0.0	0.0	29.73
C3	26	AAN	1,003	9,960	60,766	0	9	16.5	0.0	-69	1,860	2,329	0.0	0.8	0.0	0.0	128.32
		AAN+GQ	1,003	9,960	60,766	0	9	16.5	0.0	-69	1,860	2,329	8.5	0.9	0.0	0.0	140.00
C3	27	AAN	1,003	9,960	60,766	0	121	11.1	0.0	-106	14,874	19,724	0.0	10.1	0.0	0.0	1,134.53
		AAN+GQ	1,003	9,960	60,766	0	121	11.1	0.0	-106	14,874	19,724	8.6	10.2	0.0	0.0	1,145.77
C3	28	AAN	1,008	9,960	101,846	0	0	◇	◇	◇	◇	4	0.0	0.0	0.0	0.0	16.18
		AAN+GQ	1,008	9,960	101,846	0	0	◇	◇	◇	◇	4	16.6	0.0	0.0	0.0	32.99
C3	29	AAN	1,008	9,960	101,846	0	14	26.8	0.0	-49	3,300	11,713	0.0	4.7	0.0	0.0	1,214.36
		AAN+GQ	1,008	9,960	101,846	0	14	26.8	0.0	-49	3,300	11,713	18.2	4.7	0.0	0.0	1,226.79
C3	30	AAN	1,008	9,960	101,846	0	109	17.4	7.0	-77	19,980	45,042	0.0	27.7	0.0	0.0	3,600.00
		AAN+GQ	1,008	9,960	101,846	0	109	17.4	7.1	-77	19,980	44,664	18.6	27.8	0.0	0.0	3,600.00
C3	31	AAN	1,003	10,095	61,479	0	0	25.4	0.0	-40	336	409	0.0	0.2	0.0	0.0	58.75
		AAN+GQ	1,003	10,095	61,479	0	0	25.4	0.0	-40	336	409	8.7	0.2	0.0	0.0	71.41
C3	32	AAN	1,003	10,095	61,479	0	25	13.5	0.0	-75	10	5,091	0.0	1.9	0.0	0.0	187.08
		AAN+GQ	1,003	10,095	61,479	0	25	13.5	0.0	-75	10	5,091	9.0	1.9	0.0	0.0	196.98
C3	33	AAN	1,003	10,095	61,479	0	29	7.7	0.0	-111	3,110	6,893	0.0	3.5	0.0	0.0	245.42
		AAN+GQ	1,003	10,095	61,479	0	29	7.7	0.0	-111	3,110	6,893	9.1	3.5	0.0	0.0	260.77
C3	34	AAN	1,008	10,095	102,720	0	0	◇	◇	◇	◇	218	0.0	0.1	0.0	0.0	139.19
		AAN+GQ	1,008	10,095	102,720	0	0	◇	◇	◇	◇	218	18.2	0.1	0.0	0.0	159.96
C3	35	AAN	1,008	10,095	102,720	0	62	17.9	0.0	-59	7,935	9,919	0.0	6.0	0.0	0.0	828.71
		AAN+GQ	1,008	10,095	102,720	0	62	17.9	0.0	-59	7,935	9,919	18.4	5.6	0.0	0.0	787.13
C3	36	AAN	1,008	10,095	102,720	0	163	12.9	3.2	-86	33,205	39,049	0.0	23.2	0.0	0.0	3,600.00
		AAN+GQ	1,008	10,095	102,720	0	163	12.9	3.3	-86	33,205	38,336	18.6	23.2	0.0	0.0	3,600.00

◇ Problem is infeasible

Table 23: RCSPP branching scheme results

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A2	3	ARC	99	935	2,655	0	0	1.0	0.0	-78	0	0	0.0	0.0	0.0	0.0	0.04
		ND1	99	935	2,655	0	0	1.0	0.0	-78	0	0	0.0	0.0	0.0	0.0	0.04
		ND2	99	935	2,655	0	0	1.0	0.0	-78	0	0	0.0	0.0	0.0	0.0	0.04
		ND3	99	935	2,655	0	0	1.0	0.0	-78	0	0	0.0	0.0	0.0	0.0	0.04
		ND4	99	935	2,655	0	0	1.0	0.0	-78	0	0	0.0	0.0	0.0	0.0	0.04
A2	4	ARC	99	927	2,632	0	0	0.3	0.0	-74	0	0	0.0	0.0	0.0	0.0	0.04
		ND1	99	927	2,632	0	0	0.3	0.0	-74	0	0	0.0	0.0	0.0	0.0	0.03
		ND2	99	927	2,632	0	0	0.3	0.0	-74	0	0	0.0	0.0	0.0	0.0	0.03
		ND3	99	927	2,632	0	0	0.3	0.0	-74	0	0	0.0	0.0	0.0	0.0	0.03
		ND4	99	927	2,632	0	0	0.3	0.0	-74	0	0	0.0	0.0	0.0	0.0	0.03
A2	7	ARC	89	708	7,405	124	0	1.9	0.0	-29	13	13	0.3	0.0	0.0	0.0	0.62
		ND1	89	708	7,405	126	0	1.9	0.0	-29	10	10	0.3	0.0	0.0	0.0	0.50
		ND2	89	708	7,405	125	0	1.9	0.0	-29	22	22	0.3	0.0	0.0	0.0	0.58
		ND3	89	708	7,405	130	0	1.9	0.0	-29	14	14	0.3	0.0	0.0	0.0	0.53
		ND4	89	708	7,405	126	0	1.9	0.0	-29	15	15	0.3	0.0	0.0	0.0	0.53
A2	8	ARC	50	157	1,476	18	0	1.8	0.0	-23	0	0	0.1	0.0	0.0	0.0	0.08
		ND1	50	157	1,476	18	0	1.8	0.0	-23	0	0	0.1	0.0	0.0	0.0	0.07
		ND2	50	157	1,476	18	0	1.8	0.0	-23	0	0	0.1	0.0	0.0	0.0	0.07
		ND3	50	157	1,476	18	0	1.8	0.0	-23	0	0	0.1	0.0	0.0	0.0	0.07
		ND4	50	157	1,476	18	0	1.8	0.0	-23	0	0	0.1	0.0	0.0	0.0	0.07
A2	11	ARC	195	1,915	5,471	0	0	0.5	0.0	-125	0	0	0.0	0.0	0.0	0.0	0.10
		ND1	195	1,915	5,471	0	0	0.5	0.0	-125	0	0	0.0	0.0	0.0	0.0	0.09
		ND2	195	1,915	5,471	0	0	0.5	0.0	-125	0	0	0.0	0.0	0.0	0.0	0.09
		ND3	195	1,915	5,471	0	0	0.5	0.0	-125	0	0	0.0	0.0	0.0	0.0	0.09
		ND4	195	1,915	5,471	0	0	0.5	0.0	-125	0	0	0.0	0.0	0.0	0.0	0.09
A2	12	ARC	195	1,915	5,471	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.08
		ND1	195	1,915	5,471	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.07
		ND2	195	1,915	5,471	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.07
		ND3	195	1,915	5,471	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.07
		ND4	195	1,915	5,471	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.07
A2	15	ARC	151	1,179	12,201	752	0	8.1	0.0	-24	2	2	0.7	0.0	0.0	0.0	1.30
		ND1	151	1,179	12,201	657	0	8.1	0.0	-24	1	5	0.5	0.0	0.0	0.0	1.04
		ND2	151	1,179	12,201	713	0	8.1	0.0	-24	7	7	0.5	0.0	0.0	0.0	1.17
		ND3	151	1,179	12,201	744	0	8.1	0.0	-24	2	4	0.5	0.0	0.0	0.0	1.07
		ND4	151	1,179	12,201	657	0	8.1	0.0	-24	1	5	0.5	0.0	0.0	0.0	1.05
A2	16	ARC	22	17	161	7	0	11.8	0.0	-17	0	4	0.1	0.0	0.0	0.0	0.08
		ND1	22	17	161	7	0	11.8	0.0	-17	0	4	0.1	0.0	0.0	0.0	0.06
		ND2	22	17	161	7	0	11.8	0.0	-17	0	4	0.1	0.0	0.0	0.0	0.07
		ND3	22	17	161	7	0	11.8	0.0	-17	0	4	0.1	0.0	0.0	0.0	0.06
		ND4	22	17	161	7	0	11.8	0.0	-17	0	4	0.1	0.0	0.0	0.0	0.06

◇ Problem is infeasible

Table 23 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A2	19	ARC	474	4,682	13,373	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.26
		ND1	474	4,682	13,373	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.23
		ND2	474	4,682	13,373	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.24
		ND3	474	4,682	13,373	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.23
		ND4	474	4,682	13,373	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.23
A2	20	ARC	474	4,668	13,331	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.25
		ND1	474	4,668	13,331	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.22
		ND2	474	4,668	13,331	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.23
		ND3	474	4,668	13,331	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.22
		ND4	474	4,668	13,331	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.22
A2	23	ARC	471	4,471	48,353	37	0	24.8	0.0	-36	491	1,120	4.2	0.0	0.0	1.8	52.40
		ND1	471	4,471	48,353	28	0	24.8	0.0	-36	894	2,079	3.2	0.0	0.0	3.2	50.54
		ND2	471	4,471	48,353	25	0	24.8	0.0	-36	90	363	3.3	0.0	0.0	0.6	24.26
		ND3	471	4,471	48,353	34	0	24.8	0.0	-36	710	885	3.4	0.0	0.0	1.3	26.06
		ND4	471	4,471	48,353	27	0	24.8	0.0	-36	110	490	3.5	0.0	0.0	0.7	22.96
A2	24	ARC	414	3,532	37,812	1,148	0	9.9	0.0	-29	61	61	3.1	0.0	0.0	0.1	10.98
		ND1	414	3,532	37,812	1,799	0	9.9	0.0	-29	9	12	2.7	0.0	0.0	0.0	6.15
		ND2	414	3,532	37,812	1,146	0	9.9	0.0	-29	7	7	2.6	0.0	0.0	0.0	6.53
		ND3	414	3,532	37,812	2,099	0	9.9	0.0	-29	11	11	2.8	0.0	0.0	0.0	6.77
		ND4	414	3,532	37,812	2,322	0	9.9	0.0	-29	3	6	2.8	0.0	0.0	0.0	6.69
A3	1	ARC	861	8,335	50,727	1	0	14.5	0.0	-44	660	662	2.4	0.0	0.0	1.8	22.98
		ND1	861	8,335	50,727	1	0	14.5	0.0	-44	547	961	1.9	0.0	0.0	3.1	45.30
		ND2	861	8,335	50,727	7	0	14.5	0.0	-44	70	180	2.7	0.0	0.0	0.6	18.14
		ND3	861	8,335	50,727	6	0	14.5	0.0	-44	248	467	2.2	0.0	0.0	1.4	31.15
		ND4	861	8,335	50,727	2	0	14.5	0.0	-44	54	207	2.5	0.0	0.0	0.6	15.16
A3	2	ARC	876	8,588	52,287	0	0	11.8	0.0	-64	1,970	2,649	1.9	0.0	0.0	7.5	59.51
		ND1	876	8,588	52,287	0	0	11.8	0.0	-64	608	2,756	1.6	0.0	0.0	7.9	67.58
		ND2	876	8,588	52,287	0	0	11.8	0.0	-64	790	1,150	1.6	0.0	0.0	3.3	35.83
		ND3	876	8,588	52,287	0	0	11.8	0.0	-64	2,249	2,991	1.6	0.0	0.0	9.3	84.46
		ND4	876	8,588	52,287	0	0	11.8	0.0	-64	1,247	1,772	1.7	0.0	0.0	5.2	54.45
A3	3	ARC	876	8,596	52,337	0	0	6.7	0.0	-85	0	245	1.4	0.0	0.0	0.7	11.33
		ND1	876	8,596	52,337	0	0	6.7	0.0	-85	0	461	1.2	0.0	0.0	1.3	13.89
		ND2	876	8,596	52,337	0	0	6.7	0.0	-85	0	108	1.3	0.0	0.0	0.3	8.87
		ND3	876	8,596	52,337	0	0	6.7	0.0	-85	0	181	1.3	0.0	0.0	0.5	9.74
		ND4	876	8,596	52,337	0	0	6.7	0.0	-85	0	178	1.3	0.0	0.0	0.5	9.54
A3	4	ARC	852	8,136	82,674	786	0	23.8	0.0	-32	358	725	18.3	0.0	0.0	2.1	89.37
		ND1	852	8,136	82,674	522	0	23.8	0.0	-32	1,780	2,271	12.8	0.0	0.0	6.3	134.82
		ND2	852	8,136	82,674	748	0	23.8	0.0	-32	238	329	14.9	0.0	0.0	1.1	88.20
		ND3	852	8,136	82,674	900	0	23.8	0.0	-32	230	490	14.7	0.0	0.0	1.4	81.62
		ND4	852	8,136	82,674	747	0	23.8	0.0	-32	140	346	16.0	0.0	0.0	1.1	80.23

◇ Problem is infeasible

Table 23 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	5	ARC	881	8,588	87,286	0	0	16.2	0.0	-54	2,290	8,072	5.3	0.0	0.0	23.7	244.56
		ND1	881	8,588	87,286	0	0	16.2	0.0	-54	1,090	16,712	4.2	0.0	0.0	46.2	360.65
		ND2	881	8,588	87,286	0	0	16.2	0.0	-54	410	3,843	4.4	0.0	0.0	11.0	186.74
		ND3	881	8,588	87,286	0	0	16.2	0.0	-54	5,200	9,354	4.1	0.0	0.0	27.4	346.19
		ND4	881	8,588	87,286	0	0	16.2	0.0	-54	710	5,277	4.4	0.0	0.0	14.6	207.71
A3	6	ARC	881	8,596	87,365	0	0	12.4	0.0	-73	2,580	9,798	3.3	0.0	0.0	30.8	308.17
		ND1	881	8,596	87,365	0	0	12.4	0.0	-73	16,990	37,304	2.9	0.0	0.0	108.1	761.45
		ND2	881	8,596	87,365	0	0	12.4	0.0	-73	730	5,294	2.7	0.0	0.0	15.3	201.32
		ND3	881	8,596	87,365	0	0	12.4	0.0	-73	4,100	12,072	2.9	0.0	0.0	35.6	384.33
		ND4	881	8,596	87,365	0	0	12.4	0.0	-73	200	8,655	2.9	0.0	0.0	23.5	221.47
A3	7	ARC	878	8,572	52,137	0	0	18.9	0.0	-44	40	869	2.5	0.0	0.0	2.5	37.30
		ND1	878	8,572	52,137	0	0	18.9	0.0	-44	1,850	2,173	2.2	0.0	0.0	6.5	76.24
		ND2	878	8,572	52,137	3	0	18.9	0.0	-44	104	368	2.7	0.0	0.0	1.0	24.69
		ND3	878	8,572	52,137	0	0	18.9	0.0	-44	340	615	2.0	0.0	0.0	1.8	31.78
		ND4	878	8,572	52,137	0	0	18.9	0.0	-44	90	386	2.2	0.0	0.0	1.1	24.60
A3	8	ARC	883	8,712	53,010	0	0	8.4	0.0	-70	300	579	1.5	0.0	0.0	1.7	22.15
		ND1	883	8,712	53,010	0	0	8.4	0.0	-70	2,398	2,398	1.2	0.0	0.0	7.7	91.54
		ND2	883	8,712	53,010	0	0	8.4	0.0	-70	170	274	1.3	0.0	0.0	1.0	22.83
		ND3	883	8,712	53,010	0	0	8.4	0.0	-70	229	419	1.3	0.0	0.0	1.3	25.08
		ND4	883	8,712	53,010	0	0	8.4	0.0	-70	290	438	1.2	0.0	0.0	1.5	27.87
A3	9	ARC	883	8,712	53,010	0	0	6.0	0.0	-89	140	699	1.3	0.0	0.0	2.0	21.06
		ND1	883	8,712	53,010	0	0	6.0	0.0	-89	80	1,282	1.1	0.0	0.0	3.6	23.95
		ND2	883	8,712	53,010	0	0	6.0	0.0	-89	400	559	1.1	0.0	0.0	1.7	16.02
		ND3	883	8,712	53,010	0	0	6.0	0.0	-89	1,224	1,390	1.1	0.0	0.0	4.0	26.72
		ND4	883	8,712	53,010	0	0	6.0	0.0	-89	730	957	1.0	0.0	0.0	2.8	22.33
A3	10	ARC	865	8,385	85,175	5	0	40.1	0.0	-33	388	6,334	7.5	0.0	0.0	18.0	264.86
		ND1	865	8,385	85,175	9	0	40.1	0.0	-33	1,429	13,154	6.5	0.0	0.0	35.7	399.55
		ND2	865	8,385	85,175	9	0	40.1	0.0	-33	470	2,432	7.6	0.0	0.0	7.1	244.74
		ND3	865	8,385	85,175	44	0	40.1	0.0	-33	860	4,759	8.4	0.0	0.0	12.9	256.64
		ND4	865	8,385	85,175	5	0	40.1	0.0	-33	640	4,041	7.5	0.0	0.0	11.0	245.51
A3	11	ARC	888	8,712	88,550	0	0	15.1	0.0	-60	3,547	6,480	4.4	0.0	0.0	19.3	205.02
		ND1	888	8,712	88,550	0	0	15.1	0.0	-60	27,620	28,093	4.0	0.0	0.0	85.2	723.79
		ND2	888	8,712	88,550	0	0	15.1	0.0	-60	280	2,086	3.8	0.0	0.0	6.0	122.68
		ND3	888	8,712	88,550	0	0	15.1	0.0	-60	5,430	6,431	3.7	0.0	0.0	18.6	227.37
		ND4	888	8,712	88,550	0	0	15.1	0.0	-60	4,930	5,081	4.5	0.0	0.0	14.9	210.56
A3	12	ARC	888	8,712	88,550	0	0	11.8	0.0	-79	22,350	23,083	2.9	0.0	0.0	74.9	595.74
		ND1	888	8,712	88,550	0	0	11.8	0.0	-79	49,340	52,823	2.6	0.0	0.0	154.6	954.80
		ND2	888	8,712	88,550	0	0	11.8	0.0	-79	4,357	7,364	2.8	0.0	0.0	21.6	259.51
		ND3	888	8,712	88,550	0	0	11.8	0.0	-79	21,673	21,830	2.9	0.0	0.0	63.8	504.09
		ND4	888	8,712	88,550	0	0	11.8	0.0	-79	13,946	13,946	2.9	0.0	0.0	39.7	339.83

◇ Problem is infeasible

Table 23 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	13	ARC	1,085	10,312	62,518	72	0	19.4	0.0	-42	60	428	5.9	0.0	0.0	1.4	32.58
		ND1	1,085	10,312	62,518	78	0	19.4	0.0	-42	1,093	1,396	4.5	0.0	0.0	5.1	85.06
		ND2	1,085	10,312	62,518	65	0	19.4	0.0	-42	96	241	4.4	0.0	0.0	1.0	27.90
		ND3	1,085	10,312	62,518	66	0	19.4	0.0	-42	293	429	4.4	0.0	0.0	1.5	36.98
		ND4	1,085	10,312	62,518	49	0	19.4	0.0	-42	420	651	3.6	0.0	0.0	2.2	38.93
A3	14	ARC	1,098	10,671	64,763	0	0	10.0	0.0	-67	300	568	2.3	0.0	0.0	2.1	45.34
		ND1	1,098	10,671	64,763	0	0	10.0	0.0	-67	545	988	2.1	0.0	0.0	4.0	85.91
		ND2	1,098	10,671	64,763	0	0	10.0	0.0	-67	354	414	2.0	0.0	0.0	1.8	42.42
		ND3	1,098	10,671	64,763	0	0	10.0	0.0	-67	190	328	2.0	0.0	0.0	1.3	37.68
		ND4	1,098	10,671	64,763	0	0	10.0	0.0	-67	20	317	2.0	0.0	0.0	1.1	21.81
A3	15	ARC	1,098	10,671	64,763	0	0	5.9	0.0	-89	380	441	1.6	0.0	0.0	1.9	39.69
		ND1	1,098	10,671	64,763	0	0	5.9	0.0	-89	291	462	1.3	0.0	0.0	2.0	52.67
		ND2	1,098	10,671	64,763	0	0	5.9	0.0	-89	70	108	1.4	0.0	0.0	0.4	17.22
		ND3	1,098	10,671	64,763	0	0	5.9	0.0	-89	10	155	1.4	0.0	0.0	0.5	15.16
		ND4	1,098	10,671	64,763	0	0	5.9	0.0	-89	70	195	1.3	0.0	0.0	0.8	21.65
A3	16	ARC	1,033	9,720	98,838	7,455	0	0.0	0.0	-32	0	0	20.8	0.0	0.0	0.0	58.46
		ND1	1,033	9,720	98,838	7,455	0	0.0	0.0	-32	0	0	17.6	0.0	0.0	0.0	51.19
		ND2	1,033	9,720	98,838	7,455	0	0.0	0.0	-32	0	0	17.6	0.0	0.0	0.0	51.43
		ND3	1,033	9,720	98,838	7,455	0	0.0	0.0	-32	0	0	17.3	0.0	0.0	0.0	50.79
		ND4	1,033	9,720	98,838	7,455	0	0.0	0.0	-32	0	0	17.7	0.0	0.0	0.0	50.35
A3	17	ARC	1,103	10,671	108,673	0	0	22.3	0.0	-55	7,313	13,349	5.5	0.0	0.0	49.3	514.71
		ND1	1,103	10,671	108,673	0	0	22.3	0.0	-55	2,040	23,027	5.6	0.0	0.0	79.6	747.97
		ND2	1,103	10,671	108,673	0	0	22.3	0.0	-55	390	4,387	5.7	0.0	0.0	15.1	290.59
		ND3	1,103	10,671	108,673	0	0	22.3	0.0	-55	640	9,047	5.7	0.0	0.0	30.5	373.28
		ND4	1,103	10,671	108,673	0	0	22.3	0.0	-55	2,576	9,205	5.6	0.0	0.0	31.8	437.24
A3	18	ARC	1,103	10,671	108,673	0	0	11.3	0.0	-78	1,760	4,860	4.8	0.0	0.0	17.9	205.90
		ND1	1,103	10,671	108,673	0	0	11.3	0.0	-78	150	8,789	3.9	0.0	0.0	29.9	271.64
		ND2	1,103	10,671	108,673	0	0	11.3	0.0	-78	710	2,074	4.3	0.0	0.0	8.0	189.69
		ND3	1,103	10,671	108,673	0	0	11.3	0.0	-78	1,513	3,811	4.3	0.0	0.0	13.1	169.09
		ND4	1,103	10,671	108,673	0	0	11.3	0.0	-78	1,140	3,594	4.1	0.0	0.0	12.5	199.81
A3	19	ARC	1,052	10,187	61,817	1	0	12.4	0.0	-50	200	316	3.1	0.0	0.0	1.2	34.18
		ND1	1,052	10,187	61,817	0	0	12.4	0.0	-50	1,220	1,228	2.5	0.0	0.0	5.1	133.51
		ND2	1,052	10,187	61,817	0	0	12.4	0.0	-50	300	333	2.5	0.0	0.0	1.6	54.17
		ND3	1,052	10,187	61,817	0	0	12.4	0.0	-50	249	262	2.5	0.0	0.0	0.9	21.76
		ND4	1,052	10,187	61,817	0	0	12.4	0.0	-50	300	338	2.5	0.0	0.0	1.5	54.41
A3	20	ARC	1,054	10,241	62,171	0	0	13.4	0.0	-74	280	980	2.1	0.0	0.0	3.3	47.64
		ND1	1,054	10,241	62,171	0	0	13.4	0.0	-74	10	1,592	1.8	0.0	0.0	5.0	49.82
		ND2	1,054	10,241	62,171	0	0	13.4	0.0	-74	130	514	2.0	0.0	0.0	1.7	40.68
		ND3	1,054	10,241	62,171	0	0	13.4	0.0	-74	470	995	1.9	0.0	0.0	3.3	54.55
		ND4	1,054	10,241	62,171	0	0	13.4	0.0	-74	147	753	1.9	0.0	0.0	2.4	41.33

◇ Problem is infeasible

Table 23 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	21	ARC	1,054	10,241	62,171	0	0	8.9	0.0	-97	2,695	2,695	1.6	0.0	0.0	9.1	77.81
		ND1	1,054	10,241	62,171	0	0	8.9	0.0	-97	779	1,763	1.4	0.0	0.0	6.1	68.77
		ND2	1,054	10,241	62,171	0	0	8.9	0.0	-97	758	758	1.4	0.0	0.0	2.8	41.16
		ND3	1,054	10,241	62,171	0	0	8.9	0.0	-97	1,366	1,549	1.4	0.0	0.0	5.5	68.24
		ND4	1,054	10,241	62,171	0	0	8.9	0.0	-97	1,043	1,122	1.4	0.0	0.0	3.8	45.40
A3	22	ARC	1,037	9,934	101,100	798	0	21.7	0.0	-36	20	457	26.6	0.0	0.0	1.5	79.53
		ND1	1,037	9,934	101,100	704	0	21.7	0.0	-36	1,236	1,236	19.9	0.0	0.0	4.0	108.57
		ND2	1,037	9,934	101,100	1,466	0	21.7	0.0	-36	87	114	27.6	0.0	0.0	0.4	71.45
		ND3	1,037	9,934	101,100	1,450	0	21.7	0.0	-36	530	560	26.8	0.0	0.0	1.9	105.06
		ND4	1,037	9,934	101,100	1,459	0	21.7	0.0	-36	124	132	28.1	0.0	0.0	0.5	73.03
A3	23	ARC	1,059	10,241	104,275	0	0	16.6	0.0	-59	1,960	3,840	5.6	0.0	0.0	15.4	354.42
		ND1	1,059	10,241	104,275	0	0	16.6	0.0	-59	4,970	8,153	4.7	0.0	0.0	29.2	485.29
		ND2	1,059	10,241	104,275	0	0	16.6	0.0	-59	330	1,086	4.8	0.0	0.0	3.8	131.41
		ND3	1,059	10,241	104,275	0	0	16.6	0.0	-59	3,280	4,780	5.0	0.0	0.0	18.7	444.55
		ND4	1,059	10,241	104,275	0	0	16.6	0.0	-59	1,889	2,140	5.3	0.0	0.0	7.8	219.58
A3	24	ARC	1,059	10,241	104,275	0	0	11.7	0.0	-79	8,510	10,354	4.2	0.0	0.0	41.5	503.98
		ND1	1,059	10,241	104,275	0	0	11.7	0.0	-79	4,000	11,767	3.2	0.0	0.0	43.3	609.24
		ND2	1,059	10,241	104,275	0	0	11.7	0.0	-79	670	2,188	3.6	0.0	0.0	8.1	210.68
		ND3	1,059	10,241	104,275	0	0	11.7	0.0	-79	720	4,520	3.3	0.0	0.0	15.5	241.96
		ND4	1,059	10,241	104,275	0	0	11.7	0.0	-79	3,580	4,874	3.6	0.0	0.0	18.2	333.48
A3	25	ARC	1,187	10,917	66,231	3	0	9.6	0.0	-46	594	603	4.0	0.0	0.0	2.7	74.96
		ND1	1,187	10,917	66,231	0	0	9.6	0.0	-46	980	980	2.7	0.0	0.0	4.8	132.54
		ND2	1,187	10,917	66,231	4	0	9.6	0.0	-46	60	84	2.8	0.0	0.0	0.4	32.40
		ND3	1,187	10,917	66,231	0	0	9.6	0.0	-46	360	360	2.7	0.0	0.0	1.4	30.46
		ND4	1,187	10,917	66,231	0	0	9.6	0.0	-46	220	230	2.7	0.0	0.0	1.1	40.99
A3	26	ARC	1,250	11,947	72,646	0	0	9.7	0.0	-65	180	865	3.3	0.0	0.0	3.5	53.83
		ND1	1,250	11,947	72,646	0	0	9.7	0.0	-65	2,867	3,128	2.8	0.0	0.0	12.7	113.50
		ND2	1,250	11,947	72,646	0	0	9.7	0.0	-65	309	499	3.0	0.0	0.0	2.1	44.74
		ND3	1,250	11,947	72,646	0	0	9.7	0.0	-65	80	508	2.9	0.0	0.0	2.0	36.41
		ND4	1,250	11,947	72,646	0	0	9.7	0.0	-65	110	444	3.0	0.0	0.0	1.7	39.03
A3	27	ARC	1,252	11,991	72,938	0	0	6.1	0.0	-86	1,290	1,333	2.5	0.0	0.0	6.1	77.91
		ND1	1,252	11,991	72,938	0	0	6.1	0.0	-86	230	522	2.0	0.0	0.0	2.4	54.83
		ND2	1,252	11,991	72,938	0	0	6.1	0.0	-86	277	302	2.1	0.0	0.0	1.3	30.97
		ND3	1,252	11,991	72,938	0	0	6.1	0.0	-86	360	459	2.1	0.0	0.0	2.2	51.31
		ND4	1,252	11,991	72,938	0	0	6.1	0.0	-86	119	223	2.1	0.0	0.0	1.1	34.82
A3	28	ARC	1,105	10,020	102,017	350	0	22.9	0.0	-35	2,920	2,927	14.2	0.0	0.0	11.0	245.45
		ND1	1,105	10,020	102,017	264	0	22.9	0.0	-35	3,850	3,850	11.2	0.0	0.0	13.8	284.77
		ND2	1,105	10,020	102,017	579	0	22.9	0.0	-35	459	501	13.3	0.0	0.0	2.1	142.78
		ND3	1,105	10,020	102,017	343	0	22.9	0.0	-35	678	853	12.4	0.0	0.0	3.7	224.04
		ND4	1,105	10,020	102,017	343	0	22.9	0.0	-35	1,067	1,069	12.6	0.0	0.0	4.1	166.71

◇ Problem is infeasible

Table 23 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	29	ARC	1,254	11,931	121,785	1	0	18.0	0.0	-55	770	4,686	9.6	0.0	0.0	19.8	343.25
		ND1	1,254	11,931	121,785	0	0	18.0	0.0	-55	4,320	10,892	7.4	0.0	0.0	43.6	538.24
		ND2	1,254	11,931	121,785	1	0	18.0	0.0	-55	1,899	2,618	9.3	0.0	0.0	10.9	275.22
		ND3	1,254	11,931	121,785	0	0	18.0	0.0	-55	1,660	5,038	7.6	0.0	0.0	21.3	398.44
		ND4	1,254	11,931	121,785	1	0	18.0	0.0	-55	2,350	3,796	9.6	0.0	0.0	15.5	308.32
A3	30	ARC	1,257	11,991	122,393	0	0	12.9	0.0	-74	5,620	7,685	7.4	0.0	0.0	34.6	501.06
		ND1	1,257	11,991	122,393	0	0	12.9	0.0	-74	21,190	24,740	5.3	0.0	0.0	103.4	994.31
		ND2	1,257	11,991	122,393	0	0	12.9	0.0	-74	652	2,046	5.7	0.0	0.0	8.5	224.93
		ND3	1,257	11,991	122,393	0	0	12.9	0.0	-74	894	4,651	5.6	0.0	0.0	18.6	296.96
		ND4	1,257	11,991	122,393	0	0	12.9	0.0	-74	1,110	3,682	6.2	0.0	0.0	15.0	277.93
A3	31	ARC	1,276	12,182	74,022	2	0	18.7	0.0	-45	430	778	3.8	0.0	0.0	3.1	57.19
		ND1	1,276	12,182	74,022	3	0	18.7	0.0	-45	40	678	3.4	0.0	0.0	2.6	53.79
		ND2	1,276	12,182	74,022	2	0	18.7	0.0	-45	70	229	3.3	0.0	0.0	1.0	48.48
		ND3	1,276	12,182	74,022	2	0	18.7	0.0	-45	280	543	3.3	0.0	0.0	2.3	78.56
		ND4	1,276	12,182	74,022	0	0	18.7	0.0	-45	201	364	3.3	0.0	0.0	1.7	53.75
A3	32	ARC	1,288	12,501	76,002	0	0	6.8	0.0	-68	429	442	3.4	0.0	0.0	1.8	38.10
		ND1	1,288	12,501	76,002	0	0	6.8	0.0	-68	380	479	2.9	0.0	0.0	2.2	65.75
		ND2	1,288	12,501	76,002	0	0	6.8	0.0	-68	180	206	3.0	0.0	0.0	1.0	36.31
		ND3	1,288	12,501	76,002	0	0	6.8	0.0	-68	339	382	2.8	0.0	0.0	2.1	67.66
		ND4	1,288	12,501	76,002	0	0	6.8	0.0	-68	40	121	2.9	0.0	0.0	0.6	33.99
A3	33	ARC	1,288	12,502	76,009	0	0	5.8	0.0	-86	1,128	1,151	2.9	0.0	0.0	5.9	84.70
		ND1	1,288	12,502	76,009	0	0	5.8	0.0	-86	3,263	3,604	2.6	0.0	0.0	18.9	289.61
		ND2	1,288	12,502	76,009	0	0	5.8	0.0	-86	450	502	2.2	0.0	0.0	2.7	66.33
		ND3	1,288	12,502	76,009	0	0	5.8	0.0	-86	20	226	2.4	0.0	0.0	0.9	29.75
		ND4	1,288	12,502	76,009	0	0	5.8	0.0	-86	150	423	2.3	0.0	0.0	2.0	54.21
A3	34	ARC	1,255	11,872	121,108	1,048	0	23.9	0.0	-36	1	424	26.1	0.0	0.0	1.6	91.90
		ND1	1,255	11,872	121,108	84	0	23.9	0.0	-36	4,000	4,345	12.7	0.0	0.0	17.6	544.95
		ND2	1,255	11,872	121,108	136	0	23.9	0.0	-36	400	685	15.3	0.0	0.0	3.2	219.02
		ND3	1,255	11,872	121,108	196	0	23.9	0.0	-36	1,197	1,713	17.9	0.0	0.0	7.4	396.01
		ND4	1,255	11,872	121,108	120	0	23.9	0.0	-36	147	584	13.5	0.0	0.0	2.2	128.51
A3	35	ARC	1,293	12,498	127,513	0	0	18.2	0.0	-57	770	7,822	7.5	0.0	0.0	32.7	526.69
		ND1	1,293	12,498	127,513	0	0	18.2	0.0	-57	4,630	20,151	6.5	0.0	0.0	78.7	860.47
		ND2	1,293	12,498	127,513	0	0	18.2	0.0	-57	270	3,546	6.4	0.0	0.0	14.1	317.05
		ND3	1,293	12,498	127,513	0	0	18.2	0.0	-57	3,900	9,104	5.8	0.0	0.0	35.7	512.97
		ND4	1,293	12,498	127,513	0	0	18.2	0.0	-57	520	5,661	6.5	0.0	0.0	22.2	446.32
A3	36	ARC	1,293	12,502	127,555	0	0	10.9	0.0	-77	1,400	6,382	6.5	0.0	0.0	27.1	368.19
		ND1	1,293	12,502	127,555	0	0	10.9	0.0	-77	11,780	24,538	5.9	0.0	0.0	102.4	1,175.13
		ND2	1,293	12,502	127,555	0	0	10.9	0.0	-77	2,538	4,359	6.1	0.0	0.0	19.1	380.91
		ND3	1,293	12,502	127,555	0	0	10.9	0.0	-77	1,000	6,172	6.0	0.0	0.0	24.7	389.30
		ND4	1,293	12,502	127,555	0	0	10.9	0.0	-77	2,800	7,135	5.7	0.0	0.0	30.2	587.68

◇ Problem is infeasible

Table 23 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C2	1	ARC	101	898	2,571	0	399	6.4	0.0	-2,119	2,302	4,523	0.0	0.2	0.0	1.6	7.83
		ND1	101	898	2,571	0	574	6.4	0.0	-2,119	7,172	8,738	0.0	0.5	0.0	3.4	39.36
		ND2	101	898	2,571	0	837	6.4	0.0	-2,119	9,660	9,672	0.0	0.7	0.0	3.9	43.99
		ND3	101	898	2,571	0	489	6.4	0.0	-2,119	4,245	6,072	0.0	0.3	0.0	2.1	18.13
C2	2	ARC	101	879	2,515	0	768	8.1	0.0	-1,868	4,704	7,813	0.0	0.5	0.0	2.5	15.80
		ND1	101	879	2,515	0	468	8.1	0.0	-1,868	3,508	6,987	0.0	0.3	0.0	2.5	18.80
		ND2	101	879	2,515	0	636	8.1	0.0	-1,868	7,254	7,358	0.0	0.4	0.0	2.6	24.03
		ND3	101	879	2,515	0	623	8.1	0.0	-1,868	4,864	7,858	0.0	0.3	0.0	2.5	21.90
C2	5	ARC	110	907	9,860	0	88	15.2	0.0	-1,515	1,788	2,120	0.0	0.1	0.0	0.7	5.44
		ND1	110	907	9,860	0	91	15.2	0.0	-1,515	5,945	6,122	0.0	0.3	0.0	2.3	14.17
		ND2	110	907	9,860	0	82	15.2	0.0	-1,515	1,988	2,652	0.0	0.2	0.0	1.0	7.08
		ND3	110	907	9,860	0	69	15.2	0.0	-1,515	1,830	2,863	0.0	0.1	0.0	0.9	5.20
C2	6	ARC	110	868	9,367	0	128	19.3	0.0	-1,279	4,630	5,084	0.0	0.2	0.0	1.6	8.04
		ND1	110	868	9,367	0	80	19.3	0.0	-1,279	1,080	6,336	0.0	0.3	0.0	2.2	11.11
		ND2	110	868	9,367	0	114	19.3	0.0	-1,279	4,260	4,994	0.0	0.2	0.0	1.9	12.13
		ND3	110	868	9,367	0	121	19.3	0.0	-1,279	4,785	5,945	0.0	0.2	0.0	1.8	10.43
C2	9	ARC	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		ND1	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		ND2	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		ND3	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
C2	10	ARC	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		ND1	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		ND2	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		ND3	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
C2	13	ARC	20	11	109	0	0	0.0	0.0	-593	0	0	0.0	0.0	0.0	0.0	0.00
		ND1	20	11	109	0	0	0.0	0.0	-593	0	0	0.0	0.0	0.0	0.0	0.00
		ND2	20	11	109	0	0	0.0	0.0	-593	0	0	0.0	0.0	0.0	0.0	0.00
		ND3	20	11	109	0	0	0.0	0.0	-593	0	0	0.0	0.0	0.0	0.0	0.00
C2	14	ARC	67	94	862	0	0	◇	◇	◇	◇	0	0.0	0.0	0.0	0.0	0.00
		ND1	67	94	862	0	0	◇	◇	◇	◇	0	0.0	0.0	0.0	0.0	0.00
		ND2	67	94	862	0	0	◇	◇	◇	◇	0	0.0	0.0	0.0	0.0	0.00
		ND3	67	94	862	0	0	◇	◇	◇	◇	0	0.0	0.0	0.0	0.0	0.00
C2	17	ARC	501	4,812	13,893	0	165	1.1	0.0	-40,731	1,778	1,782	0.0	0.6	0.0	3.9	23.86
		ND1	501	4,812	13,893	0	88	1.1	0.0	-40,731	1,149	2,120	0.0	0.6	0.0	4.4	25.09
		ND2	501	4,812	13,893	0	183	1.1	0.0	-40,731	2,766	3,537	0.0	1.2	0.0	9.1	53.94
		ND3	501	4,812	13,893	0	144	1.1	0.0	-40,731	1,930	2,864	0.0	0.8	0.0	6.1	34.34
C2	18	ARC	501	4,809	13,884	0	156	1.1	0.0	-36,549	559	1,258	0.0	0.4	0.0	2.8	17.22
		ND1	501	4,809	13,884	0	135	1.1	0.0	-36,549	1,709	3,273	0.0	1.0	0.0	7.8	51.19
		ND2	501	4,809	13,884	0	184	1.1	0.0	-36,549	2,434	2,680	0.0	0.8	0.0	6.1	36.26
		ND3	501	4,809	13,884	0	149	1.1	0.0	-36,549	2,548	3,390	0.0	0.9	0.0	7.1	37.17

◇ Problem is infeasible

Table 23 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C2	21	ARC	197	358	3,298	0	1	12.1	0.0	-4,221	40	57	0.0	0.0	0.0	0.0	0.32
		ND1	197	358	3,298	0	1	12.1	0.0	-4,221	28	42	0.0	0.0	0.0	0.0	0.28
		ND2	197	358	3,298	0	4	12.1	0.0	-4,221	12	30	0.0	0.0	0.0	0.0	0.28
		ND3	197	358	3,298	0	2	12.1	0.0	-4,221	10	34	0.0	0.0	0.0	0.0	0.19
C2	22	ARC	72	94	908	0	0	6.2	0.0	-3,657	6	6	0.0	0.0	0.0	0.0	0.02
		ND1	72	94	908	0	0	6.2	0.0	-3,657	2	6	0.0	0.0	0.0	0.0	0.02
		ND2	72	94	908	0	0	6.2	0.0	-3,657	2	2	0.0	0.0	0.0	0.0	0.02
		ND3	72	94	908	0	0	6.2	0.0	-3,657	2	2	0.0	0.0	0.0	0.0	0.01
C3	1	ARC	505	4,968	30,162	0	7	29.5	0.0	-43	250	695	0.0	0.2	0.0	1.3	32.09
		ND1	505	4,968	30,162	0	6	29.5	0.0	-43	930	2,603	0.0	0.5	0.0	5.0	46.70
		ND2	505	4,968	30,162	0	4	29.5	0.0	-43	120	446	0.0	0.1	0.0	0.8	19.33
		ND3	505	4,968	30,162	0	5	29.5	0.0	-43	210	519	0.0	0.1	0.0	1.0	18.18
C3	2	ARC	505	4,968	30,162	0	53	12.7	0.0	-81	8,780	9,703	0.0	2.2	0.0	18.4	219.35
		ND1	505	4,968	30,162	0	78	12.7	0.0	-81	33,619	33,748	0.0	7.4	0.0	66.8	481.04
		ND2	505	4,968	30,162	0	52	12.7	0.0	-81	5,845	6,192	0.0	1.3	0.0	10.8	120.15
		ND3	505	4,968	30,162	0	43	12.7	0.0	-81	2,530	7,771	0.0	1.7	0.0	15.7	164.11
C3	3	ARC	505	4,968	30,162	0	332	9.7	0.0	-114	41,685	44,007	0.0	10.7	0.0	88.1	905.80
		ND1	505	4,968	30,162	0	319	9.7	0.0	-114	150,809	221,104	0.0	49.4	0.0	450.1	3,097.80
		ND2	505	4,968	30,162	0	209	9.7	0.0	-114	40,967	41,658	0.0	9.1	0.0	78.7	627.13
		ND3	505	4,968	30,162	0	218	9.7	0.0	-114	61,536	84,975	0.0	20.1	0.0	181.8	1,343.83
C3	4	ARC	510	4,968	50,496	0	0	◇	◇	◇	◇	92	0.0	0.0	0.0	0.2	19.43
		ND1	510	4,968	50,496	0	0	◇	◇	◇	◇	1,764	0.0	0.4	0.0	3.5	134.46
		ND2	510	4,968	50,496	0	0	◇	◇	◇	◇	74	0.0	0.0	0.0	0.1	16.17
		ND3	510	4,968	50,496	0	0	◇	◇	◇	◇	88	0.0	0.0	0.0	0.2	18.44
C3	5	ARC	510	4,968	50,496	0	43	20.5	0.0	-59	13,866	15,122	0.0	3.1	0.0	26.9	410.79
		ND1	510	4,968	50,496	0	45	20.5	0.0	-59	28,150	56,626	0.0	11.9	0.0	108.5	999.83
		ND2	510	4,968	50,496	0	24	20.5	0.0	-59	1,508	4,273	0.0	0.8	0.0	7.4	175.89
		ND3	510	4,968	50,496	0	25	20.5	0.0	-59	2,170	6,351	0.0	1.3	0.0	12.6	236.70
C3	6	ARC	510	4,968	50,496	0	53	11.8	0.0	-88	5,510	15,514	0.0	3.4	0.0	27.7	376.24
		ND1	510	4,968	50,496	0	100	11.8	0.0	-88	180	105,456	0.0	21.6	0.0	194.8	1,408.09
		ND2	510	4,968	50,496	0	45	11.8	0.0	-88	2,070	11,537	0.0	2.1	0.0	19.2	319.45
		ND3	510	4,968	50,496	0	58	11.8	0.0	-88	1,170	18,151	0.0	3.9	0.0	34.5	402.55
C3	7	ARC	505	4,963	30,094	0	2	22.8	0.0	-41	407	409	0.0	0.1	0.0	0.7	11.66
		ND1	505	4,963	30,094	0	4	22.8	0.0	-41	4,010	4,257	0.0	1.2	0.0	12.0	122.20
		ND2	505	4,963	30,094	0	2	22.8	0.0	-41	381	381	0.0	0.1	0.0	0.7	16.70
		ND3	505	4,963	30,094	0	1	22.8	0.0	-41	298	385	0.0	0.1	0.0	0.8	14.59
C3	8	ARC	505	4,963	30,094	0	59	10.1	0.0	-82	10,510	11,700	0.0	2.7	0.0	23.5	440.15
		ND1	505	4,963	30,094	0	50	10.1	0.0	-82	10,670	16,404	0.0	3.6	0.0	34.6	257.34
		ND2	505	4,963	30,094	0	23	10.1	0.0	-82	110	1,064	0.0	0.2	0.0	1.8	22.19
		ND3	505	4,963	30,094	0	24	10.1	0.0	-82	880	2,437	0.0	0.5	0.0	4.9	60.41

◇ Problem is infeasible

Table 23 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C3	9	ARC	505	4,963	30,094	0	101	7.3	0.0	-119	7,010	10,930	0.0	2.8	0.0	23.0	312.71
		ND1	505	4,963	30,094	0	134	7.3	0.0	-119	35,540	56,266	0.0	13.4	0.0	122.5	941.10
		ND2	505	4,963	30,094	0	118	7.3	0.0	-119	6,804	9,980	0.0	2.4	0.0	21.5	256.47
		ND3	505	4,963	30,094	0	100	7.3	0.0	-119	7,210	12,015	0.0	3.0	0.0	27.6	265.87
C3	10	ARC	510	4,963	50,370	0	0	◇	◇	◇	◇	70	0.0	0.0	0.0	0.1	14.28
		ND1	510	4,963	50,370	0	0	◇	◇	◇	◇	778	0.0	0.2	0.0	1.5	77.05
		ND2	510	4,963	50,370	0	0	◇	◇	◇	◇	36	0.0	0.0	0.0	0.1	9.85
		ND3	510	4,963	50,370	0	0	◇	◇	◇	◇	48	0.0	0.0	0.0	0.1	12.35
C3	11	ARC	510	4,963	50,370	0	68	18.0	0.0	-56	5,930	9,915	0.0	2.4	0.0	19.7	718.09
		ND1	510	4,963	50,370	0	52	18.0	0.0	-56	25,000	45,157	0.0	9.7	0.0	88.9	943.32
		ND2	510	4,963	50,370	0	32	18.0	0.0	-56	2,479	4,432	0.0	0.9	0.0	7.9	190.84
		ND3	510	4,963	50,370	0	34	18.0	0.0	-56	3,040	7,054	0.0	1.5	0.0	14.1	259.30
C3	12	ARC	510	4,963	50,370	0	151	12.0	0.0	-83	61,430	72,477	0.0	17.0	0.0	142.8	2,501.29
		ND1	510	4,963	50,370	0	158	12.0	3.5	-83	114,638	188,742	0.0	48.7	0.0	438.7	3,600.00
		ND2	510	4,963	50,370	0	110	12.0	0.0	-83	15,310	28,599	0.0	5.7	0.0	51.5	883.19
		ND3	510	4,963	50,370	0	122	12.0	0.0	-83	19,370	46,806	0.0	11.1	0.0	100.7	1,373.55
C3	13	ARC	755	7,524	45,856	0	0	18.5	0.0	-35	0	13	0.0	0.0	0.0	0.0	1.58
		ND1	755	7,524	45,856	0	0	18.5	0.0	-35	0	22	0.0	0.0	0.0	0.1	1.63
		ND2	755	7,524	45,856	0	0	18.5	0.0	-35	0	6	0.0	0.0	0.0	0.0	1.27
		ND3	755	7,524	45,856	0	0	18.5	0.0	-35	0	8	0.0	0.0	0.0	0.0	1.47
C3	14	ARC	755	7,524	45,856	0	5	10.1	0.0	-73	10	952	0.0	0.3	0.0	2.4	32.03
		ND1	755	7,524	45,856	0	42	10.1	0.0	-73	4,550	7,337	0.0	2.6	0.0	23.7	296.20
		ND2	755	7,524	45,856	0	8	10.1	0.0	-73	550	920	0.0	0.3	0.0	2.3	73.31
		ND3	755	7,524	45,856	0	6	10.1	0.0	-73	20	760	0.0	0.2	0.0	2.2	30.04
C3	15	ARC	755	7,524	45,856	0	80	8.3	0.0	-105	13,349	16,202	0.0	5.5	0.0	48.8	720.60
		ND1	755	7,524	45,856	0	127	8.3	0.0	-105	46,620	62,459	0.0	21.0	0.0	192.3	1,509.42
		ND2	755	7,524	45,856	0	96	8.3	0.0	-105	13,012	13,489	0.0	4.5	0.0	40.8	564.77
		ND3	755	7,524	45,856	0	114	8.3	0.0	-105	16,715	20,020	0.0	7.6	0.0	69.2	829.93
C3	16	ARC	760	7,524	76,691	0	0	◇	◇	◇	◇	2	0.0	0.0	0.0	0.0	5.48
		ND1	760	7,524	76,691	0	0	◇	◇	◇	◇	120	0.0	0.0	0.0	0.4	41.41
		ND2	760	7,524	76,691	0	0	◇	◇	◇	◇	2	0.0	0.0	0.0	0.0	4.72
		ND3	760	7,524	76,691	0	0	◇	◇	◇	◇	2	0.0	0.0	0.0	0.0	5.61
C3	17	ARC	760	7,524	76,691	0	13	21.5	0.0	-57	3,374	6,165	0.0	1.9	0.0	16.8	438.82
		ND1	760	7,524	76,691	0	33	21.5	0.0	-57	15,240	28,548	0.0	9.0	0.0	85.2	1,430.11
		ND2	760	7,524	76,691	0	6	21.5	0.0	-57	2,165	3,291	0.0	0.9	0.0	8.4	312.85
		ND3	760	7,524	76,691	0	13	21.5	0.0	-57	4,150	6,017	0.0	2.0	0.0	18.7	527.41
C3	18	ARC	760	7,524	76,691	0	88	14.8	0.0	-85	3,210	35,060	0.0	10.7	0.0	92.4	1,402.49
		ND1	760	7,524	76,691	0	124	17.6	12.6	-83	11,360	77,112	0.0	41.0	0.0	378.2	3,600.00
		ND2	760	7,524	76,691	0	96	14.8	0.0	-85	3,420	17,353	0.0	5.3	0.0	47.2	1,116.70
		ND3	760	7,524	76,691	0	95	14.8	0.0	-85	3,080	34,069	0.0	11.1	0.0	100.9	1,922.86

◇ Problem is infeasible

Table 23 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C3	19	ARC	755	7,595	46,279	0	3	34.4	0.0	-38	1,890	2,161	0.0	0.7	0.0	5.6	85.34
		ND1	755	7,595	46,279	0	6	34.4	0.0	-38	1,220	2,087	0.0	0.6	0.0	5.9	95.73
		ND2	755	7,595	46,279	0	3	34.4	0.0	-38	313	492	0.0	0.1	0.0	1.2	24.72
		ND3	755	7,595	46,279	0	3	34.4	0.0	-38	840	895	0.0	0.3	0.0	2.6	53.32
C3	20	ARC	755	7,595	46,279	0	8	12.7	0.0	-80	30	1,532	0.0	0.5	0.0	3.6	40.08
		ND1	755	7,595	46,279	0	18	12.7	0.0	-80	30	3,181	0.0	1.0	0.0	8.8	72.78
		ND2	755	7,595	46,279	0	23	12.7	0.0	-80	310	1,100	0.0	0.3	0.0	2.8	65.64
		ND3	755	7,595	46,279	0	13	12.7	0.0	-80	10	1,257	0.0	0.4	0.0	3.5	37.05
C3	21	ARC	755	7,595	46,279	0	122	8.0	0.0	-117	19,970	19,970	0.0	6.3	0.0	57.4	1,081.01
		ND1	755	7,595	46,279	0	67	8.0	0.0	-117	7,450	16,146	0.0	5.1	0.0	47.8	484.67
		ND2	755	7,595	46,279	0	80	8.0	0.0	-117	2,740	3,635	0.0	1.2	0.0	10.7	198.50
		ND3	755	7,595	46,279	0	90	8.0	0.0	-117	2,710	5,699	0.0	2.1	0.0	19.3	294.45
C3	22	ARC	760	7,595	77,359	0	0	◇	◇	◇	◇	138	0.0	0.0	0.0	0.4	71.00
		ND1	760	7,595	77,359	0	0	◇	◇	◇	◇	1,016	0.0	0.3	0.0	3.2	290.07
		ND2	760	7,595	77,359	0	0	◇	◇	◇	◇	66	0.0	0.0	0.0	0.2	36.70
		ND3	760	7,595	77,359	0	0	◇	◇	◇	◇	130	0.0	0.0	0.0	0.4	64.88
C3	23	ARC	760	7,595	77,359	0	4	11.2	0.0	-62	840	948	0.0	0.3	0.0	2.6	69.57
		ND1	760	7,595	77,359	0	9	11.2	0.0	-62	2,350	3,605	0.0	1.3	0.0	11.7	198.75
		ND2	760	7,595	77,359	0	5	11.2	0.0	-62	540	600	0.0	0.2	0.0	1.8	58.99
		ND3	760	7,595	77,359	0	2	11.2	0.0	-62	200	421	0.0	0.2	0.0	1.4	50.85
C3	24	ARC	760	7,595	77,359	0	184	25.6	21.4	-78	8,010	17,752	0.0	7.6	0.0	69.7	3,600.00
		ND1	760	7,595	77,359	0	155	19.4	15.4	-82	40,380	55,817	0.0	31.8	0.0	294.1	3,600.00
		ND2	760	7,595	77,359	0	129	15.2	0.0	-85	13,170	50,641	0.0	15.2	0.0	135.7	3,175.52
		ND3	760	7,595	77,359	0	134	15.2	6.7	-85	43,090	44,095	0.0	22.2	0.0	210.4	3,600.00
C3	25	ARC	1,003	9,960	60,766	0	0	◇	◇	◇	◇	42	0.0	0.0	0.0	0.2	24.93
		ND1	1,003	9,960	60,766	0	0	◇	◇	◇	◇	2,166	0.0	0.9	0.0	8.7	417.77
		ND2	1,003	9,960	60,766	0	0	◇	◇	◇	◇	34	0.0	0.0	0.0	0.1	20.69
		ND3	1,003	9,960	60,766	0	0	◇	◇	◇	◇	46	0.0	0.0	0.0	0.2	26.54
C3	26	ARC	1,003	9,960	60,766	0	6	16.5	0.0	-69	2,245	2,585	0.0	1.1	0.0	9.1	148.45
		ND1	1,003	9,960	60,766	0	33	16.5	0.0	-69	10,700	11,963	0.0	5.3	0.0	50.1	874.70
		ND2	1,003	9,960	60,766	0	15	16.5	0.0	-69	1,860	2,216	0.0	0.9	0.0	7.9	251.27
		ND3	1,003	9,960	60,766	0	9	16.5	0.0	-69	1,929	2,578	0.0	1.2	0.0	11.0	285.47
C3	27	ARC	1,003	9,960	60,766	0	193	11.1	0.0	-106	15,197	18,863	0.0	8.5	0.0	78.4	2,217.91
		ND1	1,003	9,960	60,766	0	184	16.6	11.4	-101	25,720	61,922	0.0	40.4	0.0	391.2	3,600.00
		ND2	1,003	9,960	60,766	0	126	11.1	0.0	-106	14,507	14,507	0.0	5.9	0.0	52.9	997.81
		ND3	1,003	9,960	60,766	0	131	11.1	0.0	-106	20,600	25,874	0.0	13.7	0.0	130.0	2,150.18
C3	28	ARC	1,008	9,960	101,846	0	0	◇	◇	◇	◇	4	0.0	0.0	0.0	0.0	17.00
		ND1	1,008	9,960	101,846	0	0	◇	◇	◇	◇	1,052	0.0	0.4	0.0	4.2	359.73
		ND2	1,008	9,960	101,846	0	0	◇	◇	◇	◇	4	0.0	0.0	0.0	0.0	14.64
		ND3	1,008	9,960	101,846	0	0	◇	◇	◇	◇	4	0.0	0.0	0.0	0.0	18.46

◇ Problem is infeasible

Table 23 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C3	29	ARC	1,008	9,960	101,846	0	9	26.8	0.0	-49	2,430	9,893	0.0	4.1	0.0	34.2	801.18
		ND1	1,008	9,960	101,846	0	40	26.8	0.0	-49	1,310	47,031	0.0	18.5	0.0	179.3	3,361.46
		ND2	1,008	9,960	101,846	0	10	26.8	0.0	-49	1,750	4,910	0.0	1.8	0.0	16.7	810.39
		ND3	1,008	9,960	101,846	0	12	26.8	0.0	-49	150	5,677	0.0	2.4	0.0	22.0	776.17
C3	30	ARC	1,008	9,960	101,846	0	52	17.4	9.0	-77	10,970	20,417	0.0	9.7	0.0	85.9	3,600.00
		ND1	1,008	9,960	101,846	0	68	29.1	28.7	-70	17,690	23,805	0.0	17.4	0.0	173.5	3,600.00
		ND2	1,008	9,960	101,846	0	82	17.4	6.5	-77	4,530	24,213	0.0	13.2	0.0	119.0	3,600.00
		ND3	1,008	9,960	101,846	0	100	17.4	9.3	-77	7,190	20,683	0.0	13.8	0.0	130.4	3,600.00
C3	31	ARC	1,004	10,096	61,483	0	2	25.4	0.0	-40	210	248	0.0	0.1	0.0	1.0	56.56
		ND1	1,004	10,096	61,483	0	4	25.4	0.0	-40	512	850	0.0	0.4	0.0	3.5	58.94
		ND2	1,004	10,096	61,483	0	3	25.4	0.0	-40	486	510	0.0	0.2	0.0	2.0	77.45
		ND3	1,004	10,096	61,483	0	2	25.4	0.0	-40	299	304	0.0	0.2	0.0	1.4	89.84
C3	32	ARC	1,004	10,096	61,483	0	19	13.5	0.0	-75	10	5,970	0.0	2.5	0.0	20.3	154.56
		ND1	1,004	10,096	61,483	0	97	13.5	0.0	-75	30,590	54,994	0.0	23.9	0.0	229.3	2,514.77
		ND2	1,004	10,096	61,483	0	30	13.5	0.0	-75	1,280	4,412	0.0	1.8	0.0	15.3	253.87
		ND3	1,004	10,096	61,483	0	31	13.5	0.0	-75	1,930	7,235	0.0	3.2	0.0	28.6	342.15
C3	33	ARC	1,004	10,096	61,483	0	93	7.7	0.0	-111	24,876	26,677	0.0	12.1	0.0	107.4	1,235.61
		ND1	1,004	10,096	61,483	0	153	12.8	11.1	-106	61,990	66,907	0.0	47.4	0.0	462.3	3,600.00
		ND2	1,004	10,096	61,483	0	73	7.7	0.0	-111	6,760	11,091	0.0	5.2	0.0	47.1	781.15
		ND3	1,004	10,096	61,483	0	89	7.7	0.0	-111	12,620	21,806	0.0	11.3	0.0	103.7	1,341.16
C3	34	ARC	1,009	10,096	103,118	0	0	◇	◇	◇	◇	128	0.0	0.1	0.0	0.5	112.94
		ND1	1,009	10,096	103,118	0	0	◇	◇	◇	◇	4,396	0.0	1.8	0.0	17.8	1,145.83
		ND2	1,009	10,096	103,118	0	0	◇	◇	◇	◇	80	0.0	0.0	0.0	0.3	76.21
		ND3	1,009	10,096	103,118	0	0	◇	◇	◇	◇	136	0.0	0.1	0.0	0.6	116.06
C3	35	ARC	1,009	10,096	103,118	0	23	17.9	0.0	-59	1,970	4,506	0.0	2.1	0.0	17.2	571.34
		ND1	1,009	10,096	103,118	0	77	22.1	14.2	-57	35,530	44,604	0.0	29.1	0.0	288.8	3,600.00
		ND2	1,009	10,096	103,118	0	49	17.9	0.0	-59	2,920	3,792	0.0	1.8	0.0	16.4	722.84
		ND3	1,009	10,096	103,118	0	82	17.9	0.0	-59	7,452	8,837	0.0	4.7	0.0	44.9	1,658.44
C3	36	ARC	1,009	10,096	103,118	0	109	17.0	10.5	-83	13,470	14,576	0.0	8.3	0.0	76.9	3,600.00
		ND1	1,009	10,096	103,118	0	122	21.4	20.7	-80	6,980	26,066	0.0	19.1	0.0	190.7	3,600.00
		ND2	1,009	10,096	103,118	0	151	15.6	6.7	-84	12,580	18,985	0.0	11.3	0.0	101.0	3,600.00
		ND3	1,009	10,096	103,118	0	104	24.5	19.0	-78	6,150	15,428	0.0	10.9	0.0	104.7	3,600.00

◇ Problem is infeasible

Table 24: RCSPP branching scheme results with preprocessing

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A2	3	ARC+AAN	99	935	2,655	0	0	1.0	0.0	-78	0	0	0.0	0.0	0.0	0.0	0.04
		ND2+AAN	99	935	2,655	0	0	1.0	0.0	-78	0	0	0.0	0.0	0.0	0.0	0.04
A2	4	ARC+AAN	99	927	2,632	0	0	0.3	0.0	-74	0	0	0.0	0.0	0.0	0.0	0.03
		ND2+AAN	99	927	2,632	0	0	0.3	0.0	-74	0	0	0.0	0.0	0.0	0.0	0.03
A2	7	ARC+AAN	89	708	7,405	116	0	1.9	0.0	-29	8	8	0.3	0.0	0.0	0.0	0.44
		ND2+AAN	89	708	7,405	117	0	1.9	0.0	-29	18	18	0.3	0.0	0.0	0.0	0.54
A2	8	ARC+AAN	50	157	1,476	18	0	1.8	0.0	-23	0	0	0.1	0.0	0.0	0.0	0.07
		ND2+AAN	50	157	1,476	18	0	1.8	0.0	-23	0	0	0.1	0.0	0.0	0.0	0.07
A2	11	ARC+AAN	195	1,915	5,471	0	0	0.5	0.0	-125	0	0	0.0	0.0	0.0	0.0	0.09
		ND2+AAN	195	1,915	5,471	0	0	0.5	0.0	-125	0	0	0.0	0.0	0.0	0.0	0.09
A2	12	ARC+AAN	195	1,915	5,471	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.07
		ND2+AAN	195	1,915	5,471	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.07
A2	15	ARC+AAN	151	1,179	12,201	771	0	8.1	0.0	-24	2	2	0.5	0.0	0.0	0.0	1.05
		ND2+AAN	151	1,179	12,201	740	0	8.1	0.0	-24	3	4	0.5	0.0	0.0	0.0	1.03
A2	16	ARC+AAN	22	17	161	7	0	11.8	0.0	-17	0	2	0.1	0.0	0.0	0.0	0.06
		ND2+AAN	22	17	161	7	0	11.8	0.0	-17	0	2	0.1	0.0	0.0	0.0	0.07
A2	19	ARC+AAN	474	4,682	13,373	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.24
		ND2+AAN	474	4,682	13,373	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.23
A2	20	ARC+AAN	474	4,668	13,331	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.23
		ND2+AAN	474	4,668	13,331	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.22
A2	23	ARC+AAN	471	4,471	48,353	27	0	24.8	0.0	-36	62	110	3.4	0.0	0.0	1.2	14.27
		ND2+AAN	471	4,471	48,353	25	0	24.8	0.0	-36	59	148	3.4	0.0	0.0	0.7	14.80
A2	24	ARC+AAN	414	3,532	37,812	1,001	0	9.9	0.0	-29	7	7	2.6	0.0	0.0	0.1	5.97
		ND2+AAN	414	3,532	37,812	1,139	0	9.9	0.0	-29	6	6	2.6	0.0	0.0	0.1	6.24
A3	1	ARC+AAN	861	8,335	50,727	1	0	14.5	0.0	-44	108	134	2.0	0.0	0.0	1.7	17.34
		ND2+AAN	861	8,335	50,727	4	0	14.5	0.0	-44	66	136	2.7	0.0	0.0	1.0	16.71
A3	2	ARC+AAN	876	8,588	52,287	0	0	11.8	0.0	-64	505	613	1.7	0.0	0.0	5.3	34.20
		ND2+AAN	876	8,588	52,287	0	0	11.8	0.0	-64	360	621	1.6	0.0	0.0	3.7	31.59
A3	3	ARC+AAN	876	8,596	52,337	0	0	6.7	0.0	-85	0	61	1.2	0.0	0.0	0.5	8.33
		ND2+AAN	876	8,596	52,337	0	0	6.7	0.0	-85	0	65	1.3	0.0	0.0	0.4	8.43
A3	4	ARC+AAN	852	8,136	82,674	791	0	23.8	0.0	-32	180	180	15.7	0.0	0.0	4.3	70.66
		ND2+AAN	852	8,136	82,674	748	0	23.8	0.0	-32	30	87	15.1	0.0	0.0	0.9	49.44
A3	5	ARC+AAN	881	8,588	87,286	0	0	16.2	0.0	-54	520	1,589	4.5	0.0	0.0	25.2	144.38
		ND2+AAN	881	8,588	87,286	0	0	16.2	0.0	-54	295	1,732	4.4	0.0	0.0	14.9	141.88

◇ Problem is infeasible

Table 24 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	6	ARC+AAN	881	8,596	87,365	0	0	12.4	0.0	-73	800	2,877	2.8	0.0	0.0	36.5	191.30
		ND2+AAN	881	8,596	87,365	0	0	12.4	0.0	-73	570	3,061	2.7	0.0	0.0	23.1	171.83
A3	7	ARC+AAN	878	8,572	52,137	0	0	18.9	0.0	-44	10	107	2.1	0.0	0.0	1.3	14.78
		ND2+AAN	878	8,572	52,137	3	0	18.9	0.0	-44	96	174	2.7	0.0	0.0	1.2	20.27
A3	8	ARC+AAN	883	8,712	53,010	0	0	8.4	0.0	-70	120	180	1.3	0.0	0.0	2.0	20.11
		ND2+AAN	883	8,712	53,010	0	0	8.4	0.0	-70	130	204	1.3	0.0	0.0	1.4	22.25
A3	9	ARC+AAN	883	8,712	53,010	0	0	6.0	0.0	-89	60	222	1.2	0.0	0.0	1.8	14.07
		ND2+AAN	883	8,712	53,010	0	0	6.0	0.0	-89	290	354	1.1	0.0	0.0	1.9	14.48
A3	10	ARC+AAN	865	8,385	85,175	5	0	40.1	0.0	-33	186	496	6.3	0.0	0.0	11.8	97.13
		ND2+AAN	865	8,385	85,175	9	0	40.1	0.0	-33	214	548	7.7	0.0	0.0	5.3	92.34
A3	11	ARC+AAN	888	8,712	88,550	0	0	15.1	0.0	-60	609	1,031	3.8	0.0	0.0	17.2	114.15
		ND2+AAN	888	8,712	88,550	0	0	15.1	0.0	-60	888	1,211	3.8	0.0	0.0	10.4	110.50
A3	12	ARC+AAN	888	8,712	88,550	0	0	11.8	0.0	-79	7,536	7,728	2.5	0.0	0.0	100.7	400.07
		ND2+AAN	888	8,712	88,550	0	0	11.8	0.0	-79	4,063	4,905	2.8	0.0	0.0	37.4	238.83
A3	13	ARC+AAN	1,085	10,312	62,518	70	0	19.4	0.0	-42	60	71	4.8	0.0	0.0	1.1	21.02
		ND2+AAN	1,085	10,312	62,518	65	0	19.4	0.0	-42	65	116	4.4	0.0	0.0	1.0	23.37
A3	14	ARC+AAN	1,098	10,671	64,763	0	0	10.0	0.0	-67	110	208	2.0	0.0	0.0	2.5	30.97
		ND2+AAN	1,098	10,671	64,763	0	0	10.0	0.0	-67	240	304	2.0	0.0	0.0	2.5	38.31
A3	15	ARC+AAN	1,098	10,671	64,763	0	0	5.9	0.0	-89	270	340	1.4	0.0	0.0	3.6	31.53
		ND2+AAN	1,098	10,671	64,763	0	0	5.9	0.0	-89	58	85	1.4	0.0	0.0	0.6	16.83
A3	16	ARC+AAN	1,033	9,720	98,838	7,455	0	0.0	0.0	-32	0	0	17.9	0.0	0.0	0.0	56.13
		ND2+AAN	1,033	9,720	98,838	7,455	0	0.0	0.0	-32	0	0	17.8	0.0	0.0	0.0	52.39
A3	17	ARC+AAN	1,103	10,671	108,673	0	0	22.3	0.0	-55	358	1,585	4.8	0.0	0.0	32.7	211.11
		ND2+AAN	1,103	10,671	108,673	0	0	22.3	0.0	-55	290	1,806	5.7	0.0	0.0	19.6	226.89
A3	18	ARC+AAN	1,103	10,671	108,673	0	0	11.3	0.0	-78	70	1,159	4.2	0.0	0.0	18.3	128.90
		ND2+AAN	1,103	10,671	108,673	0	0	11.3	0.0	-78	400	1,233	4.3	0.0	0.0	12.7	163.01
A3	19	ARC+AAN	1,052	10,187	61,817	1	0	12.4	0.0	-50	160	173	2.7	0.0	0.0	2.8	32.40
		ND2+AAN	1,052	10,187	61,817	0	0	12.4	0.0	-50	206	230	2.5	0.0	0.0	2.7	47.95
A3	20	ARC+AAN	1,054	10,241	62,171	0	0	13.4	0.0	-74	78	238	1.9	0.0	0.0	2.5	28.33
		ND2+AAN	1,054	10,241	62,171	0	0	13.4	0.0	-74	100	334	2.0	0.0	0.0	2.3	35.13
A3	21	ARC+AAN	1,054	10,241	62,171	0	0	8.9	0.0	-97	644	654	1.4	0.0	0.0	6.8	50.92
		ND2+AAN	1,054	10,241	62,171	0	0	8.9	0.0	-97	603	603	1.4	0.0	0.0	4.3	44.37
A3	22	ARC+AAN	1,037	9,934	101,100	546	0	21.7	0.0	-36	20	49	20.5	0.0	0.0	1.4	52.10
		ND2+AAN	1,037	9,934	101,100	1,467	0	21.7	0.0	-36	20	45	27.8	0.0	0.0	0.6	61.95

◇ Problem is infeasible

Table 24 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	23	ARC+AAN	1,059	10,241	104,275	0	0	16.6	0.0	-59	566	876	4.9	0.0	0.0	18.5	167.42
		ND2+AAN	1,059	10,241	104,275	0	0	16.6	0.0	-59	249	589	4.8	0.0	0.0	6.3	112.27
A3	24	ARC+AAN	1,059	10,241	104,275	0	0	11.7	0.0	-79	2,936	3,728	3.7	0.0	0.0	72.0	423.79
		ND2+AAN	1,059	10,241	104,275	0	0	11.7	0.0	-79	640	1,573	3.6	0.0	0.0	16.8	210.72
A3	25	ARC+AAN	1,187	10,917	66,231	2	0	9.6	0.0	-46	116	122	3.2	0.0	0.0	2.3	27.93
		ND2+AAN	1,187	10,917	66,231	4	0	9.6	0.0	-46	88	94	2.8	0.0	0.0	1.0	27.21
A3	26	ARC+AAN	1,250	11,947	72,646	0	0	9.7	0.0	-65	148	234	2.9	0.0	0.0	3.5	35.84
		ND2+AAN	1,250	11,947	72,646	0	0	9.7	0.0	-65	102	214	3.0	0.0	0.0	1.9	31.48
A3	27	ARC+AAN	1,252	11,991	72,938	0	0	6.1	0.0	-86	30	80	2.2	0.0	0.0	1.0	23.43
		ND2+AAN	1,252	11,991	72,938	0	0	6.1	0.0	-86	206	225	2.1	0.0	0.0	1.8	29.79
A3	28	ARC+AAN	1,105	10,020	102,017	304	0	22.9	0.0	-35	120	145	11.6	0.0	0.0	4.0	67.99
		ND2+AAN	1,105	10,020	102,017	511	0	22.9	0.0	-35	184	187	13.1	0.0	0.0	2.4	81.46
A3	29	ARC+AAN	1,254	11,931	121,785	1	0	18.0	0.0	-55	731	944	8.5	0.0	0.0	24.1	175.92
		ND2+AAN	1,254	11,931	121,785	1	0	18.0	0.0	-55	1,129	1,269	9.3	0.0	0.0	16.6	210.20
A3	30	ARC+AAN	1,257	11,991	122,393	0	0	12.9	0.0	-74	4,990	5,997	6.4	0.0	0.0	132.8	722.64
		ND2+AAN	1,257	11,991	122,393	0	0	12.9	0.0	-74	546	1,348	5.7	0.0	0.0	15.9	205.27
A3	31	ARC+AAN	1,276	12,182	74,022	2	0	18.7	0.0	-45	67	86	3.4	0.0	0.0	1.7	29.26
		ND2+AAN	1,276	12,182	74,022	0	0	18.7	0.0	-45	46	108	3.1	0.0	0.0	1.1	33.91
A3	32	ARC+AAN	1,288	12,501	76,002	0	0	6.8	0.0	-68	193	197	2.9	0.0	0.0	3.6	50.63
		ND2+AAN	1,288	12,501	76,002	0	0	6.8	0.0	-68	198	219	3.0	0.0	0.0	2.5	50.65
A3	33	ARC+AAN	1,288	12,502	76,009	0	0	5.8	0.0	-86	530	576	2.5	0.0	0.0	7.5	71.80
		ND2+AAN	1,288	12,502	76,009	0	0	5.8	0.0	-86	964	964	2.2	0.0	0.0	9.8	106.25
A3	34	ARC+AAN	1,255	11,872	121,108	186	0	23.9	0.0	-36	2	51	15.6	0.0	0.0	2.3	61.61
		ND2+AAN	1,255	11,872	121,108	129	0	23.9	0.0	-36	110	155	15.3	0.0	0.0	2.4	91.45
A3	35	ARC+AAN	1,293	12,498	127,513	0	0	18.2	0.0	-57	450	1,420	6.4	0.0	0.0	38.3	270.56
		ND2+AAN	1,293	12,498	127,513	0	0	18.2	0.0	-57	620	1,442	6.3	0.0	0.0	19.1	245.78
A3	36	ARC+AAN	1,293	12,502	127,555	0	0	10.9	0.0	-77	10	1,313	5.5	0.0	0.0	25.4	147.49
		ND2+AAN	1,293	12,502	127,555	0	0	10.9	0.0	-77	60	1,268	6.0	0.0	0.0	14.6	166.29
C2	1	ARC+AAN	101	898	2,571	0	257	6.4	0.0	-2,119	24	2,355	0.0	0.1	0.0	0.7	2.91
		ND2+AAN	101	898	2,571	0	757	6.4	0.0	-2,119	8,275	8,655	0.0	0.5	0.0	3.2	32.18
C2	2	ARC+AAN	101	879	2,515	0	947	8.1	0.0	-1,868	10,545	11,049	0.0	0.6	0.0	3.7	23.34
		ND2+AAN	101	879	2,515	0	792	8.1	0.0	-1,868	7,441	9,426	0.0	0.5	0.0	3.3	32.45
C2	5	ARC+AAN	110	907	9,860	0	74	15.2	0.0	-1,515	1,724	2,040	0.0	0.1	0.0	2.0	5.24
		ND2+AAN	110	907	9,860	0	61	15.2	0.0	-1,515	590	1,922	0.0	0.1	0.0	1.3	5.31

◇ Problem is infeasible

Table 24 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C2	6	ARC+AAN	110	868	9,367	0	78	19.3	0.0	-1,279	998	2,933	0.0	0.1	0.0	2.7	7.10
		ND2+AAN	110	868	9,367	0	123	19.3	0.0	-1,279	4,355	5,018	0.0	0.2	0.0	3.1	11.63
C2	9	ARC+AAN	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		ND2+AAN	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
C2	10	ARC+AAN	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		ND2+AAN	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
C2	13	ARC+AAN	20	11	109	0	0	0.0	0.0	-593	0	0	0.0	0.0	0.0	0.0	0.00
		ND2+AAN	20	11	109	0	0	0.0	0.0	-593	0	0	0.0	0.0	0.0	0.0	0.00
C2	14	ARC+AAN	67	94	862	0	0	◇	◇	◇	◇	0	0.0	0.0	0.0	0.0	0.00
		ND2+AAN	67	94	862	0	0	◇	◇	◇	◇	0	0.0	0.0	0.0	0.0	0.00
C2	17	ARC+AAN	501	4,812	13,893	0	133	1.1	0.0	-40,731	1,683	2,296	0.0	0.6	0.0	5.0	18.69
		ND2+AAN	501	4,812	13,893	0	175	1.1	0.0	-40,731	2,308	3,147	0.0	0.9	0.0	8.1	44.33
C2	18	ARC+AAN	501	4,809	13,884	0	120	1.1	0.0	-36,549	1,092	1,220	0.0	0.3	0.0	2.1	7.51
		ND2+AAN	501	4,809	13,884	0	169	1.1	0.0	-36,549	1,077	2,049	0.0	0.6	0.0	4.5	26.52
C2	21	ARC+AAN	197	358	3,298	0	2	12.1	0.0	-4,221	20	30	0.0	0.0	0.0	0.0	0.20
		ND2+AAN	197	358	3,298	0	1	12.1	0.0	-4,221	11	22	0.0	0.0	0.0	0.0	0.22
C2	22	ARC+AAN	72	94	908	0	0	6.2	0.0	-3,657	2	4	0.0	0.0	0.0	0.0	0.02
		ND2+AAN	72	94	908	0	0	6.2	0.0	-3,657	2	2	0.0	0.0	0.0	0.0	0.01
C3	1	ARC+AAN	505	4,968	30,162	0	4	29.5	0.0	-43	380	586	0.0	0.1	0.0	2.9	31.30
		ND2+AAN	505	4,968	30,162	0	4	29.5	0.0	-43	120	363	0.0	0.1	0.0	1.1	19.03
C3	2	ARC+AAN	505	4,968	30,162	0	60	12.7	0.0	-81	8,657	9,660	0.0	2.1	0.0	40.5	241.66
		ND2+AAN	505	4,968	30,162	0	46	12.7	0.0	-81	3,595	5,018	0.0	1.0	0.0	15.1	133.97
C3	3	ARC+AAN	505	4,968	30,162	0	283	9.7	0.0	-114	22,273	27,339	0.0	5.8	0.0	98.3	503.47
		ND2+AAN	505	4,968	30,162	0	130	9.7	0.0	-114	9,095	18,668	0.0	4.0	0.0	52.1	357.54
C3	4	ARC+AAN	510	4,968	50,496	0	0	◇	◇	◇	◇	92	0.0	0.0	0.0	0.8	17.33
		ND2+AAN	510	4,968	50,496	0	0	◇	◇	◇	◇	74	0.0	0.0	0.0	0.4	16.81
C3	5	ARC+AAN	510	4,968	50,496	0	31	20.5	0.0	-59	7,410	10,856	0.0	2.2	0.0	72.4	441.24
		ND2+AAN	510	4,968	50,496	0	21	20.5	0.0	-59	1,549	3,956	0.0	0.8	0.0	15.2	173.87
C3	6	ARC+AAN	510	4,968	50,496	0	31	11.8	0.0	-88	20	7,437	0.0	1.4	0.0	31.5	164.43
		ND2+AAN	510	4,968	50,496	0	48	11.8	0.0	-88	1,939	10,942	0.0	2.1	0.0	35.7	315.66
C3	7	ARC+AAN	505	4,963	30,094	0	2	22.8	0.0	-41	173	178	0.0	0.0	0.0	0.7	7.68
		ND2+AAN	505	4,963	30,094	0	2	22.8	0.0	-41	303	303	0.0	0.1	0.0	0.9	15.64
C3	8	ARC+AAN	505	4,963	30,094	0	24	10.1	0.0	-82	1,030	1,908	0.0	0.4	0.0	8.3	50.05
		ND2+AAN	505	4,963	30,094	0	20	10.1	0.0	-82	110	937	0.0	0.2	0.0	2.3	21.76

◇ Problem is infeasible

Table 24 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C3	9	ARC+AAN	505	4,963	30,094	0	196	7.3	0.0	-119	11,940	15,963	0.0	3.9	0.0	87.4	592.82
		ND2+AAN	505	4,963	30,094	0	89	7.3	0.0	-119	5,220	8,167	0.0	1.9	0.0	25.7	201.75
C3	10	ARC+AAN	510	4,963	50,370	0	0	◇	◇	◇	◇	58	0.0	0.0	0.0	0.5	12.29
		ND2+AAN	510	4,963	50,370	0	0	◇	◇	◇	◇	36	0.0	0.0	0.0	0.2	10.34
C3	11	ARC+AAN	510	4,963	50,370	0	17	18.0	0.0	-56	1,870	4,085	0.0	0.8	0.0	30.0	222.86
		ND2+AAN	510	4,963	50,370	0	34	18.0	0.0	-56	2,590	4,397	0.0	0.9	0.0	18.0	212.41
C3	12	ARC+AAN	510	4,963	50,370	0	88	12.0	0.0	-83	17,828	26,731	0.0	5.4	0.0	154.2	804.85
		ND2+AAN	510	4,963	50,370	0	115	12.0	0.0	-83	20,390	33,910	0.0	7.9	0.0	152.3	1,294.04
C3	13	ARC+AAN	755	7,524	45,856	0	0	18.5	0.0	-35	0	2	0.0	0.0	0.0	0.0	1.12
		ND2+AAN	755	7,524	45,856	0	0	18.5	0.0	-35	0	4	0.0	0.0	0.0	0.0	1.37
C3	14	ARC+AAN	755	7,524	45,856	0	8	10.1	0.0	-73	130	427	0.0	0.1	0.0	2.5	31.59
		ND2+AAN	755	7,524	45,856	0	7	10.1	0.0	-73	550	839	0.0	0.2	0.0	3.7	73.80
C3	15	ARC+AAN	755	7,524	45,856	0	187	8.3	0.0	-105	19,406	19,406	0.0	6.8	0.0	144.3	1,166.83
		ND2+AAN	755	7,524	45,856	0	70	8.3	0.0	-105	6,608	8,587	0.0	3.0	0.0	41.5	432.16
C3	16	ARC+AAN	760	7,524	76,691	0	0	◇	◇	◇	◇	2	0.0	0.0	0.0	0.0	4.85
		ND2+AAN	760	7,524	76,691	0	0	◇	◇	◇	◇	2	0.0	0.0	0.0	0.0	4.61
C3	17	ARC+AAN	760	7,524	76,691	0	7	21.5	0.0	-57	2,612	2,750	0.0	0.9	0.0	29.3	383.72
		ND2+AAN	760	7,524	76,691	0	5	21.5	0.0	-57	1,900	2,786	0.0	0.7	0.0	15.2	282.49
C3	18	ARC+AAN	760	7,524	76,691	0	74	14.8	0.0	-85	7,950	18,779	0.0	5.4	0.0	172.4	1,451.97
		ND2+AAN	760	7,524	76,691	0	87	14.8	0.0	-85	2,950	15,200	0.0	4.5	0.0	82.5	1,053.69
C3	19	ARC+AAN	755	7,595	46,279	0	1	34.4	0.0	-38	200	256	0.0	0.1	0.0	1.4	17.66
		ND2+AAN	755	7,595	46,279	0	2	34.4	0.0	-38	200	251	0.0	0.1	0.0	1.0	17.84
C3	20	ARC+AAN	755	7,595	46,279	0	17	12.7	0.0	-80	900	1,436	0.0	0.5	0.0	9.7	106.24
		ND2+AAN	755	7,595	46,279	0	19	12.7	0.0	-80	455	1,128	0.0	0.3	0.0	4.9	78.29
C3	21	ARC+AAN	755	7,595	46,279	0	67	8.0	0.0	-117	6,297	7,425	0.0	2.7	0.0	66.2	430.80
		ND2+AAN	755	7,595	46,279	0	76	8.0	0.0	-117	2,791	3,595	0.0	1.1	0.0	16.6	196.26
C3	22	ARC+AAN	760	7,595	77,359	0	0	◇	◇	◇	◇	136	0.0	0.0	0.0	1.8	61.91
		ND2+AAN	760	7,595	77,359	0	0	◇	◇	◇	◇	66	0.0	0.0	0.0	0.5	35.93
C3	23	ARC+AAN	760	7,595	77,359	0	3	11.2	0.0	-62	360	470	0.0	0.2	0.0	4.5	51.54
		ND2+AAN	760	7,595	77,359	0	4	11.2	0.0	-62	519	572	0.0	0.2	0.0	3.5	56.65
C3	24	ARC+AAN	760	7,595	77,359	0	147	19.4	12.2	-82	18,750	25,981	0.0	10.5	0.0	415.2	3,600.00
		ND2+AAN	760	7,595	77,359	0	133	15.2	0.0	-85	17,370	51,930	0.0	15.1	0.0	287.9	3,398.26
C3	25	ARC+AAN	1,003	9,960	60,766	0	0	◇	◇	◇	◇	32	0.0	0.0	0.0	0.3	19.38
		ND2+AAN	1,003	9,960	60,766	0	0	◇	◇	◇	◇	34	0.0	0.0	0.0	0.2	19.42

◇ Problem is infeasible

Table 24 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C3	26	ARC+AAN	1,003	9,960	60,766	0	6	16.5	0.0	-69	666	950	0.0	0.4	0.0	7.4	86.40
		ND2+AAN	1,003	9,960	60,766	0	19	16.5	0.0	-69	1,717	1,955	0.0	0.7	0.0	12.2	283.53
C3	27	ARC+AAN	1,003	9,960	60,766	0	157	11.1	0.0	-106	12,334	15,931	0.0	7.2	0.0	150.7	1,444.12
		ND2+AAN	1,003	9,960	60,766	0	87	11.1	0.0	-106	6,516	9,558	0.0	4.2	0.0	64.0	906.44
C3	28	ARC+AAN	1,008	9,960	101,846	0	0	◇	◇	◇	◇	4	0.0	0.0	0.0	0.1	16.40
		ND2+AAN	1,008	9,960	101,846	0	0	◇	◇	◇	◇	4	0.0	0.0	0.0	0.0	14.57
C3	29	ARC+AAN	1,008	9,960	101,846	0	8	26.8	0.0	-49	2,310	4,708	0.0	2.2	0.0	82.2	921.27
		ND2+AAN	1,008	9,960	101,846	0	8	26.8	0.0	-49	1,530	4,060	0.0	1.5	0.0	29.6	731.74
C3	30	ARC+AAN	1,008	9,960	101,846	0	65	17.4	7.5	-77	7,420	22,372	0.0	12.5	0.0	426.9	3,600.00
		ND2+AAN	1,008	9,960	101,846	0	74	17.4	6.1	-77	2,990	25,998	0.0	13.0	0.0	251.8	3,600.00
C3	31	ARC+AAN	1,004	10,096	61,483	0	3	25.4	0.0	-40	77	99	0.0	0.1	0.0	1.1	28.47
		ND2+AAN	1,004	10,096	61,483	0	3	25.4	0.0	-40	240	260	0.0	0.1	0.0	1.8	61.42
C3	32	ARC+AAN	1,004	10,096	61,483	0	22	13.5	0.0	-75	1,206	3,434	0.0	1.4	0.0	24.9	207.36
		ND2+AAN	1,004	10,096	61,483	0	23	13.5	0.0	-75	720	3,377	0.0	1.3	0.0	16.9	207.76
C3	33	ARC+AAN	1,004	10,096	61,483	0	40	7.7	0.0	-111	6,550	9,575	0.0	4.3	0.0	80.7	573.05
		ND2+AAN	1,004	10,096	61,483	0	38	7.7	0.0	-111	1,420	5,575	0.0	2.2	0.0	30.0	294.86
C3	34	ARC+AAN	1,009	10,096	103,118	0	0	◇	◇	◇	◇	154	0.0	0.1	0.0	2.8	124.14
		ND2+AAN	1,009	10,096	103,118	0	0	◇	◇	◇	◇	80	0.0	0.0	0.0	0.9	75.77
C3	35	ARC+AAN	1,009	10,096	103,118	0	20	17.9	0.0	-59	4,379	4,720	0.0	2.0	0.0	68.2	864.64
		ND2+AAN	1,009	10,096	103,118	0	47	17.9	0.0	-59	1,000	2,172	0.0	1.0	0.0	22.9	516.41
C3	36	ARC+AAN	1,009	10,096	103,118	0	199	31.2	27.5	-74	6,660	8,281	0.0	5.8	0.0	272.0	3,600.00
		ND2+AAN	1,009	10,096	103,118	0	120	12.9	0.0	-86	7,934	18,536	0.0	7.6	0.0	149.9	2,536.31

◇ Problem is infeasible

Table 25: RCSPP primal heuristic results

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A2	3	HYBRID	99	935	2,655	0	0	1.0	0.0	-78	0	0	0.0	0.0	0.0	0.0	0.04
		NORINS	99	935	2,655	0	0	1.0	0.0	-78	10	10	0.0	0.0	0.0	0.0	0.05
A2	4	HYBRID	99	927	2,632	0	0	0.3	0.0	-74	4	4	0.0	0.0	0.0	0.0	0.04
		NORINS	99	927	2,632	0	0	0.3	0.0	-74	4	4	0.0	0.0	0.0	0.0	0.04
A2	7	HYBRID	89	708	7,405	105	0	1.9	0.0	-29	0	0	0.3	0.0	0.0	0.0	0.29
		NORINS	89	708	7,405	129	0	1.9	0.0	-29	5	5	0.3	0.0	0.0	0.0	0.34
A2	8	HYBRID	50	157	1,476	44	0	0.0	0.0	-23	0	0	0.1	0.0	0.0	0.0	0.08
		NORINS	50	157	1,476	44	0	0.0	0.0	-23	0	0	0.1	0.0	0.0	0.0	0.08
A2	11	HYBRID	195	1,915	5,471	0	0	0.5	0.0	-125	2	2	0.0	0.0	0.0	0.0	0.09
		NORINS	195	1,915	5,471	0	0	0.5	0.0	-125	2	2	0.0	0.0	0.0	0.0	0.09
A2	12	HYBRID	195	1,915	5,471	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.07
		NORINS	195	1,915	5,471	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.07
A2	15	HYBRID	151	1,179	12,201	661	0	8.1	0.0	-24	1	9	0.5	0.0	0.0	0.0	0.84
		NORINS	151	1,179	12,201	661	0	8.1	0.0	-24	15	19	0.5	0.0	0.0	0.0	0.88
A2	16	HYBRID	22	17	161	7	0	11.8	0.0	-17	0	2	0.1	0.0	0.0	0.0	0.07
		NORINS	22	17	161	7	0	11.8	0.0	-17	0	2	0.1	0.0	0.0	0.0	0.06
A2	19	HYBRID	474	4,682	13,373	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.23
		NORINS	474	4,682	13,373	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.23
A2	20	HYBRID	474	4,668	13,331	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.22
		NORINS	474	4,668	13,331	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.22
A2	23	HYBRID	471	4,471	48,353	50	0	24.8	0.0	-36	49	131	3.2	0.0	0.2	1.5	9.37
		NORINS	471	4,471	48,353	15	0	24.8	0.0	-36	64	86	2.9	0.0	0.0	1.0	6.43
A2	24	HYBRID	414	3,532	37,812	983	0	9.9	0.0	-29	3	7	2.5	0.0	0.0	0.1	4.52
		NORINS	414	3,532	37,812	952	0	9.9	0.0	-29	14	14	2.5	0.0	0.0	0.1	4.72
A3	1	HYBRID	861	8,335	50,727	4	0	14.5	0.0	-44	28	130	2.7	0.0	0.2	0.9	7.81
		NORINS	861	8,335	50,727	4	0	14.5	0.0	-44	33	66	2.7	0.0	0.0	0.4	5.45
A3	2	HYBRID	876	8,588	52,287	0	0	11.8	0.0	-64	57	793	1.6	0.0	1.9	4.7	26.71
		NORINS	876	8,588	52,287	0	0	11.8	0.0	-64	193	495	1.8	0.0	0.0	3.5	16.48
A3	3	HYBRID	876	8,596	52,337	0	0	6.7	0.0	-85	0	152	1.2	0.0	0.3	0.8	6.65
		NORINS	876	8,596	52,337	0	0	6.7	0.0	-85	30	98	1.3	0.0	0.0	0.7	6.00
A3	4	HYBRID	852	8,136	82,674	778	0	27.3	0.0	-32	27	108	15.8	0.0	0.2	1.6	41.94
		NORINS	852	8,136	82,674	250	0	27.3	0.0	-32	40	118	14.1	0.0	0.0	1.3	31.91
A3	5	HYBRID	881	8,588	87,286	0	0	16.2	0.0	-54	102	3,265	4.1	0.0	12.1	31.7	236.00
		NORINS	881	8,588	87,286	0	0	16.2	0.0	-54	180	1,503	4.7	0.0	0.0	14.9	84.82

◇ Problem is infeasible

Table 25 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	6	HYBRID	881	8,596	87,365	0	0	12.4	0.0	-73	2	5,076	2.9	0.0	21.1	38.1	273.55
		NORINS	881	8,596	87,365	0	0	12.4	0.0	-73	628	2,793	3.2	0.0	0.0	24.3	117.14
A3	7	HYBRID	878	8,572	52,137	2	0	18.9	0.0	-44	3	194	2.4	0.0	0.3	1.5	12.28
		NORINS	878	8,572	52,137	1	0	18.9	0.0	-44	143	144	2.2	0.0	0.0	0.9	7.53
A3	8	HYBRID	883	8,712	53,010	0	0	8.4	0.0	-70	50	250	1.4	0.0	0.5	1.4	10.87
		NORINS	883	8,712	53,010	0	0	8.4	0.0	-70	70	152	1.5	0.0	0.0	1.0	8.25
A3	9	HYBRID	883	8,712	53,010	0	0	6.0	0.0	-89	3	492	1.0	0.0	1.1	2.5	13.21
		NORINS	883	8,712	53,010	0	0	6.0	0.0	-89	363	478	1.1	0.0	0.0	3.1	14.84
A3	10	HYBRID	865	8,385	85,175	17	0	40.1	0.0	-33	13	610	8.4	0.0	1.6	7.1	87.16
		NORINS	865	8,385	85,175	14	0	40.1	0.0	-33	20	419	10.5	0.0	0.0	4.9	50.92
A3	11	HYBRID	888	8,712	88,550	0	0	15.1	0.0	-60	168	2,064	3.6	0.0	6.7	20.7	181.06
		NORINS	888	8,712	88,550	0	0	15.1	0.0	-60	280	910	3.9	0.0	0.0	9.2	61.42
A3	12	HYBRID	888	8,712	88,550	0	0	11.8	0.0	-79	732	6,175	2.6	0.0	26.0	47.1	354.37
		NORINS	888	8,712	88,550	0	0	11.8	0.0	-79	2,728	4,260	2.9	0.0	0.0	37.4	181.20
A3	13	HYBRID	1,085	10,312	62,518	66	0	19.4	0.0	-42	59	127	4.4	0.0	0.2	1.1	13.78
		NORINS	1,085	10,312	62,518	65	0	19.4	0.0	-42	72	85	5.4	0.0	0.0	0.8	11.66
A3	14	HYBRID	1,098	10,671	64,763	0	0	10.0	0.0	-67	3	122	1.8	0.0	0.3	0.9	9.63
		NORINS	1,098	10,671	64,763	0	0	10.0	0.0	-67	11	76	2.1	0.0	0.0	0.7	7.67
A3	15	HYBRID	1,098	10,671	64,763	0	0	5.9	0.0	-89	2	66	1.5	0.0	0.2	0.4	5.68
		NORINS	1,098	10,671	64,763	0	0	5.9	0.0	-89	30	77	1.7	0.0	0.0	0.7	6.91
A3	16	HYBRID	1,033	9,720	98,838	1,571	0	29.0	0.0	-32	1	67	15.5	0.0	0.2	0.9	44.41
		NORINS	1,033	9,720	98,838	235	0	29.0	0.0	-32	15	100	13.8	0.0	0.0	1.4	37.11
A3	17	HYBRID	1,103	10,671	108,673	0	0	22.3	0.0	-55	62	3,579	5.9	0.0	13.1	44.7	411.11
		NORINS	1,103	10,671	108,673	0	0	22.3	0.0	-55	53	1,497	6.5	0.0	0.0	18.5	129.38
A3	18	HYBRID	1,103	10,671	108,673	0	0	11.3	0.0	-78	37	1,778	4.3	0.0	8.5	18.2	187.10
		NORINS	1,103	10,671	108,673	0	0	11.3	0.0	-78	90	845	4.9	0.0	0.0	9.5	64.79
A3	19	HYBRID	1,052	10,187	61,817	0	0	12.4	0.0	-50	67	94	2.6	0.0	0.2	0.9	9.95
		NORINS	1,052	10,187	61,817	0	0	12.4	0.0	-50	115	129	3.0	0.0	0.0	1.3	13.55
A3	20	HYBRID	1,054	10,241	62,171	0	0	13.4	0.0	-74	27	500	1.9	0.0	1.1	3.4	30.68
		NORINS	1,054	10,241	62,171	0	0	13.4	0.0	-74	36	243	2.0	0.0	0.0	2.0	15.34
A3	21	HYBRID	1,054	10,241	62,171	0	0	8.9	0.0	-97	29	344	1.3	0.0	0.9	2.1	20.11
		NORINS	1,054	10,241	62,171	0	0	8.9	0.0	-97	187	355	1.4	0.0	0.0	2.7	19.43
A3	22	HYBRID	1,037	9,934	101,100	155	0	28.5	0.0	-36	39	187	13.0	0.0	0.6	2.7	45.34
		NORINS	1,037	9,934	101,100	34	0	28.5	0.0	-36	27	134	11.5	0.0	0.0	1.9	30.96

◇ Problem is infeasible

Table 25 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	23	HYBRID	1,059	10,241	104,275	0	0	16.6	0.0	-59	378	1,026	4.7	0.0	4.2	13.0	155.14
		NORINS	1,059	10,241	104,275	0	0	16.6	0.0	-59	103	482	5.2	0.0	0.0	6.0	57.29
A3	24	HYBRID	1,059	10,241	104,275	0	0	11.7	0.0	-79	72	2,231	3.4	0.0	10.2	22.4	227.36
		NORINS	1,059	10,241	104,275	0	0	11.7	0.0	-79	60	895	3.7	0.0	0.0	10.1	68.14
A3	25	HYBRID	1,187	10,917	66,231	17	0	9.6	0.0	-46	62	74	3.8	0.0	0.2	0.7	12.70
		NORINS	1,187	10,917	66,231	4	0	9.6	0.0	-46	36	47	3.5	0.0	0.0	0.5	8.59
A3	26	HYBRID	1,250	11,947	72,646	0	0	9.7	0.0	-65	155	348	3.0	0.0	1.0	3.2	30.38
		NORINS	1,250	11,947	72,646	0	0	9.7	0.0	-65	36	160	3.4	0.0	0.0	1.7	14.91
A3	27	HYBRID	1,252	11,991	72,938	0	0	6.1	0.0	-86	75	202	2.1	0.0	0.6	1.6	16.56
		NORINS	1,252	11,991	72,938	0	0	6.1	0.0	-86	80	155	2.4	0.0	0.0	1.5	15.60
A3	28	HYBRID	1,105	10,020	102,017	1,441	0	22.9	0.0	-35	0	13	13.7	0.0	0.0	0.2	31.03
		NORINS	1,105	10,020	102,017	511	0	22.9	0.0	-35	20	53	15.1	0.0	0.0	0.8	34.10
A3	29	HYBRID	1,254	11,931	121,785	1	0	18.0	0.0	-55	244	1,584	9.2	0.0	5.7	23.6	248.50
		NORINS	1,254	11,931	121,785	0	0	18.0	0.0	-55	300	860	9.0	0.0	0.0	12.8	113.37
A3	30	HYBRID	1,257	11,991	122,393	0	0	12.9	0.0	-74	190	2,109	5.4	0.0	12.5	25.6	306.48
		NORINS	1,257	11,991	122,393	0	0	12.9	0.0	-74	125	966	6.0	0.0	0.0	12.9	103.08
A3	31	HYBRID	1,276	12,182	74,022	0	0	18.7	0.0	-45	2	102	3.2	0.0	0.2	1.1	16.67
		NORINS	1,276	12,182	74,022	8	0	18.7	0.0	-45	10	50	5.4	0.0	0.0	0.6	13.35
A3	32	HYBRID	1,288	12,501	76,002	0	0	6.8	0.0	-68	27	78	2.9	0.0	0.2	0.7	11.49
		NORINS	1,288	12,501	76,002	0	0	6.8	0.0	-68	39	82	3.4	0.0	0.0	0.9	13.37
A3	33	HYBRID	1,288	12,502	76,009	0	0	5.8	0.0	-86	108	260	2.2	0.0	0.9	2.1	20.65
		NORINS	1,288	12,502	76,009	0	0	5.8	0.0	-86	188	283	2.5	0.0	0.0	3.1	26.18
A3	34	HYBRID	1,255	11,872	121,108	978	0	23.9	0.0	-36	19	48	22.0	0.0	0.2	0.9	58.18
		NORINS	1,255	11,872	121,108	452	0	23.9	0.0	-36	51	71	22.4	0.0	0.0	1.2	51.01
A3	35	HYBRID	1,293	12,498	127,513	0	0	18.2	0.0	-57	30	2,524	6.6	0.0	9.1	38.0	367.15
		NORINS	1,293	12,498	127,513	0	0	18.2	0.0	-57	110	1,116	7.4	0.0	0.0	16.8	126.80
A3	36	HYBRID	1,293	12,502	127,555	0	0	10.9	0.0	-77	54	3,191	5.7	0.0	15.4	40.6	464.06
		NORINS	1,293	12,502	127,555	0	0	10.9	0.0	-77	60	1,371	6.4	0.0	0.0	18.3	121.26
C2	1	HYBRID	101	898	2,571	0	365	6.4	0.0	-2,119	7	4,646	0.0	0.2	3.2	1.4	9.78
		NORINS	101	898	2,571	0	563	6.4	0.0	-2,119	7,592	7,815	0.0	0.4	0.0	2.7	11.89
C2	2	HYBRID	101	879	2,515	0	596	8.1	0.0	-1,868	1,795	9,209	0.0	0.4	4.8	2.9	20.28
		NORINS	101	879	2,515	0	522	8.1	0.0	-1,868	2,889	7,378	0.0	0.3	0.0	2.6	9.98
C2	5	HYBRID	110	907	9,860	0	78	15.2	0.0	-1,515	2,065	2,805	0.0	0.1	1.8	2.9	8.04
		NORINS	110	907	9,860	0	65	15.2	0.0	-1,515	2,160	2,670	0.0	0.1	0.0	2.7	5.49

◇ Problem is infeasible

Table 25 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C2	6	HYBRID	110	868	9,367	0	97	19.3	0.0	-1,279	456	3,737	0.0	0.1	1.9	3.9	10.38
		NORINS	110	868	9,367	0	80	19.3	0.0	-1,279	1,963	3,776	0.0	0.1	0.0	4.0	8.16
C2	9	HYBRID	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		NORINS	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
C2	10	HYBRID	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		NORINS	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
C2	13	HYBRID	20	11	109	0	0	0.0	0.0	-593	0	0	0.0	0.0	0.0	0.0	0.00
		NORINS	20	11	109	0	0	0.0	0.0	-593	0	0	0.0	0.0	0.0	0.0	0.00
C2	14	HYBRID	67	94	862	0	0	◇	◇	◇	◇	0	0.0	0.0	0.0	0.0	0.00
		NORINS	67	94	862	0	0	◇	◇	◇	◇	0	0.0	0.0	0.0	0.0	0.00
C2	17	HYBRID	501	4,812	13,893	0	95	1.1	0.0	-40,731	10	1,668	0.0	0.3	25.3	2.7	33.53
		NORINS	501	4,812	13,893	0	131	1.1	0.0	-40,731	1,877	2,267	0.0	0.6	0.0	6.2	17.75
C2	18	HYBRID	501	4,809	13,884	0	103	1.1	0.0	-36,549	242	2,159	0.0	0.4	24.2	3.7	34.90
		NORINS	501	4,809	13,884	0	162	1.1	0.0	-36,549	3,421	3,908	0.0	0.9	0.0	8.5	22.14
C2	21	HYBRID	197	358	3,298	0	0	12.1	0.0	-4,221	6	19	0.0	0.0	0.0	0.0	0.08
		NORINS	197	358	3,298	0	0	12.1	0.0	-4,221	6	12	0.0	0.0	0.0	0.0	0.07
C2	22	HYBRID	72	94	908	0	0	6.2	0.0	-3,657	0	2	0.0	0.0	0.0	0.0	0.01
		NORINS	72	94	908	0	0	6.2	0.0	-3,657	2	2	0.0	0.0	0.0	0.0	0.01
C3	1	HYBRID	505	4,968	30,162	0	3	29.5	0.0	-43	0	685	0.0	0.1	0.4	2.2	23.91
		NORINS	505	4,968	30,162	0	4	29.5	0.0	-43	452	729	0.0	0.2	0.0	2.5	25.93
C3	2	HYBRID	505	4,968	30,162	0	39	12.7	0.0	-81	427	4,885	0.0	0.8	4.0	15.5	142.87
		NORINS	505	4,968	30,162	0	37	12.7	0.0	-81	2,383	3,814	0.0	0.8	0.0	10.1	62.72
C3	3	HYBRID	505	4,968	30,162	0	116	9.7	0.0	-114	1,980	21,139	0.0	3.8	29.1	69.7	621.57
		NORINS	505	4,968	30,162	0	96	9.7	0.0	-114	1,817	13,042	0.0	2.4	0.0	31.6	164.83
C3	4	HYBRID	510	4,968	50,496	0	0	◇	◇	◇	◇	86	0.0	0.0	0.0	0.4	14.18
		NORINS	510	4,968	50,496	0	0	◇	◇	◇	◇	84	0.0	0.0	0.0	0.4	13.73
C3	5	HYBRID	510	4,968	50,496	0	24	20.5	0.0	-59	129	5,491	0.0	1.0	7.6	26.9	349.24
		NORINS	510	4,968	50,496	0	20	20.5	0.0	-59	1,184	3,786	0.0	0.7	0.0	13.6	122.63
C3	6	HYBRID	510	4,968	50,496	0	57	11.8	0.0	-88	4	15,832	0.0	2.9	26.4	77.1	939.86
		NORINS	510	4,968	50,496	0	49	11.8	0.0	-88	2,346	10,093	0.0	1.8	0.0	31.1	224.25
C3	7	HYBRID	505	4,963	30,094	0	3	22.8	0.0	-41	103	215	0.0	0.0	0.1	0.7	7.12
		NORINS	505	4,963	30,094	0	3	22.8	0.0	-41	120	195	0.0	0.0	0.0	0.6	5.45
C3	8	HYBRID	505	4,963	30,094	0	27	10.1	0.0	-82	126	1,950	0.0	0.4	1.6	6.2	49.91
		NORINS	505	4,963	30,094	0	31	10.1	0.0	-82	1,700	2,491	0.0	0.6	0.0	8.7	55.04

◇ Problem is infeasible

Table 25 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C3	9	HYBRID	505	4,963	30,094	0	42	7.3	0.0	-119	148	6,679	0.0	1.2	8.2	20.9	165.91
		NORINS	505	4,963	30,094	0	61	7.3	0.0	-119	5,454	7,028	0.0	1.3	0.0	18.1	104.67
C3	10	HYBRID	510	4,963	50,370	0	0	◇	◇	◇	◇	30	0.0	0.0	0.0	0.2	6.85
		NORINS	510	4,963	50,370	0	0	◇	◇	◇	◇	30	0.0	0.0	0.0	0.1	6.72
C3	11	HYBRID	510	4,963	50,370	0	25	18.0	0.0	-56	264	4,661	0.0	0.8	6.5	22.1	275.50
		NORINS	510	4,963	50,370	0	32	18.0	0.0	-56	1,444	3,693	0.0	0.7	0.0	13.9	127.15
C3	12	HYBRID	510	4,963	50,370	0	90	12.0	0.0	-83	247	28,503	0.0	5.0	46.9	136.3	1,589.87
		NORINS	510	4,963	50,370	0	97	12.0	0.0	-83	10,234	26,420	0.0	5.2	0.0	102.1	739.39
C3	13	HYBRID	755	7,524	45,856	0	0	18.5	0.0	-35	0	7	0.0	0.0	0.0	0.0	0.93
		NORINS	755	7,524	45,856	0	0	18.5	0.0	-35	0	6	0.0	0.0	0.0	0.0	0.90
C3	14	HYBRID	755	7,524	45,856	0	9	10.1	0.0	-73	67	951	0.0	0.2	0.9	4.4	54.24
		NORINS	755	7,524	45,856	0	6	10.1	0.0	-73	100	492	0.0	0.1	0.0	2.1	25.60
C3	15	HYBRID	755	7,524	45,856	0	50	8.3	0.0	-105	1,127	7,298	0.0	2.1	9.1	37.3	406.76
		NORINS	755	7,524	45,856	0	55	8.3	0.0	-105	2,783	5,993	0.0	1.9	0.0	28.7	206.75
C3	16	HYBRID	760	7,524	76,691	0	0	◇	◇	◇	◇	2	0.0	0.0	0.0	0.0	3.98
		NORINS	760	7,524	76,691	0	0	◇	◇	◇	◇	2	0.0	0.0	0.0	0.0	3.79
C3	17	HYBRID	760	7,524	76,691	0	10	21.5	0.0	-57	28	3,225	0.0	0.9	5.1	24.1	580.08
		NORINS	760	7,524	76,691	0	6	21.5	0.0	-57	1,199	2,355	0.0	0.7	0.0	14.0	157.89
C3	18	HYBRID	760	7,524	76,691	0	92	14.8	2.5	-85	387	20,478	0.0	6.8	52.6	180.7	3,600.00
		NORINS	760	7,524	76,691	0	82	14.8	0.0	-85	1,800	14,515	0.0	4.1	0.0	73.2	757.27
C3	19	HYBRID	755	7,595	46,279	0	2	34.4	0.0	-38	404	944	0.0	0.3	0.7	4.7	52.42
		NORINS	755	7,595	46,279	0	3	34.4	0.0	-38	506	560	0.0	0.2	0.0	2.1	20.45
C3	20	HYBRID	755	7,595	46,279	0	18	12.7	0.0	-80	0	1,693	0.0	0.4	1.7	8.1	84.71
		NORINS	755	7,595	46,279	0	20	12.7	0.0	-80	400	1,208	0.0	0.4	0.0	5.5	52.64
C3	21	HYBRID	755	7,595	46,279	0	47	8.0	0.0	-117	0	3,294	0.0	0.9	3.7	15.7	159.44
		NORINS	755	7,595	46,279	0	59	8.0	0.0	-117	928	2,524	0.0	0.7	0.0	11.3	101.47
C3	22	HYBRID	760	7,595	77,359	0	0	◇	◇	◇	◇	68	0.0	0.0	0.0	0.5	32.44
		NORINS	760	7,595	77,359	0	0	◇	◇	◇	◇	68	0.0	0.0	0.0	0.5	27.61
C3	23	HYBRID	760	7,595	77,359	0	2	11.2	0.0	-62	68	321	0.0	0.1	0.6	2.6	43.77
		NORINS	760	7,595	77,359	0	1	11.2	0.0	-62	117	256	0.0	0.1	0.0	1.5	21.05
C3	24	HYBRID	760	7,595	77,359	0	83	15.2	5.2	-85	1,269	18,891	0.0	8.0	58.4	215.3	3,600.00
		NORINS	760	7,595	77,359	0	121	15.2	0.0	-85	12,345	49,779	0.0	14.3	0.0	264.1	2,700.41
C3	25	HYBRID	1,003	9,960	60,766	0	0	◇	◇	◇	◇	34	0.0	0.0	0.0	0.2	16.58
		NORINS	1,003	9,960	60,766	0	0	◇	◇	◇	◇	32	0.0	0.0	0.0	0.2	14.70

◇ Problem is infeasible

Table 25 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C3	26	HYBRID	1,003	9,960	60,766	0	9	16.5	0.0	-69	247	1,417	0.0	0.5	1.8	9.4	155.58
		NORINS	1,003	9,960	60,766	0	10	16.5	0.0	-69	1,337	1,853	0.0	0.9	0.0	13.5	135.12
C3	27	HYBRID	1,003	9,960	60,766	0	78	11.1	0.0	-106	5,070	9,856	0.0	4.3	16.5	79.0	1,401.53
		NORINS	1,003	9,960	60,766	0	83	11.1	0.0	-106	6,369	9,287	0.0	4.1	0.0	60.5	646.27
C3	28	HYBRID	1,008	9,960	101,846	0	0	◇	◇	◇	◇	4	0.0	0.0	0.0	0.0	14.63
		NORINS	1,008	9,960	101,846	0	0	◇	◇	◇	◇	4	0.0	0.0	0.0	0.0	12.44
C3	29	HYBRID	1,008	9,960	101,846	0	7	26.8	0.0	-49	2	6,069	0.0	2.3	11.2	61.6	2,125.68
		NORINS	1,008	9,960	101,846	0	8	26.8	0.0	-49	800	3,569	0.0	1.5	0.0	29.2	377.30
C3	30	HYBRID	1,008	9,960	101,846	0	28	17.4	8.6	-77	11	10,848	0.0	6.8	46.7	182.3	3,600.00
		NORINS	1,008	9,960	101,846	0	83	17.4	5.7	-77	6,308	31,223	0.0	14.8	0.0	297.7	3,600.00
C3	31	HYBRID	1,004	10,096	61,483	0	0	25.4	0.0	-40	114	248	0.0	0.1	0.3	1.7	27.76
		NORINS	1,004	10,096	61,483	0	0	25.4	0.0	-40	129	200	0.0	0.1	0.0	1.3	22.76
C3	32	HYBRID	1,004	10,096	61,483	0	25	13.5	0.0	-75	24	6,379	0.0	2.4	8.9	43.2	614.29
		NORINS	1,004	10,096	61,483	0	23	13.5	0.0	-75	3,341	5,518	0.0	2.1	0.0	28.0	221.25
C3	33	HYBRID	1,004	10,096	61,483	0	40	7.7	0.0	-111	693	9,474	0.0	3.5	15.5	65.5	758.80
		NORINS	1,004	10,096	61,483	0	42	7.7	0.0	-111	2,496	6,721	0.0	2.9	0.0	43.4	308.62
C3	34	HYBRID	1,009	10,096	103,118	0	0	◇	◇	◇	◇	82	0.0	0.0	0.0	0.9	69.29
		NORINS	1,009	10,096	103,118	0	0	◇	◇	◇	◇	84	0.0	0.0	0.0	0.9	58.58
C3	35	HYBRID	1,009	10,096	103,118	0	13	17.9	0.0	-59	400	2,544	0.0	1.0	6.1	27.3	704.93
		NORINS	1,009	10,096	103,118	0	28	17.9	0.0	-59	3,242	4,120	0.0	1.9	0.0	41.5	530.26
C3	36	HYBRID	1,009	10,096	103,118	0	80	12.9	3.7	-86	1,228	11,631	0.0	6.1	40.1	167.0	3,600.00
		NORINS	1,009	10,096	103,118	0	150	12.9	0.0	-86	15,697	26,110	0.0	12.0	0.0	272.6	3,498.99

◇ Problem is infeasible

Table 26: RCSPP cutting plane results

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A2	3	ALL	99	935	2,655	0	5	1.0	0.0	-78	15	15	0.0	0.0	0.0	0.0	0.06
		ALL-NDP	99	935	2,655	0	5	1.0	0.0	-78	15	15	0.0	0.0	0.0	0.0	0.07
		ALL-SPP	99	935	2,655	0	1	1.0	0.0	-78	14	14	0.0	0.0	0.0	0.0	0.06
		ALL-CTP	99	935	2,655	0	5	1.0	0.0	-78	15	15	0.0	0.0	0.0	0.0	0.06
		ALL-CLQ	99	935	2,655	0	5	1.0	0.0	-78	15	15	0.0	0.0	0.0	0.0	0.06
		ALL-LOH	99	935	2,655	0	5	1.0	0.0	-78	15	15	0.0	0.0	0.0	0.0	0.06
		ALL-LPS	99	935	2,655	0	4	1.0	0.0	-78	17	17	0.0	0.0	0.0	0.0	0.06
A2	4	ALL	99	927	2,632	0	10	0.2	0.0	-74	2	2	0.0	0.0	0.0	0.0	0.07
		ALL-NDP	99	927	2,632	0	10	0.2	0.0	-74	2	2	0.0	0.0	0.0	0.0	0.07
		ALL-SPP	99	927	2,632	0	2	0.2	0.0	-74	2	2	0.0	0.0	0.0	0.0	0.07
		ALL-CTP	99	927	2,632	0	10	0.2	0.0	-74	2	2	0.0	0.0	0.0	0.0	0.06
		ALL-CLQ	99	927	2,632	0	10	0.2	0.0	-74	2	2	0.0	0.0	0.0	0.0	0.07
		ALL-LOH	99	927	2,632	0	10	0.2	0.0	-74	2	2	0.0	0.0	0.0	0.0	0.07
		ALL-LPS	99	927	2,632	0	4	0.3	0.0	-74	6	6	0.0	0.0	0.0	0.0	0.05
A2	7	ALL	89	708	7,405	141	25	0.0	0.0	-29	0	0	0.2	0.1	0.0	0.0	0.35
		ALL-NDP	89	708	7,405	141	25	0.0	0.0	-29	0	0	0.2	0.1	0.0	0.0	0.35
		ALL-SPP	89	708	7,405	125	23	0.0	0.0	-29	0	0	0.2	0.1	0.0	0.0	0.37
		ALL-CTP	89	708	7,405	141	22	0.0	0.0	-29	0	0	0.2	0.1	0.0	0.0	0.34
		ALL-CLQ	89	708	7,405	141	23	0.0	0.0	-29	0	0	0.2	0.1	0.0	0.0	0.35
		ALL-LOH	89	708	7,405	140	21	0.0	0.0	-29	0	0	0.2	0.0	0.0	0.0	0.34
		ALL-LPS	89	708	7,405	140	8	0.0	0.0	-29	0	0	0.2	0.0	0.0	0.0	0.31
A2	8	ALL	50	157	1,476	44	0	0.0	0.0	-23	0	0	0.1	0.0	0.0	0.0	0.07
		ALL-NDP	50	157	1,476	44	0	0.0	0.0	-23	0	0	0.1	0.0	0.0	0.0	0.07
		ALL-SPP	50	157	1,476	44	0	0.0	0.0	-23	0	0	0.1	0.0	0.0	0.0	0.07
		ALL-CTP	50	157	1,476	44	0	0.0	0.0	-23	0	0	0.1	0.0	0.0	0.0	0.07
		ALL-CLQ	50	157	1,476	44	0	0.0	0.0	-23	0	0	0.1	0.0	0.0	0.0	0.07
		ALL-LOH	50	157	1,476	44	0	0.0	0.0	-23	0	0	0.1	0.0	0.0	0.0	0.07
		ALL-LPS	50	157	1,476	44	0	0.0	0.0	-23	0	0	0.1	0.0	0.0	0.0	0.07
A2	11	ALL	195	1,915	5,471	0	10	0.0	0.0	-125	5	5	0.0	0.1	0.0	0.0	0.23
		ALL-NDP	195	1,915	5,471	0	10	0.0	0.0	-125	5	5	0.0	0.1	0.0	0.0	0.23
		ALL-SPP	195	1,915	5,471	0	2	0.0	0.0	-125	1	1	0.0	0.1	0.0	0.0	0.22
		ALL-CTP	195	1,915	5,471	0	10	0.0	0.0	-125	5	5	0.0	0.1	0.0	0.0	0.18
		ALL-CLQ	195	1,915	5,471	0	10	0.0	0.0	-125	5	5	0.0	0.1	0.0	0.0	0.23
		ALL-LOH	195	1,915	5,471	0	10	0.0	0.0	-125	5	5	0.0	0.1	0.0	0.0	0.23
		ALL-LPS	195	1,915	5,471	0	8	0.1	0.0	-125	1	1	0.0	0.1	0.0	0.0	0.18
A2	12	ALL	195	1,915	5,471	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.07
		ALL-NDP	195	1,915	5,471	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.07
		ALL-SPP	195	1,915	5,471	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.07
		ALL-CTP	195	1,915	5,471	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.07
		ALL-CLQ	195	1,915	5,471	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.07
		ALL-LOH	195	1,915	5,471	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.07
		ALL-LPS	195	1,915	5,471	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.07

◇ Problem is infeasible

Table 26 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A2	15	ALL	151	1,179	12,201	711	15	0.0	0.0	-24	0	0	0.5	0.1	0.0	0.0	0.86
		ALL-NDP	151	1,179	12,201	711	14	0.0	0.0	-24	0	0	0.4	0.1	0.0	0.0	0.84
		ALL-SPP	151	1,179	12,201	711	15	0.0	0.0	-24	0	0	0.5	0.1	0.0	0.0	0.85
		ALL-CTP	151	1,179	12,201	749	16	0.0	0.0	-24	0	0	0.5	0.1	0.0	0.0	0.87
		ALL-CLQ	151	1,179	12,201	711	11	0.0	0.0	-24	0	0	0.5	0.1	0.0	0.0	0.85
		ALL-LOH	151	1,179	12,201	711	10	0.0	0.0	-24	0	0	0.5	0.1	0.0	0.0	0.86
		ALL-LPS	151	1,179	12,201	742	13	0.0	0.0	-24	0	0	0.5	0.1	0.0	0.0	0.85
A2	16	ALL	22	17	161	7	6	0.0	0.0	-17	0	0	0.1	0.0	0.0	0.0	0.08
		ALL-NDP	22	17	161	7	4	0.0	0.0	-17	0	0	0.1	0.0	0.0	0.0	0.08
		ALL-SPP	22	17	161	7	5	0.0	0.0	-17	0	0	0.1	0.0	0.0	0.0	0.08
		ALL-CTP	22	17	161	7	4	0.0	0.0	-17	0	0	0.1	0.0	0.0	0.0	0.09
		ALL-CLQ	22	17	161	7	6	0.0	0.0	-17	0	0	0.1	0.0	0.0	0.0	0.08
		ALL-LOH	22	17	161	7	6	0.0	0.0	-17	0	0	0.1	0.0	0.0	0.0	0.08
		ALL-LPS	22	17	161	7	5	0.0	0.0	-17	0	0	0.1	0.0	0.0	0.0	0.09
A2	19	ALL	474	4,682	13,373	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.23
		ALL-NDP	474	4,682	13,373	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.23
		ALL-SPP	474	4,682	13,373	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.24
		ALL-CTP	474	4,682	13,373	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.23
		ALL-CLQ	474	4,682	13,373	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.23
		ALL-LOH	474	4,682	13,373	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.23
		ALL-LPS	474	4,682	13,373	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.23
A2	20	ALL	474	4,668	13,331	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.22
		ALL-NDP	474	4,668	13,331	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.22
		ALL-SPP	474	4,668	13,331	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.22
		ALL-CTP	474	4,668	13,331	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.22
		ALL-CLQ	474	4,668	13,331	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.22
		ALL-LOH	474	4,668	13,331	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.22
		ALL-LPS	474	4,668	13,331	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.22
A2	23	ALL	471	4,471	48,353	15	4	24.8	0.0	-36	100	120	2.0	0.5	0.0	1.3	7.04
		ALL-NDP	471	4,471	48,353	15	4	24.8	0.0	-36	100	120	2.0	0.6	0.0	1.3	7.16
		ALL-SPP	471	4,471	48,353	15	4	24.8	0.0	-36	100	120	2.0	0.6	0.0	1.3	7.10
		ALL-CTP	471	4,471	48,353	15	4	24.8	0.0	-36	100	120	2.0	0.6	0.0	1.3	7.06
		ALL-CLQ	471	4,471	48,353	15	7	24.8	0.0	-36	60	119	2.0	0.6	0.0	1.4	8.23
		ALL-LOH	471	4,471	48,353	15	0	24.8	0.0	-36	64	86	2.0	0.6	0.0	1.0	6.06
		ALL-LPS	471	4,471	48,353	15	4	24.8	0.0	-36	100	120	2.0	0.4	0.0	1.4	7.03
A2	24	ALL	414	3,532	37,812	1,644	28	0.0	0.0	-29	0	0	2.4	0.6	0.0	0.0	5.95
		ALL-NDP	414	3,532	37,812	1,644	28	0.0	0.0	-29	0	0	2.4	0.6	0.0	0.0	5.95
		ALL-SPP	414	3,532	37,812	1,740	19	2.3	0.0	-29	1	1	2.5	0.6	0.0	0.0	6.11
		ALL-CTP	414	3,532	37,812	1,653	24	0.0	0.0	-29	0	0	2.5	0.6	0.0	0.0	5.99
		ALL-CLQ	414	3,532	37,812	1,674	26	0.0	0.0	-29	0	0	2.5	0.6	0.0	0.0	6.10
		ALL-LOH	414	3,532	37,812	1,704	16	0.0	0.0	-29	0	0	2.5	0.6	0.0	0.0	6.10
		ALL-LPS	414	3,532	37,812	2,685	24	4.8	0.0	-29	5	5	2.6	0.4	0.0	0.0	6.71

◇ Problem is infeasible

Table 26 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	1	ALL	861	8,335	50,727	4	13	14.5	0.0	-44	29	70	1.9	1.5	0.0	0.4	6.11
		ALL-SPP	861	8,335	50,727	4	13	14.5	0.0	-44	29	70	1.8	1.5	0.0	0.4	6.09
		ALL-CTP	861	8,335	50,727	4	13	14.5	0.0	-44	29	70	1.8	1.5	0.0	0.4	6.16
		ALL-CLQ	861	8,335	50,727	4	11	14.5	0.0	-44	25	82	1.8	1.4	0.0	0.5	6.31
		ALL-LOH	861	8,335	50,727	4	2	14.5	0.0	-44	33	72	1.8	1.4	0.0	0.4	6.16
		ALL-LPS	861	8,335	50,727	4	3	14.5	0.0	-44	29	72	1.8	1.2	0.0	0.4	5.86
A3	2	ALL	876	8,588	52,287	0	22	11.6	0.0	-64	618	748	1.4	1.7	0.0	4.3	20.67
		ALL-SPP	876	8,588	52,287	0	16	11.8	0.0	-64	100	438	1.2	1.5	0.0	2.6	14.12
		ALL-CTP	876	8,588	52,287	0	22	11.6	0.0	-64	618	748	1.4	1.6	0.0	4.3	20.72
		ALL-CLQ	876	8,588	52,287	0	36	11.6	0.0	-64	566	695	1.4	1.7	0.0	4.1	20.39
		ALL-LOH	876	8,588	52,287	0	3	11.6	0.0	-64	370	610	1.3	1.6	0.0	3.7	19.77
		ALL-LPS	876	8,588	52,287	0	26	11.7	0.0	-64	244	463	1.1	1.2	0.0	2.7	14.23
A3	3	ALL	876	8,596	52,337	0	6	6.7	0.0	-85	30	111	0.8	1.4	0.0	0.6	6.37
		ALL-SPP	876	8,596	52,337	0	6	6.7	0.0	-85	30	111	0.8	1.4	0.0	0.6	6.33
		ALL-CTP	876	8,596	52,337	0	6	6.7	0.0	-85	30	111	0.8	1.3	0.0	0.6	6.29
		ALL-CLQ	876	8,596	52,337	0	13	6.7	0.0	-85	33	103	0.8	1.4	0.0	0.6	6.68
		ALL-LOH	876	8,596	52,337	0	0	6.7	0.0	-85	30	98	0.8	1.4	0.0	0.6	6.06
		ALL-LPS	876	8,596	52,337	0	6	6.7	0.0	-85	30	111	0.8	1.3	0.0	0.6	6.22
A3	4	ALL	852	8,136	82,674	298	9	27.3	0.0	-32	59	123	7.6	2.2	0.0	1.1	24.81
		ALL-SPP	852	8,136	82,674	298	9	27.3	0.0	-32	59	123	7.6	2.2	0.0	1.1	24.87
		ALL-CTP	852	8,136	82,674	298	9	27.3	0.0	-32	59	123	7.6	2.2	0.0	1.1	24.99
		ALL-CLQ	852	8,136	82,674	298	10	27.3	0.0	-32	54	123	7.5	2.1	0.0	1.1	24.86
		ALL-LOH	852	8,136	82,674	298	1	27.3	0.0	-32	50	122	7.5	2.0	0.0	1.1	23.91
		ALL-LPS	852	8,136	82,674	211	1	27.3	0.0	-32	50	115	7.4	1.6	0.0	1.1	23.48
A3	5	ALL	881	8,588	87,286	0	3	16.2	0.0	-54	203	1,546	2.8	2.0	0.0	12.7	70.97
		ALL-SPP	881	8,588	87,286	0	3	16.2	0.0	-54	203	1,546	2.8	2.0	0.0	12.6	71.12
		ALL-CTP	881	8,588	87,286	0	3	16.2	0.0	-54	203	1,546	2.9	2.0	0.0	12.8	71.34
		ALL-CLQ	881	8,588	87,286	0	7	16.2	0.0	-54	187	1,547	2.8	2.0	0.0	12.8	77.00
		ALL-LOH	881	8,588	87,286	0	0	16.2	0.0	-54	180	1,503	2.8	2.0	0.0	12.3	71.26
		ALL-LPS	881	8,588	87,286	0	3	16.2	0.0	-54	203	1,546	2.8	1.7	0.0	12.6	70.30
A3	6	ALL	881	8,596	87,365	0	6	12.4	0.0	-73	59	2,494	1.8	2.1	0.0	17.7	80.67
		ALL-SPP	881	8,596	87,365	0	6	12.4	0.0	-73	59	2,494	1.9	2.0	0.0	17.7	80.74
		ALL-CTP	881	8,596	87,365	0	6	12.4	0.0	-73	59	2,494	1.8	2.0	0.0	17.7	80.64
		ALL-CLQ	881	8,596	87,365	0	9	12.4	0.0	-73	206	2,587	1.9	2.1	0.0	18.5	85.85
		ALL-LOH	881	8,596	87,365	0	0	12.4	0.0	-73	628	2,793	1.8	2.0	0.0	20.2	100.99
		ALL-LPS	881	8,596	87,365	0	6	12.4	0.0	-73	59	2,494	1.8	1.7	0.0	17.7	80.13
A3	7	ALL	878	8,572	52,137	0	17	16.5	0.0	-44	75	135	1.4	1.5	0.0	0.9	7.72
		ALL-SPP	878	8,572	52,137	0	6	18.4	0.0	-44	15	96	1.5	1.5	0.0	0.6	7.10
		ALL-CTP	878	8,572	52,137	0	17	16.5	0.0	-44	75	135	1.5	1.4	0.0	0.9	7.68
		ALL-CLQ	878	8,572	52,137	0	24	16.5	0.0	-44	76	143	1.5	1.5	0.0	0.9	7.83
		ALL-LOH	878	8,572	52,137	0	9	16.5	0.0	-44	100	145	1.6	1.7	0.0	0.9	8.41
		ALL-LPS	878	8,572	52,137	0	15	16.5	0.0	-44	47	115	1.6	1.2	0.0	0.8	7.38

◇ Problem is infeasible

Table 26 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	8	ALL	883	8,712	53,010	0	48	7.1	0.0	-70	89	160	1.2	1.9	0.0	1.0	9.85
		ALL-SPP	883	8,712	53,010	0	5	8.4	0.0	-70	110	173	1.0	1.3	0.0	1.0	8.49
		ALL-CTP	883	8,712	53,010	0	48	7.1	0.0	-70	89	160	1.2	1.8	0.0	1.0	9.79
		ALL-CLQ	883	8,712	53,010	0	41	7.1	0.0	-70	75	145	1.2	1.9	0.0	0.8	8.15
		ALL-LOH	883	8,712	53,010	0	20	7.1	0.0	-70	104	171	1.2	1.8	0.0	1.0	8.91
		ALL-LPS	883	8,712	53,010	0	48	7.1	0.0	-70	89	160	1.2	1.5	0.0	1.0	9.43
A3	9	ALL	883	8,712	53,010	0	23	5.9	0.0	-89	380	467	0.7	1.6	0.0	2.5	13.26
		ALL-SPP	883	8,712	53,010	0	7	6.0	0.0	-89	9	196	0.7	1.4	0.0	1.1	6.40
		ALL-CTP	883	8,712	53,010	0	23	5.9	0.0	-89	380	467	0.7	1.5	0.0	2.6	13.25
		ALL-CLQ	883	8,712	53,010	0	27	5.9	0.0	-89	518	658	0.7	1.5	0.0	3.6	19.49
		ALL-LOH	883	8,712	53,010	0	3	5.9	0.0	-89	533	533	0.7	1.5	0.0	2.9	13.13
		ALL-LPS	883	8,712	53,010	0	23	5.9	0.0	-89	380	467	0.7	1.3	0.0	2.5	13.03
A3	10	ALL	865	8,385	85,175	12	1	40.1	0.0	-33	20	416	5.4	1.9	0.0	4.0	41.17
		ALL-SPP	865	8,385	85,175	12	1	40.1	0.0	-33	20	416	5.4	1.9	0.0	4.1	41.50
		ALL-CTP	865	8,385	85,175	12	1	40.1	0.0	-33	20	416	5.4	1.9	0.0	4.1	41.53
		ALL-CLQ	865	8,385	85,175	12	2	40.1	0.0	-33	27	395	5.3	1.9	0.0	4.0	41.23
		ALL-LOH	865	8,385	85,175	14	0	40.1	0.0	-33	20	419	5.4	1.9	0.0	4.1	41.67
		ALL-LPS	865	8,385	85,175	12	1	40.1	0.0	-33	20	416	5.4	1.7	0.0	4.1	41.26
A3	11	ALL	888	8,712	88,550	0	4	15.1	0.0	-60	150	867	2.5	2.0	0.0	7.3	50.45
		ALL-SPP	888	8,712	88,550	0	4	15.1	0.0	-60	150	867	2.5	2.0	0.0	7.3	50.65
		ALL-CTP	888	8,712	88,550	0	4	15.1	0.0	-60	150	867	2.5	2.0	0.0	7.3	50.08
		ALL-CLQ	888	8,712	88,550	0	9	15.1	0.0	-60	296	929	2.5	2.0	0.0	7.7	55.34
		ALL-LOH	888	8,712	88,550	0	0	15.1	0.0	-60	280	910	2.5	2.0	0.0	7.7	54.05
		ALL-LPS	888	8,712	88,550	0	4	15.1	0.0	-60	150	867	2.5	1.8	0.0	7.2	49.83
A3	12	ALL	888	8,712	88,550	0	39	11.6	0.0	-79	3,902	4,635	2.1	2.5	0.0	33.9	164.65
		ALL-SPP	888	8,712	88,550	0	17	11.7	0.0	-79	1,147	3,393	1.8	2.8	0.0	24.7	135.63
		ALL-CTP	888	8,712	88,550	0	39	11.6	0.0	-79	3,902	4,635	2.0	2.4	0.0	33.9	165.01
		ALL-CLQ	888	8,712	88,550	0	47	11.6	0.0	-79	1,644	3,826	2.1	2.5	0.0	28.3	167.40
		ALL-LOH	888	8,712	88,550	0	33	11.6	0.0	-79	1,630	3,704	2.0	2.5	0.0	27.2	148.90
		ALL-LPS	888	8,712	88,550	0	35	11.6	0.0	-79	4,429	5,177	2.1	1.8	0.0	38.1	185.41
A3	13	ALL	1,085	10,312	62,518	78	20	19.3	0.0	-42	120	126	3.4	2.3	0.0	1.0	12.84
		ALL-SPP	1,085	10,312	62,518	78	20	19.3	0.0	-42	120	126	3.4	2.2	0.0	1.0	12.79
		ALL-CTP	1,085	10,312	62,518	78	20	19.3	0.0	-42	120	126	3.4	2.3	0.0	1.0	12.92
		ALL-CLQ	1,085	10,312	62,518	80	8	19.4	0.0	-42	49	72	3.2	1.9	0.0	0.6	9.83
		ALL-LOH	1,085	10,312	62,518	77	2	19.4	0.0	-42	46	81	3.2	2.0	0.0	0.7	10.65
		ALL-LPS	1,085	10,312	62,518	78	20	19.3	0.0	-42	120	126	3.4	1.8	0.0	1.0	12.37
A3	14	ALL	1,098	10,671	64,763	0	6	10.0	0.0	-67	9	76	1.3	2.0	0.0	0.5	8.35
		ALL-SPP	1,098	10,671	64,763	0	6	10.0	0.0	-67	9	76	1.3	2.0	0.0	0.5	8.29
		ALL-CTP	1,098	10,671	64,763	0	6	10.0	0.0	-67	9	76	1.3	2.0	0.0	0.5	8.31
		ALL-CLQ	1,098	10,671	64,763	0	11	10.0	0.0	-67	8	77	1.3	2.0	0.0	0.5	8.19
		ALL-LOH	1,098	10,671	64,763	0	0	10.0	0.0	-67	11	76	1.3	2.0	0.0	0.5	8.04
		ALL-LPS	1,098	10,671	64,763	0	6	10.0	0.0	-67	9	76	1.4	1.8	0.0	0.5	8.13

◇ Problem is infeasible

Table 26 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	15	ALL	1,098	10,671	64,763	0	8	5.9	0.0	-89	20	53	1.1	2.1	0.0	0.4	7.70
		ALL-SPP	1,098	10,671	64,763	0	8	5.9	0.0	-89	20	53	1.1	2.1	0.0	0.4	7.62
		ALL-CTP	1,098	10,671	64,763	0	8	5.9	0.0	-89	20	53	1.1	2.0	0.0	0.4	7.82
		ALL-CLQ	1,098	10,671	64,763	0	13	5.9	0.0	-89	20	52	1.1	2.1	0.0	0.4	7.81
		ALL-LOH	1,098	10,671	64,763	0	0	5.9	0.0	-89	30	77	1.1	2.1	0.0	0.6	7.58
		ALL-LPS	1,098	10,671	64,763	0	8	5.9	0.0	-89	20	53	1.1	1.9	0.0	0.4	7.46
A3	16	ALL	1,033	9,720	98,838	814	11	26.6	0.0	-32	30	73	11.9	2.9	0.0	0.9	36.47
		ALL-SPP	1,033	9,720	98,838	243	4	28.9	0.0	-32	40	136	8.5	2.9	0.0	1.5	33.65
		ALL-CTP	1,033	9,720	98,838	814	11	26.6	0.0	-32	30	73	11.8	2.9	0.0	0.9	36.34
		ALL-CLQ	1,033	9,720	98,838	809	10	26.6	0.0	-32	24	74	11.9	2.9	0.0	0.8	36.78
		ALL-LOH	1,033	9,720	98,838	370	11	26.2	0.0	-32	64	128	9.8	3.9	0.0	1.4	39.70
		ALL-LPS	1,033	9,720	98,838	816	10	26.6	0.0	-32	13	67	11.9	2.6	0.0	0.7	34.82
A3	17	ALL	1,103	10,671	108,673	0	7	22.3	0.0	-55	60	1,506	4.2	3.1	0.0	15.5	113.08
		ALL-SPP	1,103	10,671	108,673	0	7	22.3	0.0	-55	60	1,506	4.2	3.1	0.0	15.5	114.11
		ALL-CTP	1,103	10,671	108,673	0	7	22.3	0.0	-55	60	1,506	4.2	3.1	0.0	15.5	114.09
		ALL-CLQ	1,103	10,671	108,673	0	7	22.3	0.0	-55	165	1,627	4.2	3.0	0.0	16.8	125.91
		ALL-LOH	1,103	10,671	108,673	0	0	22.3	0.0	-55	53	1,497	4.2	3.0	0.0	15.4	111.16
		ALL-LPS	1,103	10,671	108,673	0	7	22.3	0.0	-55	60	1,506	4.2	2.7	0.0	15.5	113.21
A3	18	ALL	1,103	10,671	108,673	0	17	11.1	0.0	-78	80	937	3.1	3.3	0.0	8.6	60.24
		ALL-SPP	1,103	10,671	108,673	0	4	11.3	0.0	-78	86	860	3.0	3.0	0.0	8.0	55.03
		ALL-CTP	1,103	10,671	108,673	0	17	11.1	0.0	-78	80	937	3.1	3.3	0.0	8.6	61.15
		ALL-CLQ	1,103	10,671	108,673	0	17	11.1	0.0	-78	244	1,017	3.1	3.4	0.0	9.4	72.34
		ALL-LOH	1,103	10,671	108,673	0	8	11.1	0.0	-78	80	890	3.1	3.5	0.0	8.1	58.48
		ALL-LPS	1,103	10,671	108,673	0	17	11.1	0.0	-78	80	937	3.1	2.7	0.0	8.6	59.77
A3	19	ALL	1,052	10,187	61,817	0	15	12.2	0.0	-50	109	123	2.3	2.3	0.0	1.1	12.71
		ALL-SPP	1,052	10,187	61,817	0	15	12.2	0.0	-50	109	123	2.3	2.3	0.0	1.1	12.65
		ALL-CTP	1,052	10,187	61,817	0	15	12.2	0.0	-50	109	123	2.3	2.2	0.0	1.1	12.61
		ALL-CLQ	1,052	10,187	61,817	0	13	12.2	0.0	-50	95	108	2.2	2.0	0.0	1.0	11.39
		ALL-LOH	1,052	10,187	61,817	0	2	12.2	0.0	-50	92	115	2.3	2.2	0.0	1.0	13.35
		ALL-LPS	1,052	10,187	61,817	0	11	12.2	0.0	-50	86	108	2.1	1.8	0.0	0.9	11.74
A3	20	ALL	1,054	10,241	62,171	0	22	12.0	0.0	-74	119	317	1.3	2.2	0.0	2.1	18.40
		ALL-SPP	1,054	10,241	62,171	0	2	13.4	0.0	-74	52	260	1.3	2.0	0.0	1.8	15.68
		ALL-CTP	1,054	10,241	62,171	0	22	12.0	0.0	-74	119	317	1.3	2.1	0.0	2.1	17.75
		ALL-CLQ	1,054	10,241	62,171	0	26	12.0	0.0	-74	120	340	1.3	2.2	0.0	2.3	19.16
		ALL-LOH	1,054	10,241	62,171	0	13	12.0	0.0	-74	210	387	1.3	2.2	0.0	2.6	20.33
		ALL-LPS	1,054	10,241	62,171	0	22	12.0	0.0	-74	119	317	1.3	1.9	0.0	2.1	17.44
A3	21	ALL	1,054	10,241	62,171	0	43	8.5	0.0	-97	605	632	1.0	3.1	0.0	4.2	30.24
		ALL-SPP	1,054	10,241	62,171	0	6	8.9	0.0	-97	387	507	0.9	2.1	0.0	3.2	22.34
		ALL-CTP	1,054	10,241	62,171	0	43	8.5	0.0	-97	605	632	1.0	2.8	0.0	4.2	29.52
		ALL-CLQ	1,054	10,241	62,171	0	53	8.5	0.0	-97	618	666	1.0	3.0	0.0	4.3	33.67
		ALL-LOH	1,054	10,241	62,171	0	14	8.5	0.0	-97	418	557	1.1	3.0	0.0	3.8	25.15
		ALL-LPS	1,054	10,241	62,171	0	43	8.5	0.0	-97	605	632	1.0	2.1	0.0	4.2	28.84

◇ Problem is infeasible

Table 26 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	22	ALL	1,037	9,934	101,100	30	13	27.7	0.0	-36	10	119	6.0	3.4	0.0	1.4	28.14
		ALL-SPP	1,037	9,934	101,100	30	13	27.7	0.0	-36	10	119	6.0	3.3	0.0	1.4	27.74
		ALL-CTP	1,037	9,934	101,100	17	7	28.2	0.0	-36	48	163	5.3	2.9	0.0	1.8	27.63
		ALL-CLQ	1,037	9,934	101,100	72	4	27.8	0.0	-36	12	102	8.1	3.0	0.0	1.2	29.24
		ALL-LOH	1,037	9,934	101,100	71	1	27.8	0.0	-36	12	112	8.1	3.1	0.0	1.3	29.57
		ALL-LPS	1,037	9,934	101,100	30	13	27.7	0.0	-36	10	119	6.0	2.6	0.0	1.4	27.17
A3	23	ALL	1,059	10,241	104,275	0	4	16.6	0.0	-59	88	472	3.3	2.9	0.0	4.9	50.38
		ALL-SPP	1,059	10,241	104,275	0	4	16.6	0.0	-59	88	472	3.2	2.9	0.0	4.9	50.19
		ALL-CTP	1,059	10,241	104,275	0	4	16.6	0.0	-59	88	472	3.3	2.9	0.0	4.9	51.19
		ALL-CLQ	1,059	10,241	104,275	0	7	16.6	0.0	-59	83	453	3.3	2.9	0.0	4.7	51.70
		ALL-LOH	1,059	10,241	104,275	0	0	16.6	0.0	-59	103	482	3.3	2.9	0.0	5.0	49.88
		ALL-LPS	1,059	10,241	104,275	0	4	16.6	0.0	-59	88	472	3.3	2.6	0.0	4.9	51.04
A3	24	ALL	1,059	10,241	104,275	0	4	11.7	0.0	-79	64	923	2.4	2.9	0.0	8.6	61.46
		ALL-SPP	1,059	10,241	104,275	0	4	11.7	0.0	-79	64	923	2.4	2.9	0.0	8.5	60.84
		ALL-CTP	1,059	10,241	104,275	0	4	11.7	0.0	-79	64	923	2.4	2.9	0.0	8.6	61.50
		ALL-CLQ	1,059	10,241	104,275	0	11	11.7	0.0	-79	437	1,336	2.4	2.9	0.0	12.4	95.69
		ALL-LOH	1,059	10,241	104,275	0	0	11.7	0.0	-79	60	895	2.4	2.9	0.0	8.3	59.75
		ALL-LPS	1,059	10,241	104,275	0	4	11.7	0.0	-79	64	923	2.4	2.6	0.0	8.6	61.83
A3	25	ALL	1,187	10,917	66,231	12	37	9.5	0.0	-46	37	50	3.2	3.1	0.0	0.4	12.18
		ALL-SPP	1,187	10,917	66,231	12	37	9.5	0.0	-46	37	50	3.2	3.2	0.0	0.5	12.16
		ALL-CTP	1,187	10,917	66,231	6	8	9.6	0.0	-46	22	34	2.7	2.7	0.0	0.3	9.91
		ALL-CLQ	1,187	10,917	66,231	8	20	9.5	0.0	-46	20	32	2.9	3.0	0.0	0.3	10.52
		ALL-LOH	1,187	10,917	66,231	3	7	9.5	0.0	-46	25	39	2.6	3.2	0.0	0.4	11.49
		ALL-LPS	1,187	10,917	66,231	6	34	9.5	0.0	-46	32	45	2.7	2.5	0.0	0.4	10.06
A3	26	ALL	1,250	11,947	72,646	0	25	9.7	0.0	-65	180	264	2.3	3.4	0.0	2.2	18.60
		ALL-SPP	1,250	11,947	72,646	0	3	9.7	0.0	-65	218	277	2.2	2.8	0.0	2.4	20.78
		ALL-CTP	1,250	11,947	72,646	0	25	9.7	0.0	-65	180	264	2.3	3.3	0.0	2.2	18.38
		ALL-CLQ	1,250	11,947	72,646	0	19	9.7	0.0	-65	29	164	2.1	3.1	0.0	1.4	14.34
		ALL-LOH	1,250	11,947	72,646	0	4	9.7	0.0	-65	8	121	2.3	3.4	0.0	1.1	12.50
		ALL-LPS	1,250	11,947	72,646	0	12	9.7	0.0	-65	97	226	2.2	2.6	0.0	1.9	18.07
A3	27	ALL	1,252	11,991	72,938	0	31	5.8	0.0	-86	110	189	1.8	3.7	0.0	1.5	20.77
		ALL-SPP	1,252	11,991	72,938	0	6	6.1	0.0	-86	250	304	1.5	2.9	0.0	2.6	24.03
		ALL-CTP	1,252	11,991	72,938	0	31	5.8	0.0	-86	110	189	1.8	3.5	0.0	1.5	20.29
		ALL-CLQ	1,252	11,991	72,938	0	36	5.8	0.0	-86	20	77	1.8	3.7	0.0	0.6	12.64
		ALL-LOH	1,252	11,991	72,938	0	4	5.9	0.0	-86	90	165	1.7	3.4	0.0	1.4	18.69
		ALL-LPS	1,252	11,991	72,938	0	19	5.9	0.0	-86	20	81	1.7	2.7	0.0	0.7	11.21
A3	28	ALL	1,105	10,020	102,017	840	41	21.1	0.0	-35	62	86	11.7	5.6	0.0	1.3	50.02
		ALL-SPP	1,105	10,020	102,017	840	41	21.1	0.0	-35	62	86	11.7	5.6	0.0	1.2	48.90
		ALL-CTP	1,105	10,020	102,017	840	41	21.1	0.0	-35	62	86	11.8	5.8	0.0	1.2	50.31
		ALL-CLQ	1,105	10,020	102,017	1,207	34	20.7	0.0	-35	7	29	12.8	5.0	0.0	0.4	44.25
		ALL-LOH	1,105	10,020	102,017	1,758	5	22.1	0.0	-35	34	49	12.8	3.8	0.0	0.6	43.30
		ALL-LPS	1,105	10,020	102,017	711	58	21.0	0.0	-35	65	92	11.4	3.4	0.0	1.1	44.56

◇ Problem is infeasible

Table 26 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	29	ALL	1,254	11,931	121,785	0	1	18.0	0.0	-55	479	954	5.9	4.0	0.0	11.7	108.90
		ALL-SPP	1,254	11,931	121,785	0	1	18.0	0.0	-55	479	954	5.8	4.0	0.0	11.7	110.17
		ALL-CTP	1,254	11,931	121,785	0	1	18.0	0.0	-55	479	954	5.9	3.9	0.0	11.7	112.45
		ALL-CLQ	1,254	11,931	121,785	0	6	18.0	0.0	-55	433	903	5.9	4.2	0.0	11.1	99.63
		ALL-LOH	1,254	11,931	121,785	0	0	18.0	0.0	-55	300	860	5.8	4.1	0.0	10.6	98.83
		ALL-LPS	1,254	11,931	121,785	0	1	18.0	0.0	-55	479	954	5.8	3.6	0.0	11.7	109.50
A3	30	ALL	1,257	11,991	122,393	0	25	12.7	0.0	-74	490	1,281	4.7	4.9	0.0	14.2	123.54
		ALL-SPP	1,257	11,991	122,393	0	3	12.9	0.0	-74	122	976	3.8	4.1	0.0	10.9	90.49
		ALL-CTP	1,257	11,991	122,393	0	25	12.7	0.0	-74	490	1,281	4.8	4.9	0.0	14.3	124.22
		ALL-CLQ	1,257	11,991	122,393	0	23	12.7	0.0	-74	1,908	2,291	4.8	5.0	0.0	25.5	178.71
		ALL-LOH	1,257	11,991	122,393	0	8	12.7	0.0	-74	648	1,428	4.8	5.0	0.0	16.0	137.10
		ALL-LPS	1,257	11,991	122,393	0	25	12.7	0.0	-74	490	1,281	4.8	3.8	0.0	14.3	124.76
A3	31	ALL	1,276	12,182	74,022	5	24	14.0	0.0	-45	58	91	3.6	3.5	0.0	0.9	17.28
		ALL-SPP	1,276	12,182	74,022	4	56	18.0	0.0	-45	25	74	3.5	4.8	0.0	0.7	18.20
		ALL-CTP	1,276	12,182	74,022	5	24	14.0	0.0	-45	58	91	3.6	3.4	0.0	0.9	16.93
		ALL-CLQ	1,276	12,182	74,022	9	32	14.0	0.0	-45	3	44	3.6	3.8	0.0	0.4	13.66
		ALL-LOH	1,276	12,182	74,022	14	23	14.0	0.0	-45	3	38	3.9	4.2	0.0	0.3	14.11
		ALL-LPS	1,276	12,182	74,022	18	66	13.9	0.0	-45	4	46	4.4	3.4	0.0	0.4	14.41
A3	32	ALL	1,288	12,501	76,002	0	13	6.5	0.0	-68	6	40	2.1	3.0	0.0	0.4	10.72
		ALL-SPP	1,288	12,501	76,002	0	3	6.8	0.0	-68	65	93	2.2	2.8	0.0	0.8	13.79
		ALL-CTP	1,288	12,501	76,002	0	13	6.5	0.0	-68	6	40	2.1	2.9	0.0	0.4	10.57
		ALL-CLQ	1,288	12,501	76,002	0	10	6.5	0.0	-68	19	55	2.1	3.1	0.0	0.5	12.44
		ALL-LOH	1,288	12,501	76,002	0	1	6.5	0.0	-68	10	47	2.1	3.0	0.0	0.4	10.64
		ALL-LPS	1,288	12,501	76,002	0	13	6.5	0.0	-68	6	40	2.1	2.7	0.0	0.4	10.34
A3	33	ALL	1,288	12,502	76,009	0	32	5.4	0.0	-86	200	317	1.8	3.5	0.0	2.7	28.05
		ALL-SPP	1,288	12,502	76,009	0	32	5.4	0.0	-86	200	317	1.8	3.5	0.0	2.7	28.39
		ALL-CTP	1,288	12,502	76,009	0	32	5.4	0.0	-86	200	317	1.8	3.3	0.0	2.7	27.66
		ALL-CLQ	1,288	12,502	76,009	0	13	5.5	0.0	-86	110	197	1.8	3.2	0.0	1.6	17.27
		ALL-LOH	1,288	12,502	76,009	0	1	5.5	0.0	-86	120	229	1.8	3.2	0.0	1.9	23.53
		ALL-LPS	1,288	12,502	76,009	0	6	5.8	0.0	-86	628	651	1.5	2.7	0.0	5.9	43.60
A3	34	ALL	1,255	11,872	121,108	210	8	23.9	0.0	-36	133	162	11.6	4.3	0.0	2.2	56.80
		ALL-SPP	1,255	11,872	121,108	210	8	23.9	0.0	-36	133	162	11.5	4.3	0.0	2.2	56.08
		ALL-CTP	1,255	11,872	121,108	210	8	23.9	0.0	-36	133	162	11.5	4.4	0.0	2.2	56.72
		ALL-CLQ	1,255	11,872	121,108	251	6	23.7	0.0	-36	33	74	11.9	4.3	0.0	1.1	43.02
		ALL-LOH	1,255	11,872	121,108	144	8	23.8	0.0	-36	41	105	11.4	4.7	0.0	1.5	46.91
		ALL-LPS	1,255	11,872	121,108	210	8	23.9	0.0	-36	133	162	11.6	3.7	0.0	2.2	55.99
A3	35	ALL	1,293	12,498	127,513	0	4	18.2	0.0	-57	50	1,087	4.6	4.2	0.0	13.4	108.64
		ALL-SPP	1,293	12,498	127,513	0	4	18.2	0.0	-57	50	1,087	4.6	4.2	0.0	13.4	108.99
		ALL-CTP	1,293	12,498	127,513	0	4	18.2	0.0	-57	50	1,087	4.6	4.3	0.0	13.4	110.33
		ALL-CLQ	1,293	12,498	127,513	0	7	18.2	0.0	-57	150	1,115	4.6	4.2	0.0	13.9	118.26
		ALL-LOH	1,293	12,498	127,513	0	0	18.2	0.0	-57	110	1,116	4.6	4.3	0.0	14.0	114.08
		ALL-LPS	1,293	12,498	127,513	0	4	18.2	0.0	-57	50	1,087	4.6	3.9	0.0	13.4	109.45

◇ Problem is infeasible

Table 26 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	36	ALL	1,293	12,502	127,555	0	3	10.9	0.0	-77	49	1,283	3.6	4.3	0.0	14.4	98.31
		ALL-SPP	1,293	12,502	127,555	0	3	10.9	0.0	-77	49	1,283	3.6	4.3	0.0	14.5	100.64
		ALL-CTP	1,293	12,502	127,555	0	3	10.9	0.0	-77	49	1,283	3.7	4.3	0.0	14.5	100.12
		ALL-CLQ	1,293	12,502	127,555	0	5	10.9	0.0	-77	60	1,415	3.5	4.2	0.0	15.5	105.18
		ALL-LOH	1,293	12,502	127,555	0	0	10.9	0.0	-77	60	1,371	3.6	4.4	0.0	15.2	110.25
		ALL-LPS	1,293	12,502	127,555	0	3	10.9	0.0	-77	49	1,283	3.6	3.9	0.0	14.5	100.35
C2	1	ALL	101	898	2,571	0	456	1.2	0.0	-2,119	176	187	0.0	0.4	0.0	0.1	1.29
		ALL-NDP	101	898	2,571	0	456	1.2	0.0	-2,119	176	187	0.0	0.4	0.0	0.1	1.28
		ALL-SPP	101	898	2,571	0	226	1.4	0.0	-2,119	29	39	0.0	0.1	0.0	0.0	0.25
		ALL-CTP	101	898	2,571	0	1,325	1.2	0.0	-2,119	130	151	0.0	0.4	0.0	0.1	3.15
		ALL-CLQ	101	898	2,571	0	281	1.2	0.0	-2,119	54	75	0.0	0.3	0.0	0.0	0.63
		ALL-LOH	101	898	2,571	0	418	1.2	0.0	-2,119	103	113	0.0	0.5	0.0	0.0	1.03
		ALL-GSE	101	898	2,571	0	305	1.2	0.0	-2,119	32	59	0.0	0.3	0.0	0.0	0.63
C2	2	ALL	101	879	2,515	0	274	2.5	0.0	-1,868	91	228	0.0	0.2	0.0	0.1	0.75
		ALL-NDP	101	879	2,515	0	274	2.5	0.0	-1,868	91	228	0.0	0.2	0.0	0.1	0.74
		ALL-SPP	101	879	2,515	0	286	2.7	0.0	-1,868	185	296	0.0	0.1	0.0	0.1	0.72
		ALL-CTP	101	879	2,515	0	973	2.5	0.0	-1,868	382	406	0.0	0.3	0.0	0.2	3.75
		ALL-CLQ	101	879	2,515	0	227	2.7	0.0	-1,868	155	243	0.0	0.1	0.0	0.1	0.60
		ALL-LOH	101	879	2,515	0	293	2.5	0.0	-1,868	289	291	0.0	0.3	0.0	0.1	0.99
		ALL-GSE	101	879	2,515	0	223	2.5	0.0	-1,868	146	164	0.0	0.2	0.0	0.1	0.49
C2	5	ALL	110	907	9,860	0	171	8.2	0.0	-1,515	743	982	0.0	0.1	0.0	0.8	2.24
		ALL-NDP	110	907	9,860	0	207	8.2	0.0	-1,515	605	1,147	0.0	0.1	0.0	1.3	3.71
		ALL-SPP	110	907	9,860	0	126	8.2	0.0	-1,515	679	910	0.0	0.1	0.0	0.8	2.22
		ALL-CTP	110	907	9,860	0	279	8.2	0.0	-1,515	1,166	1,411	0.0	0.2	0.0	1.3	5.96
		ALL-CLQ	110	907	9,860	0	158	8.4	0.0	-1,515	630	1,027	0.0	0.1	0.0	0.9	2.48
		ALL-LOH	110	907	9,860	0	168	8.2	0.0	-1,515	380	811	0.0	0.1	0.0	0.6	1.86
		ALL-GSE	110	907	9,860	0	135	8.2	0.0	-1,515	946	1,035	0.0	0.1	0.0	0.9	2.47
C2	6	ALL	110	868	9,367	0	227	12.2	0.0	-1,279	220	1,064	0.0	0.2	0.0	1.0	3.30
		ALL-NDP	110	868	9,367	0	230	12.2	0.0	-1,279	1,313	1,550	0.0	0.2	0.0	1.2	4.10
		ALL-SPP	110	868	9,367	0	156	12.6	0.0	-1,279	620	1,486	0.0	0.2	0.0	1.5	3.99
		ALL-CTP	110	868	9,367	0	315	12.2	0.0	-1,279	720	1,531	0.0	0.2	0.0	1.3	6.33
		ALL-CLQ	110	868	9,367	0	219	12.4	0.0	-1,279	190	1,100	0.0	0.2	0.0	0.9	2.95
		ALL-LOH	110	868	9,367	0	230	12.2	0.0	-1,279	1,280	1,508	0.0	0.2	0.0	1.3	4.21
		ALL-GSE	110	868	9,367	0	186	12.2	0.0	-1,279	1,350	1,589	0.0	0.2	0.0	1.3	4.11
C2	9	ALL	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		ALL-NDP	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		ALL-SPP	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		ALL-CTP	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		ALL-CLQ	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		ALL-LOH	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		ALL-GSE	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00

◇ Problem is infeasible

Table 26 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C2	10	ALL	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		ALL-NDP	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		ALL-SPP	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		ALL-CTP	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		ALL-CLQ	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		ALL-LOH	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
		ALL-GSE	6	5	15	0	0	0.0	0.0	-808	0	0	0.0	0.0	0.0	0.0	0.00
C2	13	ALL	20	11	109	0	0	0.0	0.0	-593	0	0	0.0	0.0	0.0	0.0	0.00
		ALL-NDP	20	11	109	0	0	0.0	0.0	-593	0	0	0.0	0.0	0.0	0.0	0.00
		ALL-SPP	20	11	109	0	0	0.0	0.0	-593	0	0	0.0	0.0	0.0	0.0	0.00
		ALL-CTP	20	11	109	0	0	0.0	0.0	-593	0	0	0.0	0.0	0.0	0.0	0.00
		ALL-CLQ	20	11	109	0	0	0.0	0.0	-593	0	0	0.0	0.0	0.0	0.0	0.00
		ALL-LOH	20	11	109	0	0	0.0	0.0	-593	0	0	0.0	0.0	0.0	0.0	0.00
		ALL-GSE	20	11	109	0	0	0.0	0.0	-593	0	0	0.0	0.0	0.0	0.0	0.00
C2	14	ALL	67	94	862	0	0	◇	◇	◇	◇	0	0.0	0.0	0.0	0.0	0.00
		ALL-NDP	67	94	862	0	0	◇	◇	◇	◇	0	0.0	0.0	0.0	0.0	0.00
		ALL-SPP	67	94	862	0	0	◇	◇	◇	◇	0	0.0	0.0	0.0	0.0	0.00
		ALL-CTP	67	94	862	0	0	◇	◇	◇	◇	0	0.0	0.0	0.0	0.0	0.00
		ALL-CLQ	67	94	862	0	0	◇	◇	◇	◇	0	0.0	0.0	0.0	0.0	0.00
		ALL-LOH	67	94	862	0	0	◇	◇	◇	◇	0	0.0	0.0	0.0	0.0	0.00
		ALL-GSE	67	94	862	0	0	◇	◇	◇	◇	0	0.0	0.0	0.0	0.0	0.00
C2	17	ALL	501	4,812	13,893	0	769	0.5	0.0	-40,731	1,254	1,577	0.0	3.7	0.0	3.5	19.82
		ALL-NDP	501	4,812	13,893	0	718	0.5	0.0	-40,731	558	891	0.0	3.8	0.0	2.0	14.06
		ALL-SPP	501	4,812	13,893	0	769	0.5	0.0	-40,731	1,254	1,577	0.0	3.6	0.0	3.5	19.60
		ALL-CTP	501	4,812	13,893	0	2,437	0.6	0.0	-40,731	2,839	3,005	0.0	6.6	0.0	7.5	139.57
		ALL-CLQ	501	4,812	13,893	0	685	0.7	0.0	-40,731	1,449	1,771	0.0	12.9	0.0	4.2	35.23
		ALL-LOH	501	4,812	13,893	0	650	0.5	0.0	-40,731	1,749	1,994	0.0	4.9	0.0	4.7	26.57
		ALL-GSE	501	4,812	13,893	0	749	0.5	0.0	-40,731	743	1,063	0.0	3.5	0.0	2.5	16.45
C2	18	ALL	501	4,809	13,884	0	497	0.7	0.0	-36,549	1,060	1,263	0.0	3.2	0.0	2.7	17.08
		ALL-NDP	501	4,809	13,884	0	500	0.7	0.0	-36,549	575	948	0.0	3.1	0.0	2.3	15.46
		ALL-SPP	501	4,809	13,884	0	497	0.7	0.0	-36,549	1,060	1,263	0.0	3.2	0.0	2.8	17.45
		ALL-CTP	501	4,809	13,884	0	1,170	0.7	0.0	-36,549	1,133	1,380	0.0	3.6	0.0	2.5	27.39
		ALL-CLQ	501	4,809	13,884	0	516	0.7	0.0	-36,549	879	943	0.0	5.0	0.0	1.7	13.99
		ALL-LOH	501	4,809	13,884	0	368	0.7	0.0	-36,549	357	824	0.0	3.1	0.0	1.9	10.37
		ALL-GSE	501	4,809	13,884	0	418	0.7	0.0	-36,549	323	730	0.0	2.6	0.0	1.7	11.15
C2	21	ALL	197	358	3,298	0	230	0.0	0.0	-4,221	0	0	0.0	0.6	0.0	0.0	0.83
		ALL-NDP	197	358	3,298	0	216	0.0	0.0	-4,221	0	0	0.0	0.6	0.0	0.0	0.82
		ALL-SPP	197	358	3,298	0	229	0.0	0.0	-4,221	0	0	0.0	0.6	0.0	0.0	0.84
		ALL-CTP	197	358	3,298	0	213	0.0	0.0	-4,221	0	0	0.0	0.6	0.0	0.0	0.82
		ALL-CLQ	197	358	3,298	0	192	0.0	0.0	-4,221	0	0	0.0	0.7	0.0	0.0	1.02
		ALL-LOH	197	358	3,298	0	153	0.0	0.0	-4,221	0	0	0.0	0.6	0.0	0.0	0.78
		ALL-GSE	197	358	3,298	0	186	0.0	0.0	-4,221	0	0	0.0	0.5	0.0	0.0	0.65

◇ Problem is infeasible

Table 26 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C2	22	ALL	72	94	908	0	22	0.0	0.0	-3,657	0	0	0.0	0.2	0.0	0.0	0.20
		ALL-NDP	72	94	908	0	18	0.0	0.0	-3,657	0	0	0.0	0.2	0.0	0.0	0.19
		ALL-SPP	72	94	908	0	22	0.0	0.0	-3,657	0	0	0.0	0.2	0.0	0.0	0.19
		ALL-CTP	72	94	908	0	20	0.0	0.0	-3,657	0	0	0.0	0.2	0.0	0.0	0.18
		ALL-CLQ	72	94	908	0	13	0.0	0.0	-3,657	0	0	0.0	0.2	0.0	0.0	0.19
		ALL-LOH	72	94	908	0	19	0.0	0.0	-3,657	0	0	0.0	0.2	0.0	0.0	0.19
		ALL-GSE	72	94	908	0	22	0.0	0.0	-3,657	0	0	0.0	0.2	0.0	0.0	0.18
C3	1	ALL	505	4,968	30,162	0	17	26.5	0.0	-43	200	430	0.0	1.0	0.0	1.3	13.49
		ALL-NDP	505	4,968	30,162	0	13	27.3	0.0	-43	145	463	0.0	1.0	0.0	1.3	11.09
		ALL-SPP	505	4,968	30,162	0	17	26.5	0.0	-43	259	548	0.0	1.0	0.0	1.6	15.52
		ALL-CTP	505	4,968	30,162	0	11	26.5	0.0	-43	189	382	0.0	1.1	0.0	1.2	11.23
		ALL-CLQ	505	4,968	30,162	0	17	26.5	0.0	-43	200	430	0.0	1.3	0.0	1.6	16.43
		ALL-LOH	505	4,968	30,162	0	17	26.5	0.0	-43	200	430	0.0	1.1	0.0	1.4	13.79
		ALL-GSE	505	4,968	30,162	0	14	26.6	0.0	-43	238	578	0.0	1.1	0.0	1.8	14.66
C3	2	ALL	505	4,968	30,162	0	66	9.5	0.0	-81	427	2,037	0.0	1.6	0.0	4.8	32.91
		ALL-NDP	505	4,968	30,162	0	76	9.5	0.0	-81	1,718	3,029	0.0	1.8	0.0	8.5	57.40
		ALL-SPP	505	4,968	30,162	0	65	9.5	0.0	-81	832	2,333	0.0	1.5	0.0	7.1	45.98
		ALL-CTP	505	4,968	30,162	0	58	10.1	0.0	-81	2,485	3,609	0.0	1.8	0.0	9.3	61.51
		ALL-CLQ	505	4,968	30,162	0	66	9.5	0.0	-81	700	2,221	0.0	1.9	0.0	6.8	48.38
		ALL-LOH	505	4,968	30,162	0	67	9.5	0.0	-81	471	2,142	0.0	1.6	0.0	5.2	36.72
		ALL-GSE	505	4,968	30,162	0	57	11.8	0.0	-81	2,372	4,173	0.0	1.8	0.0	11.3	79.73
C3	3	ALL	505	4,968	30,162	0	261	5.8	0.0	-114	23,701	23,701	0.0	6.3	0.0	56.3	308.03
		ALL-NDP	505	4,968	30,162	0	219	5.8	0.0	-114	12,103	15,055	0.0	4.6	0.0	40.6	242.84
		ALL-SPP	505	4,968	30,162	0	208	5.8	0.0	-114	14,969	16,622	0.0	4.7	0.0	42.5	232.56
		ALL-CTP	505	4,968	30,162	0	189	5.8	0.0	-114	6,080	9,854	0.0	3.4	0.0	25.7	157.50
		ALL-CLQ	505	4,968	30,162	0	214	6.0	0.0	-114	7,690	11,856	0.0	5.1	0.0	38.6	250.21
		ALL-LOH	505	4,968	30,162	0	207	5.8	0.0	-114	7,503	11,272	0.0	3.8	0.0	29.1	175.64
		ALL-GSE	505	4,968	30,162	0	194	6.6	0.0	-114	3,000	9,730	0.0	3.1	0.0	27.2	184.13
C3	4	ALL	510	4,968	50,496	0	0	◇	◇	◇	◇	84	0.0	1.6	0.0	0.4	15.29
		ALL-NDP	510	4,968	50,496	0	0	◇	◇	◇	◇	84	0.0	1.6	0.0	0.4	15.05
		ALL-SPP	510	4,968	50,496	0	0	◇	◇	◇	◇	84	0.0	1.6	0.0	0.4	15.18
		ALL-CTP	510	4,968	50,496	0	0	◇	◇	◇	◇	84	0.0	1.7	0.0	0.4	15.71
		ALL-CLQ	510	4,968	50,496	0	0	◇	◇	◇	◇	84	0.0	2.0	0.0	0.5	18.58
		ALL-LOH	510	4,968	50,496	0	0	◇	◇	◇	◇	84	0.0	1.7	0.0	0.4	15.59
		ALL-GSE	510	4,968	50,496	0	0	◇	◇	◇	◇	84	0.0	1.6	0.0	0.4	14.99
C3	5	ALL	510	4,968	50,496	0	20	20.5	0.0	-59	1,184	3,786	0.0	2.3	0.0	13.4	122.04
		ALL-NDP	510	4,968	50,496	0	20	20.5	0.0	-59	1,184	3,786	0.0	2.3	0.0	13.4	121.22
		ALL-SPP	510	4,968	50,496	0	20	20.5	0.0	-59	1,184	3,786	0.0	2.2	0.0	13.4	120.56
		ALL-CTP	510	4,968	50,496	0	20	20.5	0.0	-59	1,184	3,786	0.0	2.4	0.0	13.7	125.12
		ALL-CLQ	510	4,968	50,496	0	20	20.5	0.0	-59	1,184	3,786	0.0	2.8	0.0	16.4	148.00
		ALL-LOH	510	4,968	50,496	0	20	20.5	0.0	-59	1,184	3,786	0.0	2.4	0.0	13.7	125.11
		ALL-GSE	510	4,968	50,496	0	20	20.5	0.0	-59	1,184	3,786	0.0	2.3	0.0	13.4	120.55

◇ Problem is infeasible

Table 26 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C3	6	ALL	510	4,968	50,496	0	133	11.1	0.0	-88	2,164	10,013	0.0	3.8	0.0	31.0	266.24
		ALL-NDP	510	4,968	50,496	0	99	11.2	0.0	-88	1,781	10,518	0.0	3.7	0.0	33.6	265.82
		ALL-SPP	510	4,968	50,496	0	91	11.2	0.0	-88	1,525	9,716	0.0	3.4	0.0	30.0	232.64
		ALL-CTP	510	4,968	50,496	0	120	11.1	0.0	-88	1,250	9,606	0.0	3.9	0.0	29.5	247.09
		ALL-CLQ	510	4,968	50,496	0	132	11.1	0.0	-88	554	8,841	0.0	4.3	0.0	31.6	256.23
		ALL-LOH	510	4,968	50,496	0	129	11.1	0.0	-88	566	9,075	0.0	3.8	0.0	27.4	228.27
		ALL-GSE	510	4,968	50,496	0	105	11.1	0.0	-88	4,216	11,625	0.0	3.9	0.0	39.2	324.84
C3	7	ALL	505	4,963	30,094	0	10	20.0	0.0	-41	138	192	0.0	1.0	0.0	0.6	7.30
		ALL-NDP	505	4,963	30,094	0	10	20.0	0.0	-41	138	192	0.0	1.0	0.0	0.6	7.28
		ALL-SPP	505	4,963	30,094	0	3	22.8	0.0	-41	120	195	0.0	0.9	0.0	0.5	6.17
		ALL-CTP	505	4,963	30,094	0	10	20.0	0.0	-41	138	192	0.0	1.0	0.0	0.6	7.49
		ALL-CLQ	505	4,963	30,094	0	10	20.0	0.0	-41	138	192	0.0	1.2	0.0	0.7	8.89
		ALL-LOH	505	4,963	30,094	0	10	20.0	0.0	-41	138	192	0.0	1.0	0.0	0.6	7.46
		ALL-GSE	505	4,963	30,094	0	10	20.0	0.0	-41	138	192	0.0	1.0	0.0	0.6	7.33
C3	8	ALL	505	4,963	30,094	0	62	9.1	0.0	-82	400	1,106	0.0	1.3	0.0	3.6	27.95
		ALL-NDP	505	4,963	30,094	0	56	9.1	0.0	-82	300	919	0.0	1.2	0.0	2.8	20.94
		ALL-SPP	505	4,963	30,094	0	49	9.2	0.0	-82	291	979	0.0	1.2	0.0	2.8	19.17
		ALL-CTP	505	4,963	30,094	0	50	9.1	0.0	-82	500	1,205	0.0	1.3	0.0	3.7	24.20
		ALL-CLQ	505	4,963	30,094	0	62	9.1	0.0	-82	400	1,106	0.0	1.6	0.0	4.4	34.07
		ALL-LOH	505	4,963	30,094	0	62	9.1	0.0	-82	400	1,106	0.0	1.3	0.0	3.7	28.74
		ALL-GSE	505	4,963	30,094	0	43	9.2	0.0	-82	300	1,268	0.0	1.2	0.0	3.5	25.60
C3	9	ALL	505	4,963	30,094	0	110	5.4	0.0	-119	4,536	5,250	0.0	2.4	0.0	16.9	92.51
		ALL-NDP	505	4,963	30,094	0	104	5.4	0.0	-119	3,481	4,127	0.0	2.1	0.0	12.4	73.30
		ALL-SPP	505	4,963	30,094	0	110	5.4	0.0	-119	1,737	2,585	0.0	1.7	0.0	6.5	41.38
		ALL-CTP	505	4,963	30,094	0	103	5.4	0.0	-119	867	2,255	0.0	1.7	0.0	6.8	35.81
		ALL-CLQ	505	4,963	30,094	0	110	5.4	0.0	-119	4,536	5,250	0.0	2.9	0.0	20.6	112.62
		ALL-LOH	505	4,963	30,094	0	118	5.4	0.0	-119	1,608	2,947	0.0	1.9	0.0	9.0	54.14
		ALL-GSE	505	4,963	30,094	0	106	5.4	0.0	-119	2,256	3,460	0.0	1.8	0.0	11.1	60.06
C3	10	ALL	510	4,963	50,370	0	6	◇	◇	◇	◇	38	0.0	1.6	0.0	0.2	9.16
		ALL-NDP	510	4,963	50,370	0	6	◇	◇	◇	◇	38	0.0	1.6	0.0	0.2	9.08
		ALL-SPP	510	4,963	50,370	0	0	◇	◇	◇	◇	30	0.0	1.5	0.0	0.1	7.97
		ALL-CTP	510	4,963	50,370	0	6	◇	◇	◇	◇	38	0.0	1.7	0.0	0.2	9.68
		ALL-CLQ	510	4,963	50,370	0	5	◇	◇	◇	◇	38	0.0	2.0	0.0	0.2	11.11
		ALL-LOH	510	4,963	50,370	0	5	◇	◇	◇	◇	38	0.0	1.7	0.0	0.2	9.30
		ALL-GSE	510	4,963	50,370	0	6	◇	◇	◇	◇	38	0.0	1.6	0.0	0.2	9.08
C3	11	ALL	510	4,963	50,370	0	39	17.9	0.0	-56	2,546	4,494	0.0	2.5	0.0	17.8	169.23
		ALL-NDP	510	4,963	50,370	0	39	17.9	0.0	-56	2,546	4,494	0.0	2.5	0.0	17.8	169.42
		ALL-SPP	510	4,963	50,370	0	32	18.0	0.0	-56	1,444	3,693	0.0	2.3	0.0	13.7	125.15
		ALL-CTP	510	4,963	50,370	0	39	17.9	0.0	-56	2,546	4,494	0.0	2.6	0.0	18.2	174.40
		ALL-CLQ	510	4,963	50,370	0	39	17.9	0.0	-56	2,546	4,494	0.0	3.1	0.0	21.7	206.89
		ALL-LOH	510	4,963	50,370	0	39	17.9	0.0	-56	2,546	4,494	0.0	2.6	0.0	18.1	174.29
		ALL-GSE	510	4,963	50,370	0	39	17.9	0.0	-56	2,546	4,494	0.0	2.5	0.0	17.8	168.86

◇ Problem is infeasible

Table 26 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C3	12	ALL	510	4,963	50,370	0	104	11.8	0.0	-83	7,819	25,192	0.0	6.6	0.0	94.8	683.57
		ALL-NDP	510	4,963	50,370	0	104	11.8	0.0	-83	7,819	25,192	0.0	6.7	0.0	94.9	682.13
		ALL-SPP	510	4,963	50,370	0	80	11.9	0.0	-83	839	16,703	0.0	4.5	0.0	50.2	348.82
		ALL-CTP	510	4,963	50,370	0	104	11.8	0.0	-83	7,819	25,192	0.0	7.0	0.0	97.3	707.79
		ALL-CLQ	510	4,963	50,370	0	104	11.8	0.0	-83	7,819	25,192	0.0	8.1	0.0	116.1	834.68
		ALL-LOH	510	4,963	50,370	0	112	11.8	0.0	-83	12,200	28,777	0.0	7.9	0.0	117.0	878.41
		ALL-GSE	510	4,963	50,370	0	122	11.9	0.0	-83	12,414	27,715	0.0	7.2	0.0	107.9	826.65
C3	13	ALL	755	7,524	45,856	0	76	11.7	0.0	-35	0	3	0.0	2.3	0.0	0.0	3.27
		ALL-NDP	755	7,524	45,856	0	56	12.2	0.0	-35	0	5	0.0	2.3	0.0	0.0	3.30
		ALL-SPP	755	7,524	45,856	0	14	14.3	0.0	-35	0	5	0.0	2.0	0.0	0.0	2.93
		ALL-CTP	755	7,524	45,856	0	76	11.7	0.0	-35	0	3	0.0	2.3	0.0	0.0	3.31
		ALL-CLQ	755	7,524	45,856	0	76	11.7	0.0	-35	0	3	0.0	2.8	0.0	0.0	3.99
		ALL-LOH	755	7,524	45,856	0	74	11.7	0.0	-35	0	3	0.0	2.4	0.0	0.0	3.38
		ALL-GSE	755	7,524	45,856	0	76	11.7	0.0	-35	0	3	0.0	2.2	0.0	0.0	3.21
C3	14	ALL	755	7,524	45,856	0	71	8.9	0.0	-73	300	778	0.0	2.6	0.0	3.7	48.85
		ALL-NDP	755	7,524	45,856	0	71	8.9	0.0	-73	300	778	0.0	2.7	0.0	3.6	48.63
		ALL-SPP	755	7,524	45,856	0	6	10.1	0.0	-73	100	492	0.0	2.1	0.0	2.1	26.89
		ALL-CTP	755	7,524	45,856	0	63	8.9	0.0	-73	200	597	0.0	2.6	0.0	2.8	35.42
		ALL-CLQ	755	7,524	45,856	0	71	8.9	0.0	-73	300	778	0.0	3.2	0.0	4.5	60.41
		ALL-LOH	755	7,524	45,856	0	71	8.9	0.0	-73	300	778	0.0	2.7	0.0	3.7	50.48
		ALL-GSE	755	7,524	45,856	0	65	8.9	0.0	-73	300	698	0.0	2.5	0.0	3.5	36.25
C3	15	ALL	755	7,524	45,856	0	146	7.9	0.0	-105	3,794	6,802	0.0	4.8	0.0	28.9	264.48
		ALL-NDP	755	7,524	45,856	0	146	7.9	0.0	-105	3,794	6,802	0.0	4.8	0.0	28.9	264.07
		ALL-SPP	755	7,524	45,856	0	68	8.2	0.0	-105	72	3,080	0.0	3.1	0.0	10.0	67.87
		ALL-CTP	755	7,524	45,856	0	131	7.9	0.0	-105	3,948	6,333	0.0	4.7	0.0	30.7	250.64
		ALL-CLQ	755	7,524	45,856	0	146	7.9	0.0	-105	3,794	6,802	0.0	5.8	0.0	35.2	324.03
		ALL-LOH	755	7,524	45,856	0	146	7.9	0.0	-105	3,794	6,802	0.0	5.0	0.0	29.6	275.92
		ALL-GSE	755	7,524	45,856	0	81	7.9	0.0	-105	3,545	5,544	0.0	3.8	0.0	21.1	159.35
C3	16	ALL	760	7,524	76,691	0	0	◇	◇	◇	◇	2	0.0	3.7	0.0	0.0	7.49
		ALL-NDP	760	7,524	76,691	0	0	◇	◇	◇	◇	2	0.0	3.7	0.0	0.0	7.52
		ALL-SPP	760	7,524	76,691	0	0	◇	◇	◇	◇	2	0.0	3.5	0.0	0.0	7.30
		ALL-CTP	760	7,524	76,691	0	0	◇	◇	◇	◇	2	0.0	3.7	0.0	0.0	8.02
		ALL-CLQ	760	7,524	76,691	0	0	◇	◇	◇	◇	2	0.0	4.5	0.0	0.0	9.17
		ALL-LOH	760	7,524	76,691	0	0	◇	◇	◇	◇	2	0.0	3.8	0.0	0.0	8.05
		ALL-GSE	760	7,524	76,691	0	0	◇	◇	◇	◇	2	0.0	3.7	0.0	0.0	7.45
C3	17	ALL	760	7,524	76,691	0	43	20.7	0.0	-57	876	1,794	0.0	4.3	0.0	9.3	157.30
		ALL-NDP	760	7,524	76,691	0	36	20.8	0.0	-57	198	1,487	0.0	4.2	0.0	7.4	130.02
		ALL-SPP	760	7,524	76,691	0	14	21.0	0.0	-57	1,678	2,534	0.0	4.4	0.0	14.8	175.26
		ALL-CTP	760	7,524	76,691	0	43	20.7	0.0	-57	876	1,794	0.0	4.4	0.0	9.5	166.59
		ALL-CLQ	760	7,524	76,691	0	43	20.7	0.0	-57	876	1,794	0.0	5.2	0.0	11.3	192.65
		ALL-LOH	760	7,524	76,691	0	43	20.7	0.0	-57	876	1,794	0.0	4.4	0.0	9.5	166.67
		ALL-GSE	760	7,524	76,691	0	43	20.7	0.0	-57	876	1,794	0.0	4.2	0.0	9.3	156.56

◇ Problem is infeasible

Table 26 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C3	18	ALL	760	7,524	76,691	0	201	13.3	0.0	-85	1,200	14,236	0.0	8.1	0.0	66.6	833.19
		ALL-NDP	760	7,524	76,691	0	192	13.3	0.0	-85	1,900	15,338	0.0	8.4	0.0	75.1	947.88
		ALL-SPP	760	7,524	76,691	0	100	13.8	0.0	-85	2,900	15,451	0.0	8.0	0.0	78.1	823.69
		ALL-CTP	760	7,524	76,691	0	201	13.3	0.0	-85	1,200	14,236	0.0	8.2	0.0	68.3	879.63
		ALL-CLQ	760	7,524	76,691	0	199	13.3	0.0	-85	1,100	13,913	0.0	9.8	0.0	80.4	1,004.24
		ALL-LOH	760	7,524	76,691	0	200	13.3	0.0	-85	1,100	14,335	0.0	8.6	0.0	68.7	888.58
		ALL-GSE	760	7,524	76,691	0	149	13.7	0.0	-85	5,300	18,081	0.0	9.4	0.0	103.9	1,152.61
C3	19	ALL	755	7,595	46,279	0	15	32.7	0.0	-38	340	380	0.0	2.3	0.0	1.5	20.71
		ALL-NDP	755	7,595	46,279	0	15	32.7	0.0	-38	340	380	0.0	2.3	0.0	1.5	20.63
		ALL-SPP	755	7,595	46,279	0	3	34.4	0.0	-38	506	560	0.0	2.2	0.0	2.1	21.86
		ALL-CTP	755	7,595	46,279	0	15	32.7	0.0	-38	340	380	0.0	2.4	0.0	1.5	21.20
		ALL-CLQ	755	7,595	46,279	0	15	32.7	0.0	-38	340	380	0.0	2.8	0.0	1.8	25.09
		ALL-LOH	755	7,595	46,279	0	15	32.7	0.0	-38	340	380	0.0	2.4	0.0	1.5	21.32
		ALL-GSE	755	7,595	46,279	0	15	32.7	0.0	-38	340	380	0.0	2.3	0.0	1.5	20.62
C3	20	ALL	755	7,595	46,279	0	37	11.9	0.0	-80	100	841	0.0	2.6	0.0	3.1	34.12
		ALL-NDP	755	7,595	46,279	0	37	11.9	0.0	-80	100	841	0.0	2.6	0.0	3.0	33.73
		ALL-SPP	755	7,595	46,279	0	20	12.7	0.0	-80	400	1,208	0.0	2.4	0.0	5.5	53.44
		ALL-CTP	755	7,595	46,279	0	35	11.9	0.0	-80	200	962	0.0	2.6	0.0	4.1	39.81
		ALL-CLQ	755	7,595	46,279	0	37	11.9	0.0	-80	100	841	0.0	3.2	0.0	3.7	41.65
		ALL-LOH	755	7,595	46,279	0	37	11.9	0.0	-80	100	841	0.0	2.7	0.0	3.1	35.50
		ALL-GSE	755	7,595	46,279	0	28	12.0	0.0	-80	100	895	0.0	2.5	0.0	3.2	29.97
C3	21	ALL	755	7,595	46,279	0	149	7.6	0.0	-117	3,003	4,019	0.0	3.8	0.0	20.1	179.25
		ALL-NDP	755	7,595	46,279	0	149	7.6	0.0	-117	3,003	4,019	0.0	3.8	0.0	20.1	178.51
		ALL-SPP	755	7,595	46,279	0	72	7.7	0.0	-117	2,433	3,746	0.0	3.4	0.0	20.1	179.23
		ALL-CTP	755	7,595	46,279	0	111	7.6	0.0	-117	2,481	4,051	0.0	3.9	0.0	23.3	178.60
		ALL-CLQ	755	7,595	46,279	0	153	7.6	0.0	-117	3,396	4,359	0.0	4.9	0.0	27.9	220.57
		ALL-LOH	755	7,595	46,279	0	153	7.6	0.0	-117	3,396	4,359	0.0	4.1	0.0	23.3	188.55
		ALL-GSE	755	7,595	46,279	0	105	7.9	0.0	-117	2,340	3,856	0.0	3.5	0.0	20.9	206.71
C3	22	ALL	760	7,595	77,359	0	7	◇	◇	◇	◇	74	0.0	3.7	0.0	0.6	35.13
		ALL-NDP	760	7,595	77,359	0	7	◇	◇	◇	◇	74	0.0	3.7	0.0	0.6	35.07
		ALL-SPP	760	7,595	77,359	0	0	◇	◇	◇	◇	68	0.0	3.5	0.0	0.5	30.74
		ALL-CTP	760	7,595	77,359	0	7	◇	◇	◇	◇	74	0.0	3.8	0.0	0.6	38.62
		ALL-CLQ	760	7,595	77,359	0	7	◇	◇	◇	◇	74	0.0	4.6	0.0	0.7	43.60
		ALL-LOH	760	7,595	77,359	0	7	◇	◇	◇	◇	74	0.0	3.8	0.0	0.6	38.91
		ALL-GSE	760	7,595	77,359	0	7	◇	◇	◇	◇	74	0.0	3.7	0.0	0.6	34.97
C3	23	ALL	760	7,595	77,359	0	48	10.4	0.0	-62	132	278	0.0	3.9	0.0	1.6	28.83
		ALL-NDP	760	7,595	77,359	0	44	10.4	0.0	-62	131	287	0.0	3.8	0.0	1.6	29.61
		ALL-SPP	760	7,595	77,359	0	1	11.2	0.0	-62	117	256	0.0	3.6	0.0	1.4	23.99
		ALL-CTP	760	7,595	77,359	0	48	10.4	0.0	-62	132	278	0.0	3.9	0.0	1.6	30.10
		ALL-CLQ	760	7,595	77,359	0	47	10.4	0.0	-62	145	287	0.0	4.7	0.0	2.0	36.64
		ALL-LOH	760	7,595	77,359	0	47	10.4	0.0	-62	138	301	0.0	4.0	0.0	1.7	31.15
		ALL-GSE	760	7,595	77,359	0	50	10.4	0.0	-62	138	313	0.0	3.8	0.0	1.8	30.91

◇ Problem is infeasible

Table 26 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C3	24	ALL	760	7,595	77,359	0	151	13.9	0.0	-85	13,315	49,336	0.0	18.0	0.0	262.1	2,633.63
		ALL-NDP	760	7,595	77,359	0	155	13.9	0.0	-85	15,873	51,464	0.0	18.3	0.0	273.6	2,883.47
		ALL-SPP	760	7,595	77,359	0	147	13.9	0.0	-85	16,588	54,259	0.0	18.8	0.0	281.1	2,889.65
		ALL-CTP	760	7,595	77,359	0	151	13.9	0.0	-85	13,315	49,336	0.0	18.9	0.0	268.3	2,768.87
		ALL-CLQ	760	7,595	77,359	0	151	13.9	0.0	-85	13,315	49,336	0.0	22.0	0.0	320.5	3,227.99
		ALL-LOH	760	7,595	77,359	0	151	13.9	0.0	-85	13,315	49,336	0.0	19.1	0.0	268.9	2,781.17
		ALL-GSE	760	7,595	77,359	0	129	14.7	0.0	-85	10,634	50,883	0.0	17.9	0.0	252.1	2,540.66
C3	25	ALL	1,003	9,960	60,766	0	2	◇	◇	◇	◇	34	0.0	3.9	0.0	0.2	19.60
		ALL-NDP	1,003	9,960	60,766	0	2	◇	◇	◇	◇	34	0.0	3.9	0.0	0.2	19.50
		ALL-SPP	1,003	9,960	60,766	0	0	◇	◇	◇	◇	32	0.0	3.6	0.0	0.2	18.26
		ALL-CTP	1,003	9,960	60,766	0	2	◇	◇	◇	◇	34	0.0	3.9	0.0	0.2	19.91
		ALL-CLQ	1,003	9,960	60,766	0	3	◇	◇	◇	◇	38	0.0	4.7	0.0	0.3	25.53
		ALL-LOH	1,003	9,960	60,766	0	2	◇	◇	◇	◇	34	0.0	3.9	0.0	0.2	21.09
		ALL-GSE	1,003	9,960	60,766	0	2	◇	◇	◇	◇	34	0.0	3.8	0.0	0.2	19.46
C3	26	ALL	1,003	9,960	60,766	0	20	16.1	0.0	-69	1,262	1,861	0.0	4.8	0.0	14.0	109.71
		ALL-NDP	1,003	9,960	60,766	0	20	16.1	0.0	-69	1,262	1,861	0.0	4.9	0.0	14.0	109.43
		ALL-SPP	1,003	9,960	60,766	0	10	16.5	0.0	-69	1,337	1,853	0.0	4.5	0.0	13.3	134.68
		ALL-CTP	1,003	9,960	60,766	0	20	16.1	0.0	-69	1,262	1,861	0.0	4.9	0.0	14.2	111.14
		ALL-CLQ	1,003	9,960	60,766	0	20	16.1	0.0	-69	1,262	1,861	0.0	5.8	0.0	17.0	130.68
		ALL-LOH	1,003	9,960	60,766	0	20	16.1	0.0	-69	1,262	1,861	0.0	5.0	0.0	14.3	115.84
		ALL-GSE	1,003	9,960	60,766	0	20	16.1	0.0	-69	1,262	1,861	0.0	4.8	0.0	14.0	109.38
C3	27	ALL	1,003	9,960	60,766	0	266	10.6	0.0	-106	7,821	8,210	0.0	8.8	0.0	53.9	703.37
		ALL-NDP	1,003	9,960	60,766	0	266	10.6	0.0	-106	7,821	8,210	0.0	8.8	0.0	54.1	703.17
		ALL-SPP	1,003	9,960	60,766	0	134	11.1	0.0	-106	10,943	12,889	0.0	9.6	0.0	86.7	915.63
		ALL-CTP	1,003	9,960	60,766	0	233	10.7	0.0	-106	12,352	13,941	0.0	11.0	0.0	99.6	1,172.15
		ALL-CLQ	1,003	9,960	60,766	0	614	10.6	0.0	-106	11,053	12,845	0.0	18.6	0.0	110.3	2,015.19
		ALL-LOH	1,003	9,960	60,766	0	231	10.6	0.0	-106	13	2,972	0.0	6.7	0.0	13.3	154.45
		ALL-GSE	1,003	9,960	60,766	0	116	10.9	0.0	-106	9,525	11,507	0.0	8.9	0.0	70.9	738.68
C3	28	ALL	1,008	9,960	101,846	0	0	◇	◇	◇	◇	4	0.0	6.6	0.0	0.0	19.05
		ALL-NDP	1,008	9,960	101,846	0	0	◇	◇	◇	◇	4	0.0	6.6	0.0	0.0	18.96
		ALL-SPP	1,008	9,960	101,846	0	0	◇	◇	◇	◇	4	0.0	6.4	0.0	0.0	18.84
		ALL-CTP	1,008	9,960	101,846	0	0	◇	◇	◇	◇	4	0.0	6.8	0.0	0.0	19.38
		ALL-CLQ	1,008	9,960	101,846	0	0	◇	◇	◇	◇	4	0.0	8.1	0.0	0.0	22.36
		ALL-LOH	1,008	9,960	101,846	0	0	◇	◇	◇	◇	4	0.0	6.8	0.0	0.0	20.87
		ALL-GSE	1,008	9,960	101,846	0	0	◇	◇	◇	◇	4	0.0	6.6	0.0	0.0	18.92
C3	29	ALL	1,008	9,960	101,846	0	27	25.2	0.0	-49	100	3,008	0.0	7.8	0.0	19.7	374.92
		ALL-NDP	1,008	9,960	101,846	0	23	25.2	0.0	-49	100	3,096	0.0	7.8	0.0	20.2	357.34
		ALL-SPP	1,008	9,960	101,846	0	11	26.6	0.0	-49	100	2,849	0.0	7.5	0.0	18.6	340.31
		ALL-CTP	1,008	9,960	101,846	0	27	25.2	0.0	-49	100	3,008	0.0	8.0	0.0	19.9	380.79
		ALL-CLQ	1,008	9,960	101,846	0	27	25.2	0.0	-49	100	3,008	0.0	9.5	0.0	23.8	441.73
		ALL-LOH	1,008	9,960	101,846	0	27	25.2	0.0	-49	100	3,008	0.0	8.0	0.0	19.9	380.40
		ALL-GSE	1,008	9,960	101,846	0	27	25.2	0.0	-49	100	3,008	0.0	7.7	0.0	19.6	372.80

◇ Problem is infeasible

Table 26 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C3	30	ALL	1,008	9,960	101,846	0	87	17.4	5.6	-77	6,308	32,717	0.0	21.9	0.0	299.5	3,600.00
		ALL-NDP	1,008	9,960	101,846	0	87	17.4	5.6	-77	6,308	32,724	0.0	21.7	0.0	300.1	3,600.00
		ALL-SPP	1,008	9,960	101,846	0	87	17.4	5.6	-77	6,308	32,764	0.0	21.5	0.0	299.8	3,600.00
		ALL-CTP	1,008	9,960	101,846	0	83	17.4	5.7	-77	6,308	31,673	0.0	21.8	0.0	298.1	3,600.00
		ALL-CLQ	1,008	9,960	101,846	0	71	17.4	6.3	-77	6,308	25,192	0.0	22.6	0.0	307.2	3,600.00
		ALL-LOH	1,008	9,960	101,846	0	83	17.4	5.7	-77	6,308	31,678	0.0	21.8	0.0	298.2	3,600.00
		ALL-GSE	1,008	9,960	101,846	0	87	17.4	5.6	-77	6,308	32,686	0.0	21.8	0.0	299.7	3,600.00
C3	31	ALL	1,004	10,096	61,483	0	5	22.1	0.0	-40	35	64	0.0	3.9	0.0	0.4	19.09
		ALL-NDP	1,004	10,096	61,483	0	5	22.1	0.0	-40	35	64	0.0	4.0	0.0	0.4	19.03
		ALL-SPP	1,004	10,096	61,483	0	5	22.1	0.0	-40	35	64	0.0	3.7	0.0	0.4	18.71
		ALL-CTP	1,004	10,096	61,483	0	5	22.1	0.0	-40	126	126	0.0	4.0	0.0	0.8	22.90
		ALL-CLQ	1,004	10,096	61,483	0	6	22.1	0.0	-40	35	64	0.0	4.8	0.0	0.5	22.66
		ALL-LOH	1,004	10,096	61,483	0	5	22.1	0.0	-40	35	64	0.0	4.0	0.0	0.4	20.51
		ALL-GSE	1,004	10,096	61,483	0	2	22.1	0.0	-40	30	71	0.0	3.9	0.0	0.5	19.80
C3	32	ALL	1,004	10,096	61,483	0	35	11.3	0.0	-75	1,725	3,894	0.0	5.5	0.0	20.0	180.78
		ALL-NDP	1,004	10,096	61,483	0	24	11.6	0.0	-75	1,988	3,995	0.0	5.4	0.0	19.5	170.93
		ALL-SPP	1,004	10,096	61,483	0	39	11.6	0.0	-75	3,051	4,990	0.0	5.8	0.0	28.1	225.94
		ALL-CTP	1,004	10,096	61,483	0	34	11.3	0.0	-75	1,276	3,463	0.0	5.3	0.0	17.6	162.93
		ALL-CLQ	1,004	10,096	61,483	0	35	11.3	0.0	-75	1,725	3,894	0.0	6.7	0.0	24.3	217.50
		ALL-LOH	1,004	10,096	61,483	0	35	11.3	0.0	-75	1,725	3,894	0.0	5.7	0.0	20.5	189.00
		ALL-GSE	1,004	10,096	61,483	0	16	11.6	0.0	-75	864	3,316	0.0	5.1	0.0	15.3	132.87
C3	33	ALL	1,004	10,096	61,483	0	65	5.8	0.0	-111	1,035	3,260	0.0	5.5	0.0	19.1	129.40
		ALL-NDP	1,004	10,096	61,483	0	66	5.8	0.0	-111	2,500	4,817	0.0	6.3	0.0	32.2	228.95
		ALL-SPP	1,004	10,096	61,483	0	67	6.0	0.0	-111	3,666	6,013	0.0	6.9	0.0	48.0	323.61
		ALL-CTP	1,004	10,096	61,483	0	69	5.8	0.0	-111	3,133	5,529	0.0	6.9	0.0	41.5	249.75
		ALL-CLQ	1,004	10,096	61,483	0	65	5.8	0.0	-111	1,035	3,260	0.0	6.7	0.0	23.1	155.37
		ALL-LOH	1,004	10,096	61,483	0	65	5.8	0.0	-111	1,035	3,260	0.0	5.7	0.0	19.5	137.23
		ALL-GSE	1,004	10,096	61,483	0	48	6.2	0.0	-111	1,233	4,243	0.0	5.7	0.0	24.6	178.64
C3	34	ALL	1,009	10,096	103,118	0	0	◇	◇	◇	◇	84	0.0	6.7	0.0	0.9	64.82
		ALL-NDP	1,009	10,096	103,118	0	0	◇	◇	◇	◇	84	0.0	6.7	0.0	0.9	64.48
		ALL-SPP	1,009	10,096	103,118	0	0	◇	◇	◇	◇	84	0.0	6.5	0.0	0.9	64.22
		ALL-CTP	1,009	10,096	103,118	0	0	◇	◇	◇	◇	84	0.0	6.9	0.0	0.9	65.70
		ALL-CLQ	1,009	10,096	103,118	0	0	◇	◇	◇	◇	84	0.0	8.3	0.0	1.1	74.70
		ALL-LOH	1,009	10,096	103,118	0	0	◇	◇	◇	◇	84	0.0	7.0	0.0	0.9	66.26
		ALL-GSE	1,009	10,096	103,118	0	0	◇	◇	◇	◇	84	0.0	6.8	0.0	0.9	64.37
C3	35	ALL	1,009	10,096	103,118	0	31	17.4	0.0	-59	1,722	2,678	0.0	7.8	0.0	22.5	345.54
		ALL-NDP	1,009	10,096	103,118	0	36	17.6	0.0	-59	3,165	4,007	0.0	8.6	0.0	39.1	469.91
		ALL-SPP	1,009	10,096	103,118	0	34	17.7	0.0	-59	4,284	5,122	0.0	9.1	0.0	53.0	649.72
		ALL-CTP	1,009	10,096	103,118	0	31	17.4	0.0	-59	1,722	2,678	0.0	7.9	0.0	22.7	350.80
		ALL-CLQ	1,009	10,096	103,118	0	31	17.4	0.0	-59	1,722	2,678	0.0	9.5	0.0	27.2	406.67
		ALL-LOH	1,009	10,096	103,118	0	31	17.4	0.0	-59	1,722	2,678	0.0	8.0	0.0	22.8	351.13
		ALL-GSE	1,009	10,096	103,118	0	31	17.4	0.0	-59	1,722	2,678	0.0	7.9	0.0	22.4	343.43

◇ Problem is infeasible

Table 26 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
C3	36	ALL	1,009	10,096	103,118	0	109	12.4	0.0	-86	5,834	16,596	0.0	14.3	0.0	153.4	1,466.39
		ALL-NDP	1,009	10,096	103,118	0	221	12.2	3.8	-86	23,372	24,347	0.0	21.9	0.0	341.0	3,600.00
		ALL-SPP	1,009	10,096	103,118	0	195	15.1	5.3	-84	14,935	27,774	0.0	21.3	0.0	319.6	3,600.00
		ALL-CTP	1,009	10,096	103,118	0	109	12.4	0.0	-86	5,834	16,596	0.0	14.4	0.0	155.5	1,497.14
		ALL-CLQ	1,009	10,096	103,118	0	109	12.4	0.0	-86	5,834	16,596	0.0	17.1	0.0	185.9	1,730.49
		ALL-LOH	1,009	10,096	103,118	0	109	12.4	0.0	-86	5,834	16,596	0.0	14.4	0.0	155.3	1,495.70
		ALL-GSE	1,009	10,096	103,118	0	126	12.5	0.0	-86	7,048	18,860	0.0	14.5	0.0	163.0	2,027.90

◇ Problem is infeasible

Table 27: RCSPP cutting plane results without arc fixing

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A2	3	NFX+ALL	99	935	2,655	0	5	1.0	0.0	-78	15	15	0.0	0.0	0.0	0.0	0.06
		NFX+ALL-NDP	99	935	2,655	0	5	1.0	0.0	-78	15	15	0.0	0.0	0.0	0.0	0.06
		NFX+ALL-SPP	99	935	2,655	0	1	1.0	0.0	-78	14	14	0.0	0.0	0.0	0.0	0.05
		NFX+ALL-CTP	99	935	2,655	0	5	1.0	0.0	-78	15	15	0.0	0.0	0.0	0.0	0.05
		NFX+ALL-CLQ	99	935	2,655	0	5	1.0	0.0	-78	15	15	0.0	0.0	0.0	0.0	0.06
		NFX+ALL-LOH	99	935	2,655	0	5	1.0	0.0	-78	15	15	0.0	0.0	0.0	0.0	0.06
		NFX+ALL-LPS	99	935	2,655	0	4	1.0	0.0	-78	17	17	0.0	0.0	0.0	0.0	0.05
A2	4	NFX+ALL	99	927	2,632	0	10	0.2	0.0	-74	2	2	0.0	0.0	0.0	0.0	0.07
		NFX+ALL-NDP	99	927	2,632	0	10	0.2	0.0	-74	2	2	0.0	0.0	0.0	0.0	0.07
		NFX+ALL-SPP	99	927	2,632	0	2	0.2	0.0	-74	2	2	0.0	0.0	0.0	0.0	0.06
		NFX+ALL-CTP	99	927	2,632	0	10	0.2	0.0	-74	2	2	0.0	0.0	0.0	0.0	0.05
		NFX+ALL-CLQ	99	927	2,632	0	10	0.2	0.0	-74	2	2	0.0	0.0	0.0	0.0	0.07
		NFX+ALL-LOH	99	927	2,632	0	10	0.2	0.0	-74	2	2	0.0	0.0	0.0	0.0	0.07
		NFX+ALL-LPS	99	927	2,632	0	4	0.3	0.0	-74	6	6	0.0	0.0	0.0	0.0	0.04
A2	7	NFX+ALL	89	708	7,405	0	92	8.5	0.0	-29	17	19	0.1	0.3	0.0	0.0	0.49
		NFX+ALL-NDP	89	708	7,405	0	92	8.5	0.0	-29	17	19	0.1	0.3	0.0	0.0	0.49
		NFX+ALL-SPP	89	708	7,405	0	283	11.0	0.0	-29	51	51	0.1	1.1	0.0	0.1	1.67
		NFX+ALL-CTP	89	708	7,405	0	146	8.2	0.0	-29	27	29	0.1	0.6	0.0	0.0	0.90
		NFX+ALL-CLQ	89	708	7,405	0	91	8.3	0.0	-29	5	7	0.1	0.2	0.0	0.0	0.40
		NFX+ALL-LOH	89	708	7,405	0	77	8.3	0.0	-29	5	7	0.1	0.4	0.0	0.0	0.56
		NFX+ALL-LPS	89	708	7,405	0	66	8.4	0.0	-29	21	24	0.1	0.0	0.0	0.0	0.22
A2	8	NFX+ALL	50	157	1,476	0	48	0.9	0.0	-23	1	1	0.1	0.0	0.0	0.0	0.08
		NFX+ALL-NDP	50	157	1,476	0	42	0.9	0.0	-23	1	1	0.1	0.0	0.0	0.0	0.08
		NFX+ALL-SPP	50	157	1,476	0	40	0.0	0.0	-23	0	0	0.1	0.0	0.0	0.0	0.08
		NFX+ALL-CTP	50	157	1,476	0	41	0.0	0.0	-23	0	0	0.1	0.0	0.0	0.0	0.08
		NFX+ALL-CLQ	50	157	1,476	0	47	0.0	0.0	-23	0	0	0.1	0.0	0.0	0.0	0.08
		NFX+ALL-LOH	50	157	1,476	0	50	0.0	0.0	-23	0	0	0.1	0.0	0.0	0.0	0.08
		NFX+ALL-LPS	50	157	1,476	0	85	0.0	0.0	-23	0	0	0.1	0.0	0.0	0.0	0.09
A2	11	NFX+ALL	195	1,915	5,471	0	10	0.0	0.0	-125	5	5	0.0	0.2	0.0	0.0	0.23
		NFX+ALL-NDP	195	1,915	5,471	0	10	0.0	0.0	-125	5	5	0.0	0.1	0.0	0.0	0.23
		NFX+ALL-SPP	195	1,915	5,471	0	2	0.0	0.0	-125	1	1	0.0	0.1	0.0	0.0	0.22
		NFX+ALL-CTP	195	1,915	5,471	0	10	0.0	0.0	-125	5	5	0.0	0.1	0.0	0.0	0.17
		NFX+ALL-CLQ	195	1,915	5,471	0	10	0.0	0.0	-125	5	5	0.0	0.1	0.0	0.0	0.23
		NFX+ALL-LOH	195	1,915	5,471	0	10	0.0	0.0	-125	5	5	0.0	0.1	0.0	0.0	0.23
		NFX+ALL-LPS	195	1,915	5,471	0	8	0.1	0.0	-125	1	1	0.0	0.1	0.0	0.0	0.18
A2	12	NFX+ALL	195	1,915	5,471	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.06
		NFX+ALL-NDP	195	1,915	5,471	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.06
		NFX+ALL-SPP	195	1,915	5,471	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.06
		NFX+ALL-CTP	195	1,915	5,471	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.06
		NFX+ALL-CLQ	195	1,915	5,471	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.06
		NFX+ALL-LOH	195	1,915	5,471	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.06
		NFX+ALL-LPS	195	1,915	5,471	0	0	0.4	0.0	-119	0	0	0.0	0.0	0.0	0.0	0.06

◇ Problem is infeasible

Table 27 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A2	15	NFX+ALL	151	1,179	12,201	0	109	29.0	0.0	-24	6	15	0.2	0.5	0.0	0.1	0.95
		NFX+ALL-NDP	151	1,179	12,201	0	108	29.0	0.0	-24	6	15	0.2	0.5	0.0	0.0	0.95
		NFX+ALL-SPP	151	1,179	12,201	0	68	34.5	0.0	-24	10	29	0.2	0.5	0.0	0.1	1.02
		NFX+ALL-CTP	151	1,179	12,201	0	52	30.0	0.0	-24	16	29	0.2	0.3	0.0	0.1	0.72
		NFX+ALL-CLQ	151	1,179	12,201	0	59	30.3	0.0	-24	28	34	0.2	0.2	0.0	0.1	0.70
		NFX+ALL-LOH	151	1,179	12,201	0	54	30.3	0.0	-24	14	28	0.2	0.2	0.0	0.1	0.65
		NFX+ALL-LPS	151	1,179	12,201	0	127	28.9	0.0	-24	6	23	0.2	0.1	0.0	0.1	0.65
A2	16	NFX+ALL	22	17	161	0	10	0.0	0.0	-17	0	0	0.1	0.0	0.0	0.0	0.08
		NFX+ALL-NDP	22	17	161	0	8	0.0	0.0	-17	0	0	0.1	0.0	0.0	0.0	0.09
		NFX+ALL-SPP	22	17	161	0	9	0.0	0.0	-17	0	0	0.1	0.0	0.0	0.0	0.09
		NFX+ALL-CTP	22	17	161	0	6	0.0	0.0	-17	0	0	0.1	0.0	0.0	0.0	0.09
		NFX+ALL-CLQ	22	17	161	0	9	0.0	0.0	-17	0	0	0.1	0.0	0.0	0.0	0.09
		NFX+ALL-LOH	22	17	161	0	10	0.0	0.0	-17	0	0	0.1	0.0	0.0	0.0	0.08
		NFX+ALL-LPS	22	17	161	0	8	5.7	0.0	-17	0	0	0.1	0.0	0.0	0.0	0.08
A2	19	NFX+ALL	474	4,682	13,373	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.21
		NFX+ALL-NDP	474	4,682	13,373	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.21
		NFX+ALL-SPP	474	4,682	13,373	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.21
		NFX+ALL-CTP	474	4,682	13,373	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.21
		NFX+ALL-CLQ	474	4,682	13,373	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.21
		NFX+ALL-LOH	474	4,682	13,373	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.21
		NFX+ALL-LPS	474	4,682	13,373	0	0	0.0	0.0	-99	0	0	0.1	0.0	0.0	0.0	0.21
A2	20	NFX+ALL	474	4,668	13,331	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.21
		NFX+ALL-NDP	474	4,668	13,331	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.21
		NFX+ALL-SPP	474	4,668	13,331	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.21
		NFX+ALL-CTP	474	4,668	13,331	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.21
		NFX+ALL-CLQ	474	4,668	13,331	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.21
		NFX+ALL-LOH	474	4,668	13,331	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.20
		NFX+ALL-LPS	474	4,668	13,331	0	0	0.0	0.0	-92	0	0	0.1	0.0	0.0	0.0	0.21
A2	23	NFX+ALL	471	4,471	48,353	0	42	22.9	0.0	-36	41	64	0.6	1.0	0.0	0.8	4.51
		NFX+ALL-NDP	471	4,471	48,353	0	42	22.9	0.0	-36	41	64	0.6	1.0	0.0	0.7	4.44
		NFX+ALL-SPP	471	4,471	48,353	0	4	25.3	0.0	-36	144	158	0.6	0.6	0.0	1.8	7.41
		NFX+ALL-CTP	471	4,471	48,353	0	42	22.9	0.0	-36	41	64	0.6	0.9	0.0	0.7	4.36
		NFX+ALL-CLQ	471	4,471	48,353	0	46	22.9	0.0	-36	100	126	0.6	1.0	0.0	1.6	8.29
		NFX+ALL-LOH	471	4,471	48,353	0	31	22.9	0.0	-36	24	63	0.6	1.0	0.0	0.8	4.79
		NFX+ALL-LPS	471	4,471	48,353	0	41	22.9	0.0	-36	47	72	0.6	0.5	0.0	0.8	4.35
A2	24	NFX+ALL	414	3,532	37,812	0	143	26.1	0.0	-29	17	17	1.0	1.4	0.0	0.2	3.65
		NFX+ALL-NDP	414	3,532	37,812	0	143	26.1	0.0	-29	17	17	1.0	1.4	0.0	0.2	3.64
		NFX+ALL-SPP	414	3,532	37,812	0	156	30.8	0.0	-29	2	11	1.0	3.6	0.0	0.1	5.70
		NFX+ALL-CTP	414	3,532	37,812	0	112	26.6	0.0	-29	28	30	1.0	1.2	0.0	0.4	3.85
		NFX+ALL-CLQ	414	3,532	37,812	0	140	26.8	0.0	-29	7	12	1.0	2.3	0.0	0.1	4.27
		NFX+ALL-LOH	414	3,532	37,812	0	122	26.5	0.0	-29	19	19	1.0	1.8	0.0	0.2	4.11
		NFX+ALL-LPS	414	3,532	37,812	0	123	26.6	0.0	-29	7	17	1.0	0.5	0.0	0.2	2.66

◇ Problem is infeasible

Table 27 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	1	NFX+ALL	861	8,335	50,727	0	22	14.3	0.0	-44	30	83	0.5	1.5	0.0	0.5	5.17
		NFX+ALL-SPP	861	8,335	50,727	0	22	14.3	0.0	-44	30	83	0.5	1.5	0.0	0.5	5.15
		NFX+ALL-CTP	861	8,335	50,727	0	22	14.3	0.0	-44	30	83	0.4	1.5	0.0	0.5	5.25
		NFX+ALL-CLQ	861	8,335	50,727	0	7	14.5	0.0	-44	23	64	0.4	1.4	0.0	0.4	4.52
		NFX+ALL-LOH	861	8,335	50,727	0	2	14.5	0.0	-44	107	157	0.4	1.5	0.0	1.0	8.05
		NFX+ALL-LPS	861	8,335	50,727	0	22	14.3	0.0	-44	30	83	0.4	1.2	0.0	0.5	4.86
A3	2	NFX+ALL	876	8,588	52,287	0	22	11.6	0.0	-64	618	748	0.4	1.6	0.0	4.3	19.70
		NFX+ALL-SPP	876	8,588	52,287	0	16	11.8	0.0	-64	100	438	0.4	1.5	0.0	2.6	13.26
		NFX+ALL-CTP	876	8,588	52,287	0	22	11.6	0.0	-64	618	748	0.4	1.6	0.0	4.3	19.89
		NFX+ALL-CLQ	876	8,588	52,287	0	36	11.6	0.0	-64	566	695	0.4	1.6	0.0	4.1	19.41
		NFX+ALL-LOH	876	8,588	52,287	0	3	11.6	0.0	-64	370	610	0.4	1.6	0.0	3.6	18.73
		NFX+ALL-LPS	876	8,588	52,287	0	26	11.7	0.0	-64	244	463	0.4	1.2	0.0	2.7	13.58
A3	3	NFX+ALL	876	8,596	52,337	0	6	6.7	0.0	-85	30	111	0.3	1.4	0.0	0.7	5.80
		NFX+ALL-SPP	876	8,596	52,337	0	6	6.7	0.0	-85	30	111	0.3	1.4	0.0	0.6	5.86
		NFX+ALL-CTP	876	8,596	52,337	0	6	6.7	0.0	-85	30	111	0.3	1.3	0.0	0.6	5.83
		NFX+ALL-CLQ	876	8,596	52,337	0	13	6.7	0.0	-85	33	103	0.3	1.4	0.0	0.6	6.20
		NFX+ALL-LOH	876	8,596	52,337	0	0	6.7	0.0	-85	30	98	0.3	1.4	0.0	0.6	5.64
		NFX+ALL-LPS	876	8,596	52,337	0	6	6.7	0.0	-85	30	111	0.3	1.2	0.0	0.6	5.67
A3	4	NFX+ALL	852	8,136	82,674	0	8	31.5	0.0	-32	60	167	1.0	2.1	0.0	1.7	17.09
		NFX+ALL-SPP	852	8,136	82,674	0	8	31.5	0.0	-32	60	167	1.0	2.1	0.0	1.7	17.07
		NFX+ALL-CTP	852	8,136	82,674	0	8	31.5	0.0	-32	60	167	1.0	2.2	0.0	1.7	17.39
		NFX+ALL-CLQ	852	8,136	82,674	0	3	31.6	0.0	-32	30	128	1.0	1.9	0.0	1.3	16.49
		NFX+ALL-LOH	852	8,136	82,674	0	2	31.5	0.0	-32	61	150	1.0	2.0	0.0	1.5	16.29
		NFX+ALL-LPS	852	8,136	82,674	0	8	31.5	0.0	-32	60	167	1.0	1.7	0.0	1.7	18.25
A3	5	NFX+ALL	881	8,588	87,286	0	3	16.2	0.0	-54	203	1,546	0.7	2.0	0.0	12.7	68.84
		NFX+ALL-SPP	881	8,588	87,286	0	3	16.2	0.0	-54	203	1,546	0.7	2.0	0.0	12.6	68.89
		NFX+ALL-CTP	881	8,588	87,286	0	3	16.2	0.0	-54	203	1,546	0.7	2.0	0.0	12.6	69.46
		NFX+ALL-CLQ	881	8,588	87,286	0	7	16.2	0.0	-54	187	1,547	0.7	2.0	0.0	12.8	77.97
		NFX+ALL-LOH	881	8,588	87,286	0	0	16.2	0.0	-54	180	1,503	0.7	2.0	0.0	12.3	69.81
		NFX+ALL-LPS	881	8,588	87,286	0	3	16.2	0.0	-54	203	1,546	0.7	1.7	0.0	12.6	69.93
A3	6	NFX+ALL	881	8,596	87,365	0	6	12.4	0.0	-73	59	2,494	0.5	2.0	0.0	17.7	79.39
		NFX+ALL-SPP	881	8,596	87,365	0	6	12.4	0.0	-73	59	2,494	0.5	2.0	0.0	17.6	79.36
		NFX+ALL-CTP	881	8,596	87,365	0	6	12.4	0.0	-73	59	2,494	0.5	2.0	0.0	17.7	79.32
		NFX+ALL-CLQ	881	8,596	87,365	0	9	12.4	0.0	-73	206	2,587	0.5	2.0	0.0	18.6	87.29
		NFX+ALL-LOH	881	8,596	87,365	0	0	12.4	0.0	-73	628	2,793	0.5	2.0	0.0	20.3	97.93
		NFX+ALL-LPS	881	8,596	87,365	0	6	12.4	0.0	-73	59	2,494	0.5	1.7	0.0	17.7	79.21
A3	7	NFX+ALL	878	8,572	52,137	0	17	16.5	0.0	-44	75	135	0.6	1.5	0.0	0.9	6.75
		NFX+ALL-SPP	878	8,572	52,137	0	6	18.4	0.0	-44	15	96	0.5	1.5	0.0	0.6	6.17
		NFX+ALL-CTP	878	8,572	52,137	0	17	16.5	0.0	-44	75	135	0.5	1.4	0.0	0.9	6.67
		NFX+ALL-CLQ	878	8,572	52,137	0	24	16.5	0.0	-44	76	143	0.6	1.6	0.0	0.9	6.98
		NFX+ALL-LOH	878	8,572	52,137	0	9	16.5	0.0	-44	100	145	0.6	1.6	0.0	0.9	7.47
		NFX+ALL-LPS	878	8,572	52,137	0	15	16.5	0.0	-44	47	115	0.6	1.2	0.0	0.8	6.45

◇ Problem is infeasible

Table 27 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	8	NFX+ALL	883	8,712	53,010	0	48	7.1	0.0	-70	89	160	0.4	1.9	0.0	1.0	9.02
		NFX+ALL-SPP	883	8,712	53,010	0	5	8.4	0.0	-70	110	173	0.4	1.4	0.0	1.0	7.60
		NFX+ALL-CTP	883	8,712	53,010	0	48	7.1	0.0	-70	89	160	0.4	1.8	0.0	1.0	8.94
		NFX+ALL-CLQ	883	8,712	53,010	0	41	7.1	0.0	-70	75	145	0.4	1.9	0.0	0.8	7.46
		NFX+ALL-LOH	883	8,712	53,010	0	20	7.1	0.0	-70	104	171	0.4	1.9	0.0	1.0	8.15
		NFX+ALL-LPS	883	8,712	53,010	0	48	7.1	0.0	-70	89	160	0.4	1.3	0.0	1.0	8.45
A3	9	NFX+ALL	883	8,712	53,010	0	23	5.9	0.0	-89	380	467	0.3	1.6	0.0	2.6	12.93
		NFX+ALL-SPP	883	8,712	53,010	0	7	6.0	0.0	-89	9	196	0.3	1.4	0.0	1.1	5.97
		NFX+ALL-CTP	883	8,712	53,010	0	23	5.9	0.0	-89	380	467	0.3	1.5	0.0	2.6	12.79
		NFX+ALL-CLQ	883	8,712	53,010	0	27	5.9	0.0	-89	518	658	0.3	1.6	0.0	3.6	19.25
		NFX+ALL-LOH	883	8,712	53,010	0	3	5.9	0.0	-89	533	533	0.3	1.5	0.0	3.0	12.90
		NFX+ALL-LPS	883	8,712	53,010	0	23	5.9	0.0	-89	380	467	0.3	1.3	0.0	2.6	12.57
A3	10	NFX+ALL	865	8,385	85,175	0	2	40.6	0.0	-33	150	405	1.0	1.9	0.0	4.1	41.24
		NFX+ALL-SPP	865	8,385	85,175	0	2	40.6	0.0	-33	150	405	1.0	2.0	0.0	4.1	41.16
		NFX+ALL-CTP	865	8,385	85,175	0	2	40.6	0.0	-33	150	405	1.0	1.9	0.0	4.1	41.17
		NFX+ALL-CLQ	865	8,385	85,175	0	2	40.6	0.0	-33	170	423	1.0	2.0	0.0	4.3	48.81
		NFX+ALL-LOH	865	8,385	85,175	0	0	40.6	0.0	-33	153	439	1.0	2.0	0.0	4.2	42.50
		NFX+ALL-LPS	865	8,385	85,175	0	2	40.6	0.0	-33	150	405	1.0	1.7	0.0	4.1	48.28
A3	11	NFX+ALL	888	8,712	88,550	0	4	15.1	0.0	-60	150	867	0.7	2.0	0.0	7.3	48.39
		NFX+ALL-SPP	888	8,712	88,550	0	4	15.1	0.0	-60	150	867	0.7	2.0	0.0	7.3	48.50
		NFX+ALL-CTP	888	8,712	88,550	0	4	15.1	0.0	-60	150	867	0.7	2.0	0.0	7.3	48.74
		NFX+ALL-CLQ	888	8,712	88,550	0	9	15.1	0.0	-60	296	929	0.7	2.0	0.0	7.7	52.61
		NFX+ALL-LOH	888	8,712	88,550	0	0	15.1	0.0	-60	280	910	0.7	2.0	0.0	7.7	51.31
		NFX+ALL-LPS	888	8,712	88,550	0	4	15.1	0.0	-60	150	867	0.7	1.7	0.0	7.2	50.02
A3	12	NFX+ALL	888	8,712	88,550	0	39	11.6	0.0	-79	3,902	4,635	0.4	2.5	0.0	33.9	163.27
		NFX+ALL-SPP	888	8,712	88,550	0	17	11.7	0.0	-79	1,147	3,393	0.4	2.8	0.0	24.7	133.53
		NFX+ALL-CTP	888	8,712	88,550	0	39	11.6	0.0	-79	3,902	4,635	0.4	2.4	0.0	33.9	164.83
		NFX+ALL-CLQ	888	8,712	88,550	0	47	11.6	0.0	-79	1,644	3,826	0.4	2.6	0.0	28.4	165.47
		NFX+ALL-LOH	888	8,712	88,550	0	33	11.6	0.0	-79	1,630	3,704	0.4	2.5	0.0	27.2	147.88
		NFX+ALL-LPS	888	8,712	88,550	0	35	11.6	0.0	-79	4,429	5,177	0.4	1.8	0.0	38.1	192.51
A3	13	NFX+ALL	1,085	10,312	62,518	0	51	18.6	0.0	-42	64	90	0.7	2.7	0.0	0.7	9.51
		NFX+ALL-SPP	1,085	10,312	62,518	0	59	19.9	0.0	-42	21	71	0.7	3.0	0.0	0.6	9.22
		NFX+ALL-CTP	1,085	10,312	62,518	0	55	18.5	0.0	-42	3	52	0.7	3.1	0.0	0.4	8.66
		NFX+ALL-CLQ	1,085	10,312	62,518	0	54	17.6	0.0	-42	36	76	0.7	2.6	0.0	0.6	8.51
		NFX+ALL-LOH	1,085	10,312	62,518	0	38	17.6	0.0	-42	8	61	0.7	3.2	0.0	0.5	8.93
		NFX+ALL-LPS	1,085	10,312	62,518	0	49	18.5	0.0	-42	46	89	0.7	2.0	0.0	0.7	8.75
A3	14	NFX+ALL	1,098	10,671	64,763	0	6	10.0	0.0	-67	9	76	0.6	2.0	0.0	0.5	7.56
		NFX+ALL-SPP	1,098	10,671	64,763	0	6	10.0	0.0	-67	9	76	0.6	2.0	0.0	0.5	7.59
		NFX+ALL-CTP	1,098	10,671	64,763	0	6	10.0	0.0	-67	9	76	0.6	2.0	0.0	0.5	7.69
		NFX+ALL-CLQ	1,098	10,671	64,763	0	11	10.0	0.0	-67	8	77	0.6	2.0	0.0	0.5	7.64
		NFX+ALL-LOH	1,098	10,671	64,763	0	0	10.0	0.0	-67	11	76	0.6	2.0	0.0	0.5	7.46
		NFX+ALL-LPS	1,098	10,671	64,763	0	6	10.0	0.0	-67	9	76	0.6	1.8	0.0	0.5	7.39

◇ Problem is infeasible

Table 27 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	15	NFX+ALL	1,098	10,671	64,763	0	8	5.9	0.0	-89	20	53	0.4	2.1	0.0	0.4	6.82
		NFX+ALL-SPP	1,098	10,671	64,763	0	8	5.9	0.0	-89	20	53	0.4	2.1	0.0	0.4	6.97
		NFX+ALL-CTP	1,098	10,671	64,763	0	8	5.9	0.0	-89	20	53	0.4	2.0	0.0	0.4	6.87
		NFX+ALL-CLQ	1,098	10,671	64,763	0	13	5.9	0.0	-89	20	52	0.4	2.1	0.0	0.4	7.08
		NFX+ALL-LOH	1,098	10,671	64,763	0	0	5.9	0.0	-89	30	77	0.4	2.1	0.0	0.6	6.89
		NFX+ALL-LPS	1,098	10,671	64,763	0	8	5.9	0.0	-89	20	53	0.4	1.9	0.0	0.4	6.81
A3	16	NFX+ALL	1,033	9,720	98,838	0	140	34.3	0.0	-32	68	187	1.8	10.2	0.0	2.1	40.01
		NFX+ALL-SPP	1,033	9,720	98,838	0	140	34.3	0.0	-32	68	187	1.8	10.1	0.0	2.1	39.95
		NFX+ALL-CTP	1,033	9,720	98,838	0	70	34.6	0.0	-32	67	183	1.8	6.1	0.0	2.1	33.44
		NFX+ALL-CLQ	1,033	9,720	98,838	0	19	35.7	0.0	-32	58	173	1.8	4.0	0.0	1.9	27.59
		NFX+ALL-LOH	1,033	9,720	98,838	0	2	35.8	0.0	-32	74	191	1.8	3.2	0.0	2.1	27.15
		NFX+ALL-LPS	1,033	9,720	98,838	0	2	36.2	0.0	-32	44	181	1.8	2.5	0.0	2.0	24.93
A3	17	NFX+ALL	1,103	10,671	108,673	0	7	22.3	0.0	-55	60	1,506	1.2	3.0	0.0	15.6	110.00
		NFX+ALL-SPP	1,103	10,671	108,673	0	7	22.3	0.0	-55	60	1,506	1.2	3.0	0.0	15.5	111.18
		NFX+ALL-CTP	1,103	10,671	108,673	0	7	22.3	0.0	-55	60	1,506	1.2	3.0	0.0	15.5	110.97
		NFX+ALL-CLQ	1,103	10,671	108,673	0	7	22.3	0.0	-55	165	1,627	1.2	3.1	0.0	16.8	123.28
		NFX+ALL-LOH	1,103	10,671	108,673	0	0	22.3	0.0	-55	53	1,497	1.2	3.0	0.0	15.4	108.99
		NFX+ALL-LPS	1,103	10,671	108,673	0	7	22.3	0.0	-55	60	1,506	1.2	2.6	0.0	15.5	111.20
A3	18	NFX+ALL	1,103	10,671	108,673	0	17	11.1	0.0	-78	80	937	0.7	3.4	0.0	8.6	57.88
		NFX+ALL-SPP	1,103	10,671	108,673	0	4	11.3	0.0	-78	86	860	0.7	3.0	0.0	8.0	53.08
		NFX+ALL-CTP	1,103	10,671	108,673	0	17	11.1	0.0	-78	80	937	0.7	3.3	0.0	8.6	58.37
		NFX+ALL-CLQ	1,103	10,671	108,673	0	17	11.1	0.0	-78	244	1,017	0.7	3.4	0.0	9.4	69.36
		NFX+ALL-LOH	1,103	10,671	108,673	0	8	11.1	0.0	-78	80	890	0.7	3.4	0.0	8.1	55.98
		NFX+ALL-LPS	1,103	10,671	108,673	0	17	11.1	0.0	-78	80	937	0.7	2.7	0.0	8.6	57.98
A3	19	NFX+ALL	1,052	10,187	61,817	0	15	12.2	0.0	-50	109	123	0.9	2.2	0.0	1.1	11.25
		NFX+ALL-SPP	1,052	10,187	61,817	0	15	12.2	0.0	-50	109	123	0.9	2.3	0.0	1.1	11.24
		NFX+ALL-CTP	1,052	10,187	61,817	0	15	12.2	0.0	-50	109	123	0.9	2.2	0.0	1.1	11.27
		NFX+ALL-CLQ	1,052	10,187	61,817	0	13	12.2	0.0	-50	95	108	0.9	2.0	0.0	1.0	10.35
		NFX+ALL-LOH	1,052	10,187	61,817	0	2	12.2	0.0	-50	92	115	0.9	2.3	0.0	1.0	12.18
		NFX+ALL-LPS	1,052	10,187	61,817	0	11	12.2	0.0	-50	86	108	0.9	1.8	0.0	0.9	10.55
A3	20	NFX+ALL	1,054	10,241	62,171	0	22	12.0	0.0	-74	119	317	0.6	2.1	0.0	2.2	17.07
		NFX+ALL-SPP	1,054	10,241	62,171	0	2	13.4	0.0	-74	52	260	0.6	2.0	0.0	1.8	14.96
		NFX+ALL-CTP	1,054	10,241	62,171	0	22	12.0	0.0	-74	119	317	0.6	2.1	0.0	2.1	17.19
		NFX+ALL-CLQ	1,054	10,241	62,171	0	26	12.0	0.0	-74	120	340	0.6	2.2	0.0	2.3	18.88
		NFX+ALL-LOH	1,054	10,241	62,171	0	13	12.0	0.0	-74	210	387	0.6	2.1	0.0	2.6	19.93
		NFX+ALL-LPS	1,054	10,241	62,171	0	22	12.0	0.0	-74	119	317	0.6	1.9	0.0	2.1	16.73
A3	21	NFX+ALL	1,054	10,241	62,171	0	43	8.5	0.0	-97	605	632	0.4	3.0	0.0	4.2	29.09
		NFX+ALL-SPP	1,054	10,241	62,171	0	6	8.9	0.0	-97	387	507	0.4	2.1	0.0	3.2	21.81
		NFX+ALL-CTP	1,054	10,241	62,171	0	43	8.5	0.0	-97	605	632	0.4	2.8	0.0	4.2	29.01
		NFX+ALL-CLQ	1,054	10,241	62,171	0	53	8.5	0.0	-97	618	666	0.4	3.2	0.0	4.3	31.70
		NFX+ALL-LOH	1,054	10,241	62,171	0	14	8.5	0.0	-97	418	557	0.4	3.0	0.0	3.6	23.47
		NFX+ALL-LPS	1,054	10,241	62,171	0	43	8.5	0.0	-97	605	632	0.4	2.2	0.0	4.2	28.53

◇ Problem is infeasible

Table 27 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	22	NFX+ALL	1,037	9,934	101,100	0	12	28.7	0.0	-36	114	174	1.8	3.2	0.0	2.0	30.09
		NFX+ALL-SPP	1,037	9,934	101,100	0	12	28.7	0.0	-36	114	174	1.8	3.2	0.0	2.0	29.82
		NFX+ALL-CTP	1,037	9,934	101,100	0	16	28.6	0.0	-36	50	134	1.8	3.8	0.0	1.6	22.37
		NFX+ALL-CLQ	1,037	9,934	101,100	0	4	28.9	0.0	-36	257	260	1.8	2.8	0.0	3.0	34.50
		NFX+ALL-LOH	1,037	9,934	101,100	0	3	28.7	0.0	-36	104	198	1.8	3.6	0.0	2.3	32.25
		NFX+ALL-LPS	1,037	9,934	101,100	0	12	28.7	0.0	-36	114	174	1.8	2.5	0.0	2.0	30.01
A3	23	NFX+ALL	1,059	10,241	104,275	0	4	16.6	0.0	-59	88	472	1.1	2.9	0.0	4.9	49.05
		NFX+ALL-SPP	1,059	10,241	104,275	0	4	16.6	0.0	-59	88	472	1.1	2.9	0.0	4.9	48.53
		NFX+ALL-CTP	1,059	10,241	104,275	0	4	16.6	0.0	-59	88	472	1.1	2.9	0.0	4.9	49.23
		NFX+ALL-CLQ	1,059	10,241	104,275	0	7	16.6	0.0	-59	83	453	1.0	3.0	0.0	4.7	49.93
		NFX+ALL-LOH	1,059	10,241	104,275	0	0	16.6	0.0	-59	103	482	1.1	2.9	0.0	5.0	48.88
		NFX+ALL-LPS	1,059	10,241	104,275	0	4	16.6	0.0	-59	88	472	1.1	2.5	0.0	4.9	48.03
A3	24	NFX+ALL	1,059	10,241	104,275	0	4	11.7	0.0	-79	64	923	0.7	2.9	0.0	8.6	60.69
		NFX+ALL-SPP	1,059	10,241	104,275	0	4	11.7	0.0	-79	64	923	0.7	2.9	0.0	8.6	59.65
		NFX+ALL-CTP	1,059	10,241	104,275	0	4	11.7	0.0	-79	64	923	0.7	2.9	0.0	8.6	59.63
		NFX+ALL-CLQ	1,059	10,241	104,275	0	11	11.7	0.0	-79	437	1,336	0.7	2.9	0.0	12.4	93.66
		NFX+ALL-LOH	1,059	10,241	104,275	0	0	11.7	0.0	-79	60	895	0.7	2.9	0.0	8.3	57.90
		NFX+ALL-LPS	1,059	10,241	104,275	0	4	11.7	0.0	-79	64	923	0.7	2.6	0.0	8.6	59.97
A3	25	NFX+ALL	1,187	10,917	66,231	0	57	9.5	0.0	-46	112	117	1.0	3.6	0.0	1.0	13.78
		NFX+ALL-SPP	1,187	10,917	66,231	0	57	9.5	0.0	-46	112	117	1.0	3.5	0.0	1.0	13.82
		NFX+ALL-CTP	1,187	10,917	66,231	0	8	9.6	0.0	-46	25	34	1.0	2.7	0.0	0.3	7.98
		NFX+ALL-CLQ	1,187	10,917	66,231	0	20	9.5	0.0	-46	24	34	1.0	3.1	0.0	0.3	8.44
		NFX+ALL-LOH	1,187	10,917	66,231	0	7	9.5	0.0	-46	109	113	1.0	3.1	0.0	1.1	13.70
		NFX+ALL-LPS	1,187	10,917	66,231	0	45	9.5	0.0	-46	73	87	1.1	2.6	0.0	0.8	10.91
A3	26	NFX+ALL	1,250	11,947	72,646	0	25	9.7	0.0	-65	180	264	0.8	3.4	0.0	2.2	17.26
		NFX+ALL-SPP	1,250	11,947	72,646	0	3	9.7	0.0	-65	218	277	0.8	2.7	0.0	2.4	19.40
		NFX+ALL-CTP	1,250	11,947	72,646	0	25	9.7	0.0	-65	180	264	0.8	3.5	0.0	2.2	17.22
		NFX+ALL-CLQ	1,250	11,947	72,646	0	19	9.7	0.0	-65	29	164	0.8	3.0	0.0	1.4	12.76
		NFX+ALL-LOH	1,250	11,947	72,646	0	4	9.7	0.0	-65	8	121	0.8	3.4	0.0	1.1	10.96
		NFX+ALL-LPS	1,250	11,947	72,646	0	12	9.7	0.0	-65	97	226	0.8	2.6	0.0	2.0	16.60
A3	27	NFX+ALL	1,252	11,991	72,938	0	31	5.8	0.0	-86	110	189	0.6	4.2	0.0	1.5	22.23
		NFX+ALL-SPP	1,252	11,991	72,938	0	6	6.1	0.0	-86	250	304	0.6	2.9	0.0	2.6	23.47
		NFX+ALL-CTP	1,252	11,991	72,938	0	31	5.8	0.0	-86	110	189	0.6	3.5	0.0	1.5	19.00
		NFX+ALL-CLQ	1,252	11,991	72,938	0	36	5.8	0.0	-86	20	77	0.6	3.7	0.0	0.6	11.61
		NFX+ALL-LOH	1,252	11,991	72,938	0	4	5.9	0.0	-86	90	165	0.6	3.4	0.0	1.4	17.66
		NFX+ALL-LPS	1,252	11,991	72,938	0	19	5.9	0.0	-86	20	81	0.6	2.8	0.0	0.6	10.00
A3	28	NFX+ALL	1,105	10,020	102,017	0	14	26.6	0.0	-35	180	180	2.5	4.0	0.0	2.5	40.91
		NFX+ALL-SPP	1,105	10,020	102,017	0	14	26.6	0.0	-35	180	180	2.5	3.9	0.0	2.5	35.65
		NFX+ALL-CTP	1,105	10,020	102,017	0	14	26.6	0.0	-35	180	180	2.5	3.9	0.0	2.5	35.96
		NFX+ALL-CLQ	1,105	10,020	102,017	0	8	26.8	0.0	-35	192	192	2.5	3.7	0.0	2.6	41.56
		NFX+ALL-LOH	1,105	10,020	102,017	0	4	26.6	0.0	-35	40	105	2.5	3.7	0.0	1.3	21.02
		NFX+ALL-LPS	1,105	10,020	102,017	0	12	26.6	0.0	-35	27	95	2.5	3.2	0.0	1.1	20.06

◇ Problem is infeasible

Table 27 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	29	NFX+ALL	1,254	11,931	121,785	0	1	18.0	0.0	-55	479	954	1.5	4.0	0.0	11.8	128.68
		NFX+ALL-SPP	1,254	11,931	121,785	0	1	18.0	0.0	-55	479	954	1.5	4.0	0.0	11.8	106.29
		NFX+ALL-CTP	1,254	11,931	121,785	0	1	18.0	0.0	-55	479	954	1.5	4.0	0.0	11.8	105.29
		NFX+ALL-CLQ	1,254	11,931	121,785	0	6	18.0	0.0	-55	433	903	1.6	4.3	0.0	11.2	107.28
		NFX+ALL-LOH	1,254	11,931	121,785	0	0	18.0	0.0	-55	300	860	1.5	4.0	0.0	10.6	95.96
		NFX+ALL-LPS	1,254	11,931	121,785	0	1	18.0	0.0	-55	479	954	1.5	3.8	0.0	11.9	136.13
A3	30	NFX+ALL	1,257	11,991	122,393	0	25	12.7	0.0	-74	490	1,281	1.1	5.0	0.0	14.2	129.60
		NFX+ALL-SPP	1,257	11,991	122,393	0	3	12.9	0.0	-74	122	976	1.1	4.1	0.0	10.9	86.54
		NFX+ALL-CTP	1,257	11,991	122,393	0	25	12.7	0.0	-74	490	1,281	1.1	4.9	0.0	14.3	120.34
		NFX+ALL-CLQ	1,257	11,991	122,393	0	23	12.7	0.0	-74	1,908	2,291	1.1	5.0	0.0	25.5	193.28
		NFX+ALL-LOH	1,257	11,991	122,393	0	8	12.7	0.0	-74	648	1,428	1.1	5.0	0.0	16.0	133.39
		NFX+ALL-LPS	1,257	11,991	122,393	0	25	12.7	0.0	-74	490	1,281	1.1	4.1	0.0	14.3	131.35
A3	31	NFX+ALL	1,276	12,182	74,022	0	64	13.9	0.0	-45	4	34	1.0	5.2	0.0	0.3	12.51
		NFX+ALL-SPP	1,276	12,182	74,022	0	112	18.0	0.0	-45	20	71	1.0	7.1	0.0	0.7	18.86
		NFX+ALL-CTP	1,276	12,182	74,022	0	44	14.0	0.0	-45	4	48	1.0	3.9	0.0	0.4	10.89
		NFX+ALL-CLQ	1,276	12,182	74,022	0	37	14.1	0.0	-45	3	39	1.0	4.2	0.0	0.4	10.89
		NFX+ALL-LOH	1,276	12,182	74,022	0	27	14.0	0.0	-45	4	42	1.0	4.6	0.0	0.4	11.65
		NFX+ALL-LPS	1,276	12,182	74,022	0	61	13.9	0.0	-45	4	32	1.0	3.3	0.0	0.3	10.66
A3	32	NFX+ALL	1,288	12,501	76,002	0	13	6.5	0.0	-68	6	40	0.5	3.0	0.0	0.4	9.53
		NFX+ALL-SPP	1,288	12,501	76,002	0	3	6.8	0.0	-68	65	93	0.5	2.8	0.0	0.8	11.99
		NFX+ALL-CTP	1,288	12,501	76,002	0	13	6.5	0.0	-68	6	40	0.5	2.9	0.0	0.4	8.84
		NFX+ALL-CLQ	1,288	12,501	76,002	0	10	6.5	0.0	-68	19	55	0.5	3.0	0.0	0.5	10.83
		NFX+ALL-LOH	1,288	12,501	76,002	0	1	6.5	0.0	-68	10	47	0.5	3.0	0.0	0.4	9.02
		NFX+ALL-LPS	1,288	12,501	76,002	0	13	6.5	0.0	-68	6	40	0.5	2.7	0.0	0.4	8.64
A3	33	NFX+ALL	1,288	12,502	76,009	0	32	5.4	0.0	-86	200	317	0.4	3.4	0.0	2.7	27.87
		NFX+ALL-SPP	1,288	12,502	76,009	0	32	5.4	0.0	-86	200	317	0.4	3.5	0.0	2.7	26.17
		NFX+ALL-CTP	1,288	12,502	76,009	0	32	5.4	0.0	-86	200	317	0.4	3.2	0.0	2.7	26.27
		NFX+ALL-CLQ	1,288	12,502	76,009	0	13	5.5	0.0	-86	110	197	0.4	3.2	0.0	1.6	15.95
		NFX+ALL-LOH	1,288	12,502	76,009	0	1	5.5	0.0	-86	120	229	0.4	3.3	0.0	1.9	22.39
		NFX+ALL-LPS	1,288	12,502	76,009	0	6	5.8	0.0	-86	628	651	0.4	2.7	0.0	5.9	42.17
A3	34	NFX+ALL	1,255	11,872	121,108	0	38	27.6	0.0	-36	23	105	2.3	6.4	0.0	1.6	35.54
		NFX+ALL-SPP	1,255	11,872	121,108	0	38	27.6	0.0	-36	23	105	2.1	6.1	0.0	1.6	35.02
		NFX+ALL-CTP	1,255	11,872	121,108	0	9	27.9	0.0	-36	31	108	2.2	4.3	0.0	1.5	30.47
		NFX+ALL-CLQ	1,255	11,872	121,108	0	6	28.0	0.0	-36	5	102	2.2	4.1	0.0	1.4	31.51
		NFX+ALL-LOH	1,255	11,872	121,108	0	12	27.7	0.0	-36	32	108	2.1	5.9	0.0	1.5	33.10
		NFX+ALL-LPS	1,255	11,872	121,108	0	38	27.6	0.0	-36	23	105	2.1	3.9	0.0	1.6	32.50
A3	35	NFX+ALL	1,293	12,498	127,513	0	4	18.2	0.0	-57	50	1,087	0.9	4.3	0.0	13.8	106.55
		NFX+ALL-SPP	1,293	12,498	127,513	0	4	18.2	0.0	-57	50	1,087	0.9	4.3	0.0	13.4	107.07
		NFX+ALL-CTP	1,293	12,498	127,513	0	4	18.2	0.0	-57	50	1,087	0.9	4.2	0.0	13.4	107.13
		NFX+ALL-CLQ	1,293	12,498	127,513	0	7	18.2	0.0	-57	150	1,115	0.9	4.3	0.0	14.3	138.92
		NFX+ALL-LOH	1,293	12,498	127,513	0	0	18.2	0.0	-57	110	1,116	0.9	4.2	0.0	14.0	108.01
		NFX+ALL-LPS	1,293	12,498	127,513	0	4	18.2	0.0	-57	50	1,087	0.9	3.9	0.0	13.5	131.52

◇ Problem is infeasible

Table 27 (continued)

Class	Instance	Setting	Rows	Cols	Nzs	Fixes	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_P	t_C	t_H	t_B	t
A3	36	NFX+ALL	1,293	12,502	127,555	0	3	10.9	0.0	-77	49	1,283	0.7	4.2	0.0	14.5	98.07
		NFX+ALL-SPP	1,293	12,502	127,555	0	3	10.9	0.0	-77	49	1,283	0.7	4.4	0.0	14.5	97.79
		NFX+ALL-CTP	1,293	12,502	127,555	0	3	10.9	0.0	-77	49	1,283	0.7	4.3	0.0	14.5	97.29
		NFX+ALL-CLQ	1,293	12,502	127,555	0	5	10.9	0.0	-77	60	1,415	0.7	4.2	0.0	15.6	104.04
		NFX+ALL-LOH	1,293	12,502	127,555	0	0	10.9	0.0	-77	60	1,371	0.7	4.3	0.0	15.3	105.62
		NFX+ALL-LPS	1,293	12,502	127,555	0	3	10.9	0.0	-77	49	1,283	0.7	3.9	0.0	14.6	107.45

◇ Problem is infeasible

APPENDIX B

COMPUTATIONAL RESULTS FOR THE DIAL-A-FLIGHT PROBLEM

Table 28: DAFP default CPLEX results

Class	Instance	Setting	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_C	t_B	t
A	1	CPX	0	0.8	0.0	2,608	28	32	0.0	0.0	16.51
		CPX-C	0	1.2	0.0	2,608	581	1,103	0.0	0.0	43.63
A	2	CPX	0	3.8	0.0	2,888	318	579	0.0	0.0	43.02
		CPX-C	0	3.8	0.0	2,888	119	335	0.0	0.0	22.97
A	3	CPX	0	3.3	0.0	3,106	1,608	4,324	0.0	0.0	412.47
		CPX-C	0	3.3	0.0	3,106	6,368	22,290	0.0	0.0	1,244.65
A	4	CPX	0	3.3	0.0	3,105	439	6,592	0.0	0.0	1,542.72
		CPX-C	0	3.3	0.0	3,105	13	3,570	0.0	0.0	385.67
A	5	CPX	0	2.9	2.5	2,797	17,882	25,887	0.0	0.0	3,600.00
		CPX-C	0	3.5	2.6	2,797	12,121	28,293	0.0	0.0	3,600.00
A	6	CPX	0	2.4	0.0	3,226	1,640	2,887	0.0	0.0	224.43
		CPX-C	0	4.1	0.0	3,226	1,532	8,481	0.0	0.0	396.28
A	7	CPX	0	1.0	0.0	3,314	1,093	2,443	0.0	0.0	231.22
		CPX-C	0	3.4	0.0	3,314	1,539	9,336	0.0	0.0	636.22
A	8	CPX	0	0.3	0.0	2,894	16	38	0.0	0.0	20.65
		CPX-C	0	1.3	0.0	2,894	131	724	0.0	0.0	49.41
A	9	CPX	0	4.4	3.3	2,994	5,464	28,274	0.0	0.0	3,600.00
		CPX-C	0	4.9	3.0	2,994	8,210	30,250	0.0	0.0	3,600.00
A	10	CPX	0	6.0	4.6	2,851	27,156	34,341	0.0	0.0	3,600.00
		CPX-C	0	8.1	4.6	2,828	16,137	52,337	0.0	0.0	3,600.00
A	11	CPX	0	4.4	0.0	2,410	2,576	30,258	0.0	0.0	1,458.69
		CPX-C	0	6.2	0.0	2,410	5,040	21,208	0.0	0.0	1,023.03
A	12	CPX	0	0.3	0.0	2,587	2	4	0.0	0.0	13.26
		CPX-C	0	4.2	0.0	2,587	3,236	6,219	0.0	0.0	294.60
A	13	CPX	0	2.6	0.0	3,371	2,600	6,115	0.0	0.0	3,151.88
		CPX-C	0	4.0	2.8	3,371	10,158	11,912	0.0	0.0	3,600.00
A	14	CPX	0	2.7	0.0	2,976	95	436	0.0	0.0	47.34
		CPX-C	0	4.7	0.0	2,976	528	1,608	0.0	0.0	106.63
A	15	CPX	0	1.7	0.0	3,133	569	2,009	0.0	0.0	615.83
		CPX-C	0	5.3	0.0	3,133	927	18,548	0.0	0.0	2,223.51
A	16	CPX	0	0.3	0.0	3,445	39	115	0.0	0.0	36.24
		CPX-C	0	2.7	0.0	3,445	430	844	0.0	0.0	98.42
A	17	CPX	0	3.1	0.0	2,894	3,276	6,606	0.0	0.0	1,239.45
		CPX-C	0	6.2	2.6	2,894	5,280	29,988	0.0	0.0	3,600.00
A	18	CPX	0	7.4	4.9	2,959	16,879	21,758	0.0	0.0	3,600.00
		CPX-C	0	7.7	5.4	2,959	25,692	29,907	0.0	0.0	3,600.00
A	19	CPX	0	4.0	2.4	3,356	2,660	20,768	0.0	0.0	3,600.00
		CPX-C	0	5.2	2.1	3,356	6,374	23,007	0.0	0.0	3,600.00
A	20	CPX	0	2.8	0.3	2,879	522	61,226	0.0	0.0	3,600.00
		CPX-C	0	4.1	1.2	2,879	3,018	61,261	0.0	0.0	3,600.00
B	1	CPX	0	0.0	0.0	3,128	◇	0	0.0	0.0	12.91
		CPX-C	0	4.2	0.0	3,128	◇	678	0.0	0.0	391.02
B	2	CPX	0	3.3	0.0	2,971	975	1,805	0.0	0.0	892.63
		CPX-C	0	6.2	3.5	3,007	10,595	14,947	0.0	0.0	3,600.00
B	3	CPX	0	5.6	4.9	3,355	2,530	2,539	0.0	0.0	3,599.77
		CPX-C	0	8.6	8.6	3,396	◇	1,941	0.0	0.0	3,599.84
B	4	CPX	0	3.2	0.8	3,165	876	8,720	0.0	0.0	3,600.00
		CPX-C	0	5.1	2.7	3,165	7,257	7,276	0.0	0.0	3,599.88
B	5	CPX	0	1.7	0.0	3,063	238	529	0.0	0.0	198.09
		CPX-C	0	2.9	0.0	3,063	110	386	0.0	0.0	171.56
B	6	CPX	0	2.0	0.0	3,557	20	69	0.0	0.0	20.07
		CPX-C	0	2.4	0.0	3,557	46	1,287	0.0	0.0	57.18
B	7	CPX	0	4.8	2.1	2,865	5,201	9,427	0.0	0.0	3,600.00
		CPX-C	0	7.3	4.6	2,865	7,071	9,064	0.0	0.0	3,600.00

◇ No improvement found

Table 28 (continued)

Class	Instance	Setting	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_C	t_B	t
B	8	CPX	0	1.2	0.0	3,510	249	502	0.0	0.0	212.40
		CPX-C	0	2.7	0.0	3,510	1,289	1,998	0.0	0.0	773.33
B	9	CPX	0	4.2	2.3	3,113	7,650	8,395	0.0	0.0	3,600.00
		CPX-C	0	6.8	4.1	3,105	6,834	7,999	0.0	0.0	3,599.93
B	10	CPX	0	10.9	10.8	3,216	◇	2,932	0.0	0.0	3,599.88
		CPX-C	0	13.5	13.5	3,216	◇	3,136	0.0	0.0	3,599.76

◇ No improvement found

Table 29: DAFP branching scheme results

Class	Instance	Setting	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_C	t_B	t
A	1	ARC	0	0.8	0.0	2,608	6	41	0.0	0.0	12.58
		ND1	0	0.8	0.0	2,608	13	64	0.0	0.1	12.85
		ND2	0	0.8	0.0	2,608	12	279	0.0	0.3	22.07
		ND3	0	0.8	0.0	2,608	21	27	0.0	0.0	12.89
A	2	ARC	0	3.8	0.0	2,888	110	223	0.0	0.2	22.45
		ND1	0	3.8	0.0	2,888	17	7,932	0.0	6.6	122.55
		ND2	0	3.8	0.0	2,888	5	5,031	0.0	4.8	109.40
		ND3	0	3.8	0.0	2,888	76	33,425	0.0	27.2	445.76
A	3	ARC	0	3.3	0.0	3,106	1,306	18,185	0.0	25.0	540.66
		ND1	0	3.3	0.0	3,106	4,032	17,641	0.0	29.9	1,074.90
		ND2	0	3.3	0.0	3,106	1,795	10,884	0.0	21.4	827.91
		ND3	0	3.3	0.0	3,106	259	5,035	0.0	7.7	321.29
A	4	ARC	0	3.3	0.0	3,105	50	11,084	0.0	18.9	644.51
		ND1	0	3.3	2.3	3,105	539	45,867	0.0	107.1	3,600.00
		ND2	0	3.3	2.0	3,105	1,329	29,574	0.0	81.6	3,600.00
		ND3	0	3.3	0.0	3,105	210	57,647	0.0	104.3	3,166.04
A	5	ARC	0	2.9	2.6	2,797	512	17,704	0.0	38.3	3,600.00
		ND1	0	2.9	2.9	2,797	915	13,100	0.0	40.0	3,600.00
		ND2	0	2.9	2.6	2,797	655	17,155	0.0	63.4	3,600.00
		ND3	0	2.9	2.6	2,797	1,055	23,044	0.0	68.4	3,600.00
A	6	ARC	0	2.4	0.0	3,226	559	1,678	0.0	1.8	154.72
		ND1	0	2.4	0.0	3,226	10,101	40,748	0.0	42.1	1,067.60
		ND2	0	2.4	0.0	3,226	408	30,590	0.0	37.2	1,048.35
		ND3	0	2.4	0.0	3,226	156	2,107	0.0	2.0	94.15
A	7	ARC	0	1.0	0.0	3,314	70	1,605	0.0	2.1	95.25
		ND1	0	1.0	0.0	3,314	3	1,582	0.0	2.2	86.13
		ND2	0	1.0	0.0	3,314	11	551	0.0	0.9	85.20
		ND3	0	1.0	0.0	3,314	50	513	0.0	0.7	78.38
A	8	ARC	0	0.3	0.0	2,894	29	32	0.0	0.0	21.01
		ND1	0	0.3	0.0	2,894	95	163	0.0	0.2	28.76
		ND2	0	0.3	0.0	2,894	41	214	0.0	0.3	33.43
		ND3	0	0.3	0.0	2,894	66	76	0.0	0.1	21.82
A	9	ARC	0	4.4	2.5	2,994	355	35,758	0.0	63.3	3,600.00
		ND1	0	4.4	3.4	2,994	3,034	18,633	0.0	44.6	3,600.00
		ND2	0	4.4	2.8	2,994	843	20,081	0.0	59.6	3,600.00
		ND3	0	4.4	1.8	2,994	504	46,511	0.0	98.3	3,600.00
A	10	ARC	0	9.6	8.4	2,963	6,019	6,834	0.0	15.6	3,600.00
		ND1	0	5.2	3.5	2,828	9,404	20,706	0.0	48.9	3,600.00
		ND2	0	5.2	3.2	2,828	13,871	17,847	0.0	54.0	3,600.00
		ND3	0	5.2	2.5	2,827	4,570	25,136	0.0	51.7	3,600.00
A	11	ARC	0	4.4	0.0	2,410	43	97,456	0.0	100.7	2,102.77
		ND1	0	4.4	3.3	2,410	20,017	63,684	0.0	98.3	3,600.00
		ND2	0	4.4	2.6	2,410	3,173	58,986	0.0	103.3	3,600.00
		ND3	0	4.4	0.0	2,410	507	10,985	0.0	12.2	550.15
A	12	ARC	0	0.3	0.0	2,587	3	4	0.0	0.0	13.69
		ND1	0	0.3	0.0	2,587	16	20	0.0	0.0	23.71
		ND2	0	0.3	0.0	2,587	34	42	0.0	0.1	31.14
		ND3	0	0.3	0.0	2,587	9	14	0.0	0.0	13.01
A	13	ARC	0	7.2	6.6	3,537	◇	935	0.0	3.7	3,599.93
		ND1	0	7.2	7.0	3,537	◇	1,841	0.0	7.2	3,599.79
		ND2	0	2.6	0.0	3,371	701	15,433	0.0	40.9	3,147.43
		ND3	0	2.6	1.0	3,371	210	24,311	0.0	63.4	3,600.00
A	14	ARC	0	2.7	0.0	2,976	89	253	0.0	0.2	35.87
		ND1	0	2.7	0.0	2,976	266	19,137	0.0	23.1	386.69
		ND2	0	2.7	0.0	2,976	531	15,414	0.0	21.2	391.62
		ND3	0	2.7	0.0	2,976	78	2,896	0.0	3.2	131.40
A	15	ARC	0	1.7	0.0	3,133	41	472	0.0	0.7	158.64
		ND1	0	1.7	0.0	3,133	2,521	47,862	0.0	93.2	2,754.71
		ND2	0	1.7	0.0	3,133	37	38,785	0.0	88.5	2,113.49
		ND3	0	1.7	0.0	3,133	21	6,021	0.0	10.8	394.92

◇ No improvement found

Table 29 (continued)

Class	Instance	Setting	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_C	t_B	t
A	16	ARC	0	0.3	0.0	3,445	5	19	0.0	0.0	18.35
		ND1	0	0.3	0.0	3,445	1	4	0.0	0.0	12.73
		ND2	0	0.3	0.0	3,445	26	51	0.0	0.1	28.87
		ND3	0	0.3	0.0	3,445	5	32	0.0	0.0	19.04
A	17	ARC	0	3.1	0.0	2,894	181	75,980	0.0	114.8	2,865.11
		ND1	0	3.1	2.7	2,894	1,632	24,654	0.0	58.7	3,600.00
		ND2	0	3.1	2.2	2,894	545	31,102	0.0	85.6	3,600.00
		ND3	0	3.1	2.8	2,894	35	39,713	0.0	97.8	3,600.00
A	18	ARC	0	7.4	2.7	2,959	13,790	33,918	0.0	53.0	3,600.00
		ND1	0	11.0	10.1	3,076	6,181	11,325	0.0	27.8	3,600.00
		ND2	0	7.4	5.8	2,959	12,082	15,877	0.0	47.6	3,600.00
		ND3	0	7.4	4.5	2,959	2,010	29,900	0.0	62.8	3,600.00
A	19	ARC	0	4.0	2.8	3,356	361	24,210	0.0	37.8	3,600.00
		ND1	0	4.0	2.7	3,356	1,372	27,414	0.0	50.1	3,600.00
		ND2	0	4.0	1.7	3,356	44	26,972	0.0	57.9	3,600.00
		ND3	0	4.0	2.3	3,356	618	25,203	0.0	49.8	3,600.00
A	20	ARC	0	2.8	0.9	2,879	302	117,163	0.0	156.9	3,600.00
		ND1	0	2.8	1.5	2,879	222	87,767	0.0	149.6	3,600.00
		ND2	0	2.8	0.5	2,879	122	164,782	0.0	242.4	3,600.00
		ND3	0	2.8	1.2	2,879	277	98,960	0.0	184.1	3,600.00
B	1	ARC	0	0.0	0.0	3,128	◇	0	0.0	0.0	12.54
		ND1	0	0.0	0.0	3,128	◇	0	0.0	0.0	14.01
		ND2	0	0.0	0.0	3,128	◇	0	0.0	0.0	14.21
		ND3	0	0.0	0.0	3,128	◇	0	0.0	0.0	11.51
B	2	ARC	0	3.3	0.0	2,971	1,100	6,049	0.0	13.5	2,195.11
		ND1	0	3.3	1.7	2,971	1,396	14,993	0.0	41.7	3,600.00
		ND2	0	3.3	0.0	2,971	406	9,205	0.0	21.1	1,948.94
		ND3	0	3.3	0.0	2,971	874	9,480	0.0	22.4	2,118.75
B	3	ARC	0	2.6	1.5	3,251	453	1,923	0.0	7.4	3,599.71
		ND1	0	6.8	6.6	3,396	◇	272	0.0	2.0	3,599.90
		ND2	0	6.8	6.2	3,396	◇	528	0.0	3.8	3,599.83
		ND3	0	2.6	1.6	3,251	503	2,914	0.0	14.7	3,599.80
B	4	ARC	0	3.2	0.0	3,165	469	9,368	0.0	27.9	2,245.31
		ND1	0	3.2	3.0	3,165	682	5,264	0.0	26.8	3,600.00
		ND2	0	3.2	2.4	3,165	337	7,582	0.0	38.2	3,600.00
		ND3	0	3.2	0.8	3,165	286	9,539	0.0	38.6	3,600.00
B	5	ARC	0	1.7	0.0	3,063	3	4,124	0.0	6.8	291.88
		ND1	0	1.7	0.0	3,063	7	3,222	0.0	6.1	388.10
		ND2	0	1.7	0.0	3,063	55	1,449	0.0	2.8	393.47
		ND3	0	1.7	0.0	3,063	16	371	0.0	0.7	187.29
B	6	ARC	0	2.0	0.0	3,557	1	34	0.0	0.0	14.87
		ND1	0	2.0	0.0	3,557	10	527	0.0	0.8	39.00
		ND2	0	2.0	0.0	3,557	37	656	0.0	1.0	45.86
		ND3	0	2.0	0.0	3,557	15	95	0.0	0.2	24.32
B	7	ARC	0	5.0	2.8	2,872	3,362	8,182	0.0	32.5	3,600.00
		ND1	0	9.4	9.3	3,011	1,515	2,015	0.0	13.9	3,599.85
		ND2	0	5.0	4.3	2,872	3,249	4,259	0.0	27.5	3,599.99
		ND3	0	5.0	4.6	2,872	585	3,985	0.0	25.5	3,600.00
B	8	ARC	0	1.2	0.0	3,510	95	227	0.0	0.6	324.46
		ND1	0	1.2	0.0	3,510	28	180	0.0	0.5	196.48
		ND2	0	1.2	0.0	3,510	128	274	0.0	0.9	296.15
		ND3	0	1.2	0.0	3,510	40	191	0.0	0.5	236.26
B	9	ARC	0	3.9	1.6	3,105	265	22,262	0.0	56.5	3,600.00
		ND1	0	4.5	3.9	3,124	1,774	2,716	0.0	12.8	3,599.89
		ND2	0	3.9	2.2	3,105	779	7,520	0.0	29.4	3,600.00
		ND3	0	3.9	1.3	3,105	2,230	11,240	0.0	38.5	3,600.00
B	10	ARC	0	10.9	10.2	3,216	◇	737	0.0	4.3	3,599.69
		ND1	0	10.9	10.8	3,216	◇	785	0.0	4.9	3,599.67
		ND2	0	7.5	7.1	3,099	879	1,859	0.0	11.2	3,599.76
		ND3	0	5.8	5.2	3,042	300	1,460	0.0	7.7	3,599.87

◇ No improvement found

Table 30: DAFP branching scheme results with preprocessing

Class	Instance	Setting	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_C	t_B	t
A	1	ARC+AAN	0	0.8	0.0	2,608	30	33	0.0	0.2	16.58
		ND3+AAN	0	0.8	0.0	2,608	9	11	0.0	0.0	9.72
A	2	ARC+AAN	0	3.8	0.0	2,888	41	162	0.0	0.3	15.78
		ND3+AAN	0	3.8	0.0	2,888	77	28,442	0.0	41.8	440.28
A	3	ARC+AAN	0	3.3	0.0	3,106	50	158	0.0	1.0	51.13
		ND3+AAN	0	3.3	0.0	3,106	155	713	0.0	6.3	145.50
A	4	ARC+AAN	0	3.3	0.0	3,105	1,586	12,094	0.0	136.9	1,244.55
		ND3+AAN	0	3.3	0.0	3,105	237	42,809	0.0	403.0	3,400.77
A	5	ARC+AAN	0	4.2	4.2	2,834	7,898	8,825	0.0	352.7	3,600.00
		ND3+AAN	0	2.9	2.6	2,797	1,996	18,286	0.0	396.4	3,600.00
A	6	ARC+AAN	0	2.4	0.0	3,226	132	828	0.0	2.9	69.24
		ND3+AAN	0	2.4	0.0	3,226	410	1,344	0.0	4.3	110.60
A	7	ARC+AAN	0	1.0	0.0	3,314	522	3,352	0.0	24.6	145.53
		ND3+AAN	0	1.0	0.0	3,314	21	233	0.0	1.5	85.55
A	8	ARC+AAN	0	0.3	0.0	2,894	28	60	0.0	0.4	19.72
		ND3+AAN	0	0.3	0.0	2,894	24	52	0.0	0.3	19.43
A	9	ARC+AAN	0	4.4	2.6	2,994	7,383	24,730	0.0	348.9	3,600.00
		ND3+AAN	0	4.4	1.8	2,994	480	34,150	0.0	298.2	3,600.00
A	10	ARC+AAN	0	5.2	2.9	2,827	7,939	23,206	0.0	263.4	3,600.00
		ND3+AAN	0	5.2	2.1	2,827	4,109	23,084	0.0	179.5	3,600.00
A	11	ARC+AAN	0	4.4	0.0	2,410	86	8,084	0.0	29.4	242.75
		ND3+AAN	0	4.4	0.0	2,410	98	2,818	0.0	12.5	283.14
A	12	ARC+AAN	0	0.3	0.0	2,587	6	12	0.0	0.1	11.94
		ND3+AAN	0	0.3	0.0	2,587	9	14	0.0	0.1	14.69
A	13	ARC+AAN	0	7.2	7.2	3,537	◇	1,594	0.0	81.3	3,599.77
		ND3+AAN	0	2.6	0.0	3,371	80	24,222	0.0	254.9	2,024.71
A	14	ARC+AAN	0	2.7	0.0	2,976	109	277	0.0	1.6	51.72
		ND3+AAN	0	2.7	0.0	2,976	73	903	0.0	3.6	93.22
A	15	ARC+AAN	0	1.7	0.0	3,133	31	64	0.0	1.3	113.76
		ND3+AAN	0	1.7	0.0	3,133	18	3,275	0.0	22.9	289.88
A	16	ARC+AAN	0	0.3	0.0	3,445	7	19	0.0	0.1	20.76
		ND3+AAN	0	0.3	0.0	3,445	5	23	0.0	0.1	21.07
A	17	ARC+AAN	0	3.1	0.0	2,894	56	8,504	0.0	60.4	590.17
		ND3+AAN	0	3.1	2.4	2,894	76	36,795	0.0	361.2	3,600.00
A	18	ARC+AAN	0	7.4	0.0	2,959	3,017	8,942	0.0	96.1	1,578.42
		ND3+AAN	0	7.4	3.7	2,959	4,136	26,102	0.0	285.8	3,600.00
A	19	ARC+AAN	0	4.0	2.8	3,356	1,398	51,166	0.0	390.2	3,600.00
		ND3+AAN	0	4.0	2.4	3,356	913	23,688	0.0	135.0	3,600.00
A	20	ARC+AAN	0	2.8	0.8	2,879	236	114,133	0.0	470.3	3,600.00
		ND3+AAN	0	2.8	1.2	2,879	275	105,815	0.0	411.1	3,600.00
B	1	ARC+AAN	0	0.0	0.0	3,128	◇	0	0.0	0.0	11.46
		ND3+AAN	0	0.0	0.0	3,128	◇	0	0.0	0.0	11.53
B	2	ARC+AAN	0	3.3	0.0	2,971	607	1,861	0.0	45.1	951.08
		ND3+AAN	0	3.3	0.0	2,971	118	1,518	0.0	23.4	545.63
B	3	ARC+AAN	0	2.6	0.0	3,251	773	4,827	0.0	156.8	3,336.83
		ND3+AAN	0	2.6	0.0	3,251	77	6,959	0.0	179.9	2,907.92
B	4	ARC+AAN	0	3.2	2.2	3,165	737	3,590	0.0	295.7	3,599.86
		ND3+AAN	0	3.2	0.0	3,165	233	7,766	0.0	303.3	2,516.30
B	5	ARC+AAN	0	1.7	0.0	3,063	40	63	0.0	0.8	71.51
		ND3+AAN	0	1.7	0.0	3,063	104	191	0.0	2.3	179.25
B	6	ARC+AAN	0	2.0	0.0	3,557	1	29	0.0	0.1	16.66
		ND3+AAN	0	2.0	0.0	3,557	15	75	0.0	0.4	22.02
B	7	ARC+AAN	0	4.8	2.6	2,865	1,555	10,164	0.0	583.6	3,600.00
		ND3+AAN	0	5.8	5.2	2,894	631	4,511	0.0	287.9	3,600.00

◇ No improvement found

Table 30 (continued)

Class	Instance	Setting	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_C	t_B	t
B	8	ARC+AAN	0	1.2	0.0	3,510	202	314	0.0	10.3	349.52
		ND3+AAN	0	1.2	0.0	3,510	41	184	0.0	2.9	172.57
B	9	ARC+AAN	0	3.9	2.5	3,105	3,391	5,994	0.0	366.8	3,599.99
		ND3+AAN	0	3.9	0.0	3,105	1,654	14,306	0.0	357.1	3,198.00
B	10	ARC+AAN	0	10.9	10.9	3,216	◇	1,022	0.0	139.6	3,599.80
		ND3+AAN	0	7.0	6.3	3,083	1,308	1,823	0.0	130.2	3,600.00

◇ No improvement found

Table 31: DAFP cutting plane results

Class	Instance	Setting	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_C	t_B	t
A	1	ALL	26	0.0	0.0	2,608	1	1	0.4	0.0	4.02
		ALL-NDP	26	0.0	0.0	2,608	1	1	0.4	0.0	4.07
		ALL-SPP	39	0.7	0.0	2,608	15	27	0.4	0.1	9.23
		ALL-CTP	18	0.2	0.0	2,608	8	9	0.3	0.1	11.39
		ALL-CLQ	29	0.4	0.0	2,608	3	5	0.4	0.0	9.51
		ALL-LOH	20	0.4	0.0	2,608	1	9	0.4	0.0	7.70
		ALL-LPS	57	0.1	0.0	2,608	17	17	0.4	0.1	11.37
A	2	ALL	123	0.9	0.0	2,888	51	57	0.4	0.3	21.35
		ALL-NDP	127	0.3	0.0	2,888	42	66	0.5	0.2	23.27
		ALL-SPP	96	0.9	0.0	2,888	51	57	0.4	0.3	22.55
		ALL-CTP	135	0.3	0.0	2,888	15	20	0.4	0.1	11.19
		ALL-CLQ	156	0.3	0.0	2,888	17	17	0.5	0.1	16.67
		ALL-LOH	153	0.3	0.0	2,888	1	3	0.5	0.0	6.68
		ALL-LPS	174	0.2	0.0	2,888	26	27	0.4	0.1	22.57
A	3	ALL	427	0.0	0.0	3,106	0	0	3.4	0.0	15.05
		ALL-NDP	290	0.0	0.0	3,106	0	0	3.0	0.0	12.22
		ALL-SPP	248	0.0	0.0	3,106	0	0	3.0	0.0	13.45
		ALL-CTP	305	0.0	0.0	3,106	0	0	3.0	0.0	13.98
		ALL-CLQ	354	0.0	0.0	3,106	1	1	3.5	0.0	15.03
		ALL-LOH	255	0.0	0.0	3,106	0	0	3.0	0.0	11.64
		ALL-LPS	355	0.0	0.0	3,106	0	0	2.5	0.0	11.95
A	4	ALL	296	1.1	0.0	3,105	59	862	6.7	11.4	202.81
		ALL-NDP	274	1.1	0.0	3,105	225	1,443	7.0	18.7	447.04
		ALL-SPP	376	1.1	0.0	3,105	1	617	7.5	6.7	159.01
		ALL-CTP	214	1.1	0.0	3,105	123	4,319	6.1	53.5	564.53
		ALL-CLQ	379	1.1	0.0	3,105	18	4,606	7.0	53.5	463.31
		ALL-LOH	237	1.0	0.0	3,105	251	2,865	6.9	36.7	406.70
		ALL-LPS	327	1.2	0.0	3,105	244	1,163	5.6	14.7	330.16
A	5	ALL	837	1.0	0.0	2,797	134	14,643	9.1	185.5	2,933.80
		ALL-NDP	567	1.2	1.0	2,797	106	8,447	9.8	123.8	3,600.00
		ALL-SPP	605	1.2	0.8	2,797	299	19,005	7.9	238.3	3,600.00
		ALL-CTP	688	1.2	0.9	2,797	328	29,363	8.0	364.3	3,600.00
		ALL-CLQ	771	1.2	0.0	2,797	128	10,528	7.7	126.2	1,804.02
		ALL-LOH	746	1.0	0.8	2,797	80	22,612	9.0	271.6	3,600.00
		ALL-LPS	843	1.8	1.6	2,797	243	6,904	5.4	121.8	3,600.00
A	6	ALL	329	0.7	0.0	3,226	167	508	1.1	1.7	64.67
		ALL-NDP	211	0.7	0.0	3,226	59	207	1.1	0.8	59.10
		ALL-SPP	287	0.7	0.0	3,226	167	508	1.1	1.6	72.21
		ALL-CTP	288	0.7	0.0	3,226	44	155	0.9	0.9	57.48
		ALL-CLQ	343	0.7	0.0	3,226	164	267	1.0	1.0	63.16
		ALL-LOH	268	0.7	0.0	3,226	152	874	1.1	2.9	86.73
		ALL-LPS	368	0.7	0.0	3,226	77	183	0.8	0.8	51.08
A	7	ALL	308	0.8	0.0	3,314	53	229	2.8	2.2	57.95
		ALL-NDP	202	0.7	0.0	3,314	13	50	3.0	0.8	50.72
		ALL-SPP	249	0.8	0.0	3,314	50	170	2.8	1.6	69.83
		ALL-CTP	228	0.8	0.0	3,314	86	637	2.7	5.5	61.73
		ALL-CLQ	287	0.8	0.0	3,314	31	193	2.8	2.2	64.29
		ALL-LOH	285	0.8	0.0	3,314	32	85	2.9	0.8	31.33
		ALL-LPS	271	0.8	0.0	3,314	59	883	2.3	6.7	84.78
A	8	ALL	191	0.0	0.0	2,894	0	0	1.6	0.0	6.28
		ALL-NDP	157	0.0	0.0	2,894	0	0	1.6	0.0	7.08
		ALL-SPP	157	0.0	0.0	2,894	0	0	1.6	0.0	6.77
		ALL-CTP	155	0.0	0.0	2,894	1	1	1.5	0.0	6.87
		ALL-CLQ	173	0.0	0.0	2,894	0	0	1.6	0.0	6.46
		ALL-LOH	106	0.0	0.0	2,894	0	0	1.6	0.0	6.54
		ALL-LPS	133	0.0	0.0	2,894	0	0	1.3	0.0	5.77
A	9	ALL	547	1.1	0.0	2,994	17	309	4.4	2.6	133.27
		ALL-NDP	429	1.1	0.0	2,994	1	61	4.0	0.6	94.45
		ALL-SPP	442	1.1	0.0	2,994	81	172	4.4	1.8	169.22
		ALL-CTP	538	1.1	0.0	2,994	1	85	3.5	0.7	74.93
		ALL-CLQ	605	1.1	0.0	2,994	12	67	4.3	0.7	133.85
		ALL-LOH	525	1.1	0.0	2,994	13	231	4.3	1.9	166.69
		ALL-LPS	592	1.2	0.0	2,994	50	2,408	3.1	19.0	425.53

◇ No improvement found

Table 31 (continued)

Class	Instance	Setting	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_C	t_B	t
A	10	ALL	617	1.5	0.0	2,827	1,199	66,009	2.7	320.6	3,053.77
		ALL-NDP	456	1.6	0.0	2,827	615	49,765	2.4	252.5	2,947.65
		ALL-SPP	505	1.4	0.0	2,827	460	54,212	2.6	252.0	2,363.91
		ALL-CTP	557	1.5	0.4	2,827	1,685	60,667	2.3	330.6	3,600.00
		ALL-CLQ	538	1.8	0.0	2,827	944	38,244	2.4	197.0	2,601.55
		ALL-LOH	418	1.5	1.1	2,827	8,609	17,430	2.7	134.9	3,600.00
		ALL-LPS	569	2.2	0.0	2,827	496	45,893	1.9	239.0	2,835.27
A	11	ALL	517	0.0	0.0	2,410	0	0	2.0	0.0	19.82
		ALL-NDP	405	0.0	0.0	2,410	1	1	2.4	0.0	32.05
		ALL-SPP	427	0.0	0.0	2,410	0	0	1.9	0.0	21.68
		ALL-CTP	426	0.0	0.0	2,410	0	0	1.8	0.0	16.63
		ALL-CLQ	552	0.0	0.0	2,410	0	0	1.9	0.0	16.86
		ALL-LOH	524	0.0	0.0	2,410	0	0	2.0	0.0	19.69
		ALL-LPS	504	1.8	0.0	2,410	123	206	1.2	1.1	79.40
A	12	ALL	214	0.1	0.0	2,587	1	16	0.7	0.0	10.46
		ALL-NDP	131	0.1	0.0	2,587	1	6	0.8	0.0	10.22
		ALL-SPP	179	0.1	0.0	2,587	1	16	0.7	0.0	11.26
		ALL-CTP	163	0.1	0.0	2,587	1	10	0.6	0.0	8.60
		ALL-CLQ	187	0.2	0.0	2,587	3	10	0.6	0.0	16.98
		ALL-LOH	178	0.1	0.0	2,587	27	63	0.7	0.3	17.91
		ALL-LPS	229	0.1	0.0	2,587	1	23	0.6	0.1	11.80
A	13	ALL	527	0.1	0.0	3,371	66	224	6.8	3.9	225.90
		ALL-NDP	383	0.1	0.0	3,371	63	1,409	6.8	18.8	519.29
		ALL-SPP	425	0.1	0.0	3,371	27	344	6.7	4.9	299.49
		ALL-CTP	432	0.1	0.0	3,371	86	197	6.4	6.4	551.67
		ALL-CLQ	532	0.1	0.0	3,371	17	50	6.8	1.0	141.59
		ALL-LOH	421	0.1	0.0	3,371	13	228	6.8	3.1	238.11
		ALL-LPS	832	0.2	0.0	3,371	144	2,771	6.0	39.3	1,292.42
A	14	ALL	282	0.4	0.0	2,976	18	200	1.2	0.7	28.71
		ALL-NDP	215	0.4	0.0	2,976	85	86	1.3	0.6	34.53
		ALL-SPP	219	0.4	0.0	2,976	18	200	1.2	0.7	31.03
		ALL-CTP	251	0.2	0.0	2,976	12	12	1.1	0.2	26.54
		ALL-CLQ	255	0.0	0.0	2,976	0	0	1.2	0.0	7.27
		ALL-LOH	257	0.0	0.0	2,976	0	0	1.2	0.0	8.21
		ALL-LPS	247	0.0	0.0	2,976	4	4	1.0	0.0	14.46
A	15	ALL	608	0.0	0.0	3,133	2	2	5.5	0.1	39.68
		ALL-NDP	413	0.2	0.0	3,133	11	17	4.9	0.4	96.15
		ALL-SPP	486	0.0	0.0	3,133	2	2	5.4	0.1	43.26
		ALL-CTP	548	0.0	0.0	3,133	0	0	4.7	0.0	28.70
		ALL-CLQ	628	0.0	0.0	3,133	1	1	5.1	0.1	42.71
		ALL-LOH	608	0.0	0.0	3,133	0	0	5.0	0.0	34.53
		ALL-LPS	511	0.1	0.0	3,133	6	6	4.1	0.2	86.13
A	16	ALL	141	0.3	0.0	3,445	3	5	1.3	0.1	14.51
		ALL-NDP	101	0.3	0.0	3,445	3	5	1.3	0.1	19.79
		ALL-SPP	118	0.3	0.0	3,445	8	10	1.3	0.1	20.45
		ALL-CTP	109	0.3	0.0	3,445	4	4	1.2	0.0	15.72
		ALL-CLQ	168	0.4	0.0	3,445	42	43	1.3	0.3	20.65
		ALL-LOH	214	0.3	0.0	3,445	5	5	1.5	0.0	19.89
		ALL-LPS	116	0.3	0.0	3,445	7	7	1.1	0.1	17.63
A	17	ALL	492	1.9	0.0	2,894	71	128	3.4	1.5	91.09
		ALL-NDP	354	1.9	0.0	2,894	914	1,306	3.4	14.6	276.78
		ALL-SPP	391	1.9	0.0	2,894	71	128	3.4	1.5	100.45
		ALL-CTP	357	1.9	0.0	2,894	22	43	3.1	1.0	105.59
		ALL-CLQ	444	1.9	0.0	2,894	15	54	3.4	0.7	72.21
		ALL-LOH	359	1.9	0.0	2,894	1	78	3.4	0.6	59.05
		ALL-LPS	418	2.1	0.0	2,894	33	35,602	2.7	320.0	1,947.20
A	18	ALL	876	2.7	0.0	2,959	219	11,115	4.5	91.8	1,441.18
		ALL-NDP	623	2.6	0.0	2,959	3,722	7,572	4.6	70.5	1,942.32
		ALL-SPP	826	2.7	0.0	2,959	1,773	10,013	4.4	100.6	2,969.78
		ALL-CTP	799	2.6	0.0	2,959	123	5,361	4.3	44.7	974.62
		ALL-CLQ	901	2.6	0.0	2,959	57	3,668	4.6	31.4	828.97
		ALL-LOH	721	2.6	0.0	2,959	158	10,759	4.7	91.4	1,577.63
		ALL-LPS	838	3.4	0.0	2,959	79	3,014	3.0	25.7	707.35

◇ No improvement found

Table 31 (continued)

Class	Instance	Setting	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_C	t_B	t
A	19	ALL	466	2.2	0.0	3,356	209	13,264	2.0	65.0	1,286.64
		ALL-NDP	373	2.2	0.0	3,356	71	21,321	2.0	108.4	1,889.60
		ALL-SPP	343	2.2	1.1	3,356	161	60,242	1.9	312.5	3,600.00
		ALL-CTP	405	2.8	2.2	3,356	1,583	3,157	1.8	35.0	3,600.00
		ALL-CLQ	506	2.2	0.0	3,356	20	28,539	1.9	131.4	2,160.15
		ALL-LOH	417	2.2	0.0	3,356	182	15,063	1.9	72.2	1,492.38
		ALL-LPS	465	3.3	2.7	3,356	122	15,542	1.4	98.5	3,600.00
A	20	ALL	278	1.8	0.5	2,879	58	156,188	1.3	525.0	3,600.00
		ALL-NDP	202	1.8	0.5	2,879	58	146,856	1.3	491.9	3,600.00
		ALL-SPP	202	1.8	0.5	2,879	58	152,504	1.3	507.8	3,600.00
		ALL-CTP	241	1.8	0.4	2,879	201	195,734	1.0	617.8	3,600.00
		ALL-CLQ	306	1.8	0.5	2,879	2,585	142,248	1.2	483.1	3,600.00
		ALL-LOH	261	1.8	0.5	2,879	24	201,181	1.3	652.3	3,600.00
		ALL-LPS	231	2.3	0.3	2,879	501	173,173	0.9	544.4	3,600.00
B	1	ALL	69	0.0	0.0	3,128	◇	0	6.3	0.0	15.61
		ALL-NDP	51	0.0	0.0	3,128	◇	0	6.3	0.0	15.54
		ALL-SPP	51	0.0	0.0	3,128	◇	0	6.3	0.0	15.53
		ALL-CTP	51	0.0	0.0	3,128	◇	0	6.3	0.0	16.26
		ALL-CLQ	66	0.0	0.0	3,128	◇	0	6.4	0.0	15.78
		ALL-LOH	63	0.0	0.0	3,128	◇	0	6.3	0.0	15.37
		ALL-LPS	60	0.0	0.0	3,128	◇	0	6.0	0.0	15.42
B	2	ALL	724	0.0	0.0	2,971	0	0	8.9	0.0	39.04
		ALL-NDP	473	0.0	0.0	2,971	0	0	8.9	0.0	39.77
		ALL-SPP	600	0.0	0.0	2,971	0	0	8.9	0.0	38.84
		ALL-CTP	628	0.0	0.0	2,971	0	0	8.6	0.0	38.96
		ALL-CLQ	775	0.0	0.0	2,971	0	0	8.9	0.0	37.31
		ALL-LOH	727	0.0	0.0	2,971	0	0	8.9	0.0	40.97
		ALL-LPS	586	0.0	0.0	2,971	0	0	7.2	0.0	36.23
B	3	ALL	760	0.9	0.0	3,251	36	894	28.1	29.3	579.80
		ALL-NDP	486	0.9	0.0	3,251	3	153	28.7	4.8	268.43
		ALL-SPP	677	0.9	0.0	3,251	36	894	28.0	29.3	578.86
		ALL-CTP	717	0.9	0.0	3,251	6	60	29.0	2.8	254.95
		ALL-CLQ	727	0.9	0.0	3,251	13	89	27.7	4.3	346.34
		ALL-LOH	666	0.9	0.0	3,251	2	72	27.3	2.3	193.79
		ALL-LPS	662	1.1	0.0	3,251	231	3,320	23.5	129.6	3,086.03
B	4	ALL	552	1.9	0.0	3,165	120	2,096	41.0	104.5	994.99
		ALL-NDP	413	1.8	0.0	3,165	41	4,830	42.1	195.4	1,546.92
		ALL-SPP	433	2.0	0.0	3,165	71	1,751	42.2	68.7	634.36
		ALL-CTP	491	1.9	0.0	3,165	114	4,555	38.4	207.1	1,535.61
		ALL-CLQ	556	1.9	0.0	3,165	133	3,483	42.7	139.6	1,086.51
		ALL-LOH	436	1.9	0.0	3,165	540	7,784	38.5	333.5	2,146.28
		ALL-LPS	560	1.9	0.0	3,165	5	2,332	32.2	90.0	662.78
B	5	ALL	168	0.0	0.0	3,063	1	1	3.1	0.1	20.73
		ALL-NDP	134	0.0	0.0	3,063	2	2	3.3	0.1	25.51
		ALL-SPP	144	0.0	0.0	3,063	1	1	3.1	0.1	20.68
		ALL-CTP	153	0.0	0.0	3,063	0	0	2.9	0.0	17.23
		ALL-CLQ	144	0.0	0.0	3,063	0	0	2.9	0.0	17.80
		ALL-LOH	141	0.0	0.0	3,063	0	0	2.9	0.0	16.53
		ALL-LPS	168	0.8	0.0	3,063	14	16	2.8	0.2	39.44
B	6	ALL	83	0.3	0.0	3,557	1	2	1.2	0.0	11.91
		ALL-NDP	67	0.3	0.0	3,557	1	2	1.2	0.0	11.84
		ALL-SPP	67	0.3	0.0	3,557	1	2	1.2	0.0	11.90
		ALL-CTP	67	0.3	0.0	3,557	7	7	1.0	0.1	22.61
		ALL-CLQ	83	0.4	0.0	3,557	13	13	1.2	0.1	18.18
		ALL-LOH	68	0.4	0.0	3,557	7	7	1.2	0.1	22.32
		ALL-LPS	72	0.3	0.0	3,557	9	9	1.0	0.1	13.85
B	7	ALL	1,122	0.9	0.0	2,865	336	4,573	29.7	154.8	1,201.74
		ALL-NDP	753	0.8	0.0	2,865	73	4,906	31.2	167.1	1,204.32
		ALL-SPP	827	0.9	0.0	2,865	28	5,515	30.9	181.1	1,283.59
		ALL-CTP	833	0.9	0.0	2,865	119	5,336	28.7	177.2	1,096.94
		ALL-CLQ	1,001	0.9	0.0	2,865	314	5,237	28.7	167.0	1,042.34
		ALL-LOH	898	0.8	0.0	2,865	363	7,028	30.6	224.8	1,168.32
		ALL-LPS	1,204	8.3	8.2	3,059	◇	1,090	24.3	218.7	3,600.00

◇ No improvement found

Table 31 (continued)

Class	Instance	Setting	Cuts	Root Gap%	Final Gap%	z_{IP}	z_{IP} Node	B&B Nodes	t_C	t_B	t
B	8	ALL	526	0.4	0.0	3,510	76	77	7.5	3.5	251.74
		ALL-NDP	522	0.3	0.0	3,510	17	18	8.7	0.9	154.22
		ALL-SPP	372	0.5	0.0	3,510	5	10	7.6	0.4	79.52
		ALL-CTP	460	0.4	0.0	3,510	7	8	7.2	0.5	94.87
		ALL-CLQ	435	0.4	0.0	3,510	3	4	7.7	0.2	68.32
		ALL-LOH	386	0.4	0.0	3,510	49	54	7.4	2.2	238.29
		ALL-LPS	454	0.5	0.0	3,510	92	95	6.4	2.3	156.07
B	9	ALL	794	1.4	0.0	3,105	52	2,213	22.4	55.1	631.63
		ALL-NDP	615	1.5	0.0	3,105	193	1,466	21.9	34.2	448.61
		ALL-SPP	620	1.5	0.0	3,105	531	1,499	21.4	39.0	439.63
		ALL-CTP	659	1.5	0.0	3,105	32	1,864	21.1	45.0	444.47
		ALL-CLQ	796	1.4	0.0	3,105	17	3,137	22.3	77.6	720.85
		ALL-LOH	577	1.4	0.0	3,105	219	1,298	21.8	33.9	413.74
		ALL-LPS	818	1.9	0.0	3,105	238	5,679	17.4	155.7	1,361.70
B	10	ALL	535	1.3	0.4	3,042	366	10,367	21.5	275.5	3,600.00
		ALL-NDP	441	1.3	0.0	3,042	41	6,603	21.6	173.8	2,027.75
		ALL-SPP	461	1.3	0.4	3,042	366	10,431	21.5	277.5	3,600.00
		ALL-CTP	499	1.3	0.0	3,042	184	11,405	21.0	299.2	3,056.02
		ALL-CLQ	540	1.3	0.6	3,042	188	11,850	21.5	323.1	3,600.00
		ALL-LOH	501	1.3	0.0	3,042	101	2,578	21.4	70.6	1,355.08
		ALL-LPS	616	1.7	0.3	3,042	21	11,435	18.9	313.3	3,600.00

◇ No improvement found

REFERENCES

[1] Ahuja, R. K., Magnanti, T. L., and Orlin, J. B., *Network Flows*. New Jersey: Prentice-Hall, 1993.

[2] Aneja, Y. P., Aggarwal, V., and Nair, K. P. K., "Shortest chain subject to side constraints," *Networks*, vol. 13, pp. 295–302, 1983.

[3] Ascheuer, N., Fischetti, M., and Grötschel, M., "A polyhedral study of the asymmetric traveling salesman problem with time windows," *Networks*, vol. 36, no. 2, pp. 69–79, 2000.

[4] Ascheuer, N., Fischetti, M., and Grötschel, M., "Solving the asymmetric travelling salesman problem with time windows by branch-and-cut," *Mathematical Programming*, vol. 90A, pp. 475–506, 2001.

[5] Atamturk, A., Nemhauser, G. L., and Savelsbergh, M. W. P., "Conflict graphs in solving integer programming problems," *European Journal of Operational Research*, vol. 121, pp. 40–55, 2000.

[6] Avella, P., Boccia, M., and Sforza, A., "A penalty function heuristic for the resource constrained shortest path problem," *European Journal of Operational Research*, vol. 142, no. 2, pp. 221–230, 2002.

[7] Avella, P., Boccia, M., and Sforza, A., "Resource constrained shortest path problems in path planning for fleet management," *Journal of Mathematical Modelling and Algorithms*, vol. 3, pp. 1–17, 2004.

[8] Barnhart, C., Boland, N., Clarke, L., Johnson, E. L., Nemhauser, G. L., and Shenoi, R. G., "Flight string models for aircraft fleeting and routing," *Transportation Science*, vol. 32, pp. 208–220, 1998.

[9] Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., and Vance, P. H., "Branch-and-price: Column generation for solving huge integer programs," Tech. Rep. COC-94-03, Computational Optimization Center, Georgia Institute of Technology, 1994 (revised January 1996).

[10] Bauer, P., Linderoth, J., and Savelsbergh, M. W. P., "A branch and cut approach to the cardinality constrained circuit problem," *Mathematical Programming*, vol. 91, pp. 307–348, 2002.

[11] Beasley, J. and Christofides, N., "An algorithm fo the resource constrained shortest path problem," *Networks*, vol. 19, pp. 379–394, 1989.

[12] Bellman, R., "On a routing problem," *Quarterly of Applied Mathematics*, vol. 16, pp. 87–90, 1958.

[13] Bixby, R. E. and Lee, E. K., "Solving a truck dispatching scheduling problem using branch-and-cut," *Operations Research*, vol. 46, no. 3, pp. 355–367, 1998.

[14] Boland, N., Dethridge, J., and Dumitrescu, I., "Accelerated label setting algorithms for the elementary resource constrained shortest path problem," *Operations Research Letters*, vol. 34, pp. 58–68, 2006.

[15] Boyd, E. A., "Polyhedral results for the precedence constrained knapsack problem," *Discrete Applied Mathematics*, vol. 41, pp. 185–201, 1993.

[16] Chabrier, A., "Vehicle routing problem with elementary shortest path based column generation," tech. rep., ILOG, Madrid, 2002.

[17] Cherkassky, B. V., Goldberg, A. V., and Radzik, T., "Shortest paths algorithms: Theory and experimental evaluation," *Mathematical Programming*, vol. 73, pp. 129–174, 1996.

[18] Christofides, N., Mingozzi, A., and Toth, P., "Exact algorithms for the vehicle routing problem based on spanning tree and shortest path relaxations," *Mathematical Programming*, vol. 20, pp. 255–282, 1981.

[19] Cook, S. A., "The complexity of theorem-proving procedures," in *Proceedings of the Third annual ACM Symposium on Theory of Computing* (Harrison, M. A., Banerji, R. B., and Ullman, J. D., eds.), pp. 151–158, Association for Computing Machinery, New York, 1971.

[20] Cordeau, J.-F. and Laporte, G., "The dial-a-ride problem (DARP): Variants, modeling issues and algorithms," *4OR—Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, vol. 1, pp. 89–101, 2003.

[21] Cormen, T., Leiserson, C., and Rivest, R., *Introduction to Algorithms*. MIT Press, 1993.

[22] Crowder, H., Johnson, E. L., and Padberg, M., "Solving large-scale zero-one linear programming problems," *Operations Research*, vol. 31, no. 5, pp. 803–834, 1983.

[23] Dahl, G. and Realfsen, B., "The cardinality-constrained shortest path problem in 2-graphs," *Networks*, vol. 36, no. 1, pp. 1–8, 2000.

[24] Danna, E., Rothberg, E., and Pape, C. L., "Exploring relaxation induced neighborhoods to improve mip solutions," *Mathematical Programming*, vol. Ser. A 102, pp. 71–90, 2005.

[25] Dantzig, G. B., "Maximization of a linear function of variables subject to linear inequalities," in *Activity Analysis of Production and Allocation* (Koopmans, T. C., ed.), pp. 339–347, John Wiley & Sons, New York, 1951.

[26] Dantzig, G. B., Fulkerson, D. R., and Johnson, S. M., "Solution of a large scale traveling salesman problem," *Operations Research*, vol. 2, pp. 393–410, 1954.

[27] Dash Optimization, *XPRESS-MP User's Guide, Release 13*, 2001.

[28] Desrochers, M., Desrosiers, J., and Solomon, M., "A new optimization algorithm for the vehicle routing problem with time windows," *Operations Research*, vol. 40, pp. 342–354, 1992.

[29] Desrochers, M. and Soumis, F., "A generalized permanent labelling algorithm for the shortest path problem with time windows," *INFOR*, vol. 26, pp. 191–212, 1988.

[30] Desrosiers, J., Dumas, Y., Solomon, M., and Soumis, F., "Time constrained routing and scheduling," in *Handbooks in Operations Research and Management Science: Network Routing* (Ball, M. O., Magnanti, T. L., Monma, C. L., and Nemhauser, G. L., eds.), vol. 8, pp. 35–139, North-Holland, Amsterdam, 1995.

[31] Desrosiers, J., Pelletier, P., and Soumis, F., "Plus court chemin avec constraints d'horaires," *RAIRO*, vol. 17, pp. 357–377, 1983.

[32] Desrosiers, J., Soumis, F., and Desrochers, M., "Routing with time windows by column generation," *Networks*, vol. 14, pp. 545–565, 1984.

[33] Dijkstra, E., "A note on two problems in connexion with graphs," *Numeriche Mathematics*, vol. 1, pp. 269–271, 1959.

[34] Dumitrescu, I., *Constrained Path and Cycle Problems*. PhD thesis, University of Melbourne, Victoria, Australia, 2002.

[35] Dumitrescu, I. and Boland, N., "Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem," *Networks*, vol. 42, no. 3, pp. 135–153, 2003.

[36] Elimam, A. and Kohler, D., "Two engineering applications of a constrained shortest path model," *European Journal of Operational Research*, vol. 103, pp. 426–438, 1997.

[37] Eppstein, D., "Finding the k shortest paths," *SIAM Journal on Computing*, vol. 28, pp. 652–673, 1999.

[38] Eppstein, D., September 2007. `http://www.ics.uci.edu/~eppstein/bibs`.

[39] Espinoza, D., Garcia, R., Goycoolea, M., Nemhauser, G. L., and Savelsbergh, M. W. P., "Per-seat, on-demand air transportation part I: Problem description and an integer multicommodity flow model," *Transportation Science*, vol. 42, no. 3, pp. 263–278, 2008.

[40] Espinoza, D., Garcia, R., Goycoolea, M., Nemhauser, G. L., and Savelsbergh, M. W. P., "Per-seat, on-demand air transportation part II: Parallel local search," *Transportation Science*, vol. 42, no. 3, pp. 292–301, 2008.

[41] Feillet, D., Dejax, P., Gendreau, M., and Gueguen, C., "An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems," *Networks*, vol. 142, pp. 216–229, 2004.

[42] Ferreira, C. E., Martin, A., and Weismantel, R., "Solving multiple knapsack problems by cutting planes," *SIAM Journal on Optimization*, vol. 6, pp. 858–877, 1996.

[43] Fischetti, M. and Lodi, A., "Local branching," *Mathematical Programming*, vol. Ser. B 98, pp. 23–47, 2003.

[44] Floyd, R. W., "Algorithm 97: Shortest path," *Communications of ACM*, vol. 5, p. 345, 1962.

[45] Ford, L. R., "Network flow theory," Tech. Rep. P-923, Rand Corporation, Santa Monica, California, 1956.

[46] Ford, L. R. and Fulkerson, D. R., "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, pp. 399–404, 1956.

[47] Fredman, M. L. and Tarjan, R. E., "Fibonacci heaps and their uses in improved network optimization algorithms," in *Proceedings of the 25th Annual IEEE Symposium on Foundations of Computer Science*, pp. 338–346, 1984.

[48] Gamache, M., Soumis, F., and Marquis, G., "A column generation approach for large-scale aircrew rostering problems," *Operations Research*, vol. 47, no. 2, pp. 247–263, 1999.

[49] Garey, M. R. and Johnson, D. S., *Computers and Intractibility, a Guide to the Theory of NP-Completeness*. New York: W.H. Freeman and Company, 1979.

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

[51] Gilmore, P. C. and Gomory, R. E., "A linear programming approach to the cutting stock problem: Part II," *Operations Research*, vol. 11, pp. 863–888, 1961.

[52] Goldberg, A. V., "A new max-flow algorithm," Tech. Rep. MIT/LCS/TM-291, Laboratory for Computer Science, MIT, Cambridge, MA, 1985.

[53] Gomory, R. E., "Outline of an algorithm for integer solutions to linear programs," *Bulletin of the American Mathematical Society*, vol. 64, pp. 275–278, 1958.

[54] Gomory, R. E., "Some polyhedra related to combinatorial problems," *Linear Algebra and Its Applications*, vol. 2, pp. 451–558, 1969.

[55] Graves, G. W., McBride, R. D., Gershkoff, I., Anderson, D., and Mahidhara, D., "Flight crew scheduling," *Management Science*, vol. 39, pp. 657–682, 1993.

[56] Grötschel, M., Jünger, M., and Reinelt, G., "A cutting plane algorithm for the linear ordering problem," *Operations Research*, vol. 32, pp. 1195–1220, 1984.

[57] Gu, Z., *Lifted Cover Inequalities for 0-1 and Mixed 0-1 Integer Programs*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, 1995.

[58] Gu, Z., Nemhauser, G. L., and Savelsbergh, M. W. P., "Lifted cover inequalities for 0-1 integer programs I: Computation," Tech. Rep. LEC-94-09, Georgia Institute of Technology, Atlanta, 1994.

[59] Gu, Z., Nemhauser, G. L., and Savelsbergh, M. W. P., "Sequence independent lifting," *Journal of Combinatorial Optimization*, vol. 4, pp. 109–129, 2000.

[60] Halpern, J. and Priess, J., "Shortest paths with time constraints on moving and parking," *Networks*, vol. 4, pp. 241–253, 1974.

[61] Handler, G. Y. and Zang, I., "A dual algorithm for the constrained shortest path problem," *Networks*, vol. 10, pp. 293–309, 1980.

[62] Hansen, P., "Bicriterion path problems," in *Multiple Criteria Decision Making Theory and Application: Lecture Notes in Economics and Mathematical Systems 177* (Fandel, G. and Gal, T., eds.), pp. 109–127, Springer-Verlag, Berlin, 1980.

[63] Hassin, R., "Approximation schemes for the restricted shortest path problem," *Mathematics of Operations Research*, vol. 17, no. 1, pp. 36–42, 1992.

[64] Held, M. and Karp, R. M., "The travelling salesman problem and minimal spanning trees," *Operations Research*, vol. 18, pp. 1138–1162, 1970.

[65] Hoffman, K. L. and Padberg, M., "Improving lp-representations of zero-one linear programs for branch-and-cut," *ORSA Journal on Computing*, vol. 3, no. 2, pp. 121–134, 1991.

[66] Hoffman, K. L. and Padberg, M., "Solving airline crew scheduling problems by branch-and-cut," *Management Science*, vol. 39, no. 6, pp. 657–682, 1993.

[67] Hoffman, W. and Pavley, R., "A method for the solution of the nth best path problem," *Journal of the Association for Computing Machinery*, vol. 6, no. 4, pp. 506–514, 1959.

[68] Houck, D. J., Picard, J. C., Queyranne, M., and Vemuganti, R. R., "The travelling salesman problem as a constrained shortest path problem: Theory and computational experience," *Operations Research*, vol. 17, pp. 93–109, 1980.

[69] ILOG Inc., *ILOG CPLEX 9.0 User's Manual*, October 2003.

[70] Irnich, S. and Desaulniers, G., "Shortest path problems with resource constraints," Tech. Rep. G-2004-11, Les Cahiers du GERAD, Montréal, Québec, Canada, 2004.

[71] Irnich, S. and Villeneuve, D., "The shortest-path problem with resource constraints and k-cycle elimination for $k \geq 3$," Tech. Rep. G-2003-55, Les Cahiers du GERAD, Montréal, Québec, Canada, 2003.

[72] Jaumard, B., Semet, F., and Vovor, T., "A two-phase resource constrained shortest path algorithm for acyclic graphs," Tech. Rep. G-96-48, Les Cahiers du GERAD, 1996.

[73] Jimenez, V. and Marzal, A., "Computing the k shortest paths. a new algorithm and an experimental comparison," in *3rd Workshop on Algorithm Engineering (WAE99), LNCS 1668*, pp. 15–29, Springer, 1999.

[74] Joksch, H. C., "The shortest route problem with constraints," *Journal of Mathematical Analysis and Application*, vol. 14, pp. 191–197, 1966.

[75] Kallehauge, B. and Boland, N., "Path inequalities for the vehicle routing problem with time windows," Tech. Rep. 87-91-1371-60, Centre for Traffic and Transport, Technical University of Denmark, 2005.

[76] Karp, R. M., "Reducibility among combinatorial problems," in *Complexity of Computer Computations* (Miller, R. E. and Thatcher, J. W., eds.), pp. 85–103, Plenum Press, 1972.

[77] Karzanov, A. V., "Determining the maximal flow in a network by the method of preflows," *Soviet Mathematics Doklady*, vol. 15, pp. 434–437, 1974.

[78] Katoh, N., Ibaraki, T., and Mine, H., "An efficient algorithm for k shortest simple paths," *Networks*, vol. 12, pp. 411–427, 1982.

[79] Kelley, J. E., "The cutting plane method for solving convex programs," *Journal of the SIAM*, vol. 8, pp. 703–712, 1960.

[80] Khachiyan, L. G., "A polynomial algorithm in linear programming," *Soviet Mathematics Doklady*, vol. 20, pp. 191–194, 1979.

[81] Klabjan, D., Nemhauser, G. L., and Tovey, C., "The complexity of cover inequality separation," *Operations Research Letters*, vol. 23, pp. 35–40, 1998.

[82] Klose, A., "An lp-based heuristic for two-stage capacitated facility location problems," *Journal of the Operational Research Society*, vol. 50, pp. 157–166, 1999.

[83] Kohl, N., *Exact Methods for Time Constrained Routing and Related Scheduling Problems*. PhD thesis, Institute of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 1995.

[84] Kolen, A. W. J., Rinnooy-Kan, A. H. G., and Trienekens, H. W. J. M., "Vehicle routing with time windows," *Operations Research*, vol. 35, pp. 266–274, 1987.

[85] Land, A. and Powell, S., "Computer codes for problems in integer programming," in *Discrete Optimization II* (Hammer, P. L., Johnson, E. L., and Korte, B. H., eds.), pp. 221–269, North Holland Publishing Co., 1979.

[86] Land, A. H. and Doig, A. G., "An automatic method of solving discrete programming problems," *Econometrica*, vol. 28, pp. 497–520, 1960.

[87] Lavoie, S., Minoux, M., and Odier, E., "A new approach for crew pairing problems by column generation with an application to air transportation," *European Journal of Operational Research*, vol. 35, pp. 45–58, 1988.

[88] Lawler, E., *Combinatorial Optimization: Networks and Matroids*. New York: Holt, Rhinehart and Winston, 1976.

[89] Linderoth, J. T. and Savelsbergh, M. W. P., "A computational study of branch and bound search strategies for mixed integer programming," *INFORMS Journal on Computing*, vol. 11, pp. 173–187, 1999.

[90] Lorenz, D. H. and Raz, D., "A simple efficient approximation scheme for the restricted shortest path problem," *Operations Research Letters*, vol. 28, no. 5, pp. 213–219, 2001.

[91] Martins, E. Q. V. and Pascoal, M. M. B., "A new implementation of yen's ranking loopless paths algorithm," *4OR - Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, vol. 1, no. 2, pp. 121–134, 2003.

[92] Mehlhorn, K. and Ziegelmann, M., "Resource constrained shortest paths," in *7th Annual European Symposium on Algorithms (ESA2000), LNCS 1879*, pp. 326–337, 2000.

[93] Nemhauser, G. L. and Sigismondi, G., "A strong cutting plane/branch-and-bound algorithm for node packing," *Journal of the Operational Research Society*, vol. 43, no. 5, pp. 443–457, 1992.

[94] Nemhauser, G. L. and Vance, P. H., "Lifting cover facets of the 0-1 knapsack polytope with gub constraints," *Operations Research Letters*, vol. 16, pp. 255–263, 1994.

[95] Nemhauser, G. L. and Wolsey, L. A., *Integer and Combinatorial Optimization*. New York: Wiley, 1988.

[96] Nygaard, R., *Shortest Path Methods in Representation and Compression of Signals and Image Contours*. PhD thesis, Norwegian University of Science and Technology, 2000.

[97] Padberg, M. W. and Rinaldi, G., "A branch and cut algorithm for resolution of large scale symmetric traveling salesman problems," *SIAM Review*, vol. 33, pp. 60–100, 1991.

[98] Pallottino, S. and Scutellà, M. G., "Shortest path algorithms in transportation models: Classical and innovative aspects," in *Equilibrium and Advanced Transportation Modelling* (Marcotte, P. and Nguyen, S., eds.), pp. 245–281, Kluwer, 1998.

[99] Park, K. and Park, S., "Lifting cover inequalities for the precedence constrained knapsack problem," *Discrete Applied Mathematics*, vol. 72, pp. 219–241, 1997.

[100] Phillips, C., "The network inhibition problem," in *25th ACM Symposium on Theory of Computing (STOC)*, pp. 776–785, 1993.

[101] Ribeiro, C. and Minoux, M., "A heuristic approach to hard constrained shortest path problems," *Discrete Applied Mathematics*, vol. 10, pp. 125–137, 1985.

[102] Savelsbergh, M. W. P., "Preprocessing and probing techniques for mixed integer programming problems," *ORSA Journal on Computing*, vol. 6, no. 4, pp. 445–455, 1994.

[103] Schrijver, A., *Theory of Linear and Integer Programming*. Wiley, 1986.

[104] Shiloach, Y. and Vishkin, U., "An $\mathcal{O}(n^2 \log n)$ parallel max-flow algorithm," *Journal of Algorithms*, vol. 3, pp. 128–146, 1982.

[105] Soumis, F., "Decomposition and column generation," Tech. Rep. G-97-42, GERAD Research Report, 1997.

[106] Spoorendonk, S., Jepsen, M., and Petersen, B., "A branch-and-cut algorithm for the elementary shortest path problem with resource constraints," tech. rep., DIKU, University of Copenhagen, Denmark, 2005.

[107] van de Leensel, R. L. M. J., van Hoesel, C. P. M., and van de Klundert, J. J., "Lifting valid inequalities for the precedence constrained knapsack problem," *Mathematical Programming*, vol. 86, pp. 161–185, 1999.

[108] Vance, P., Barnhart, C., Johnson, E. L., and Nemhauser, G. L., "Airline crew scheduling: A new formulation and decomposition algorithm," *Operations Research*, vol. 45, no. 2, pp. 188–200, 1997.

[109] Warburton, A., "Approximation of pareto-optima in multiple-objective shortest path problems," *Operations Research*, vol. 35, no. 1, pp. 70–79, 1987.

[110] Wolsey, L. A., "Facets and strong valid inequalities for integer programs," *Operations Research*, vol. 24, pp. 367–372, 1976.

[111] Wolsey, L. A., "Valid inequalities for mixed integer programs with generalised and variable upper bound constraints," *Discrete Applied Mathematics*, vol. 25, pp. 251–261, 1990.

[112] Wolsey, L. A., *Integer Programming*. New York: Wiley, 1998.

[113] Xue, G., "Primal-dual algorithms for computing weight-constrained shortest paths and weight-constrained minimum spanning trees," in *19th IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pp. 271–277, 2000.

[114] Yen, J. Y., "Finding the k shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[115] Zabarankin, M., Uryasev, S., and Pardalos, P., "Optimal risk path algorithms," in *Cooperative Control and Optimization* (Murphey, R. and Pardalos, P., eds.), pp. 271–303, Kluwer, 2001.

[116] Zemel, E., "Easily computable facets of the knapsack polytope," *Mathematics of Operations Research*, vol. 14, no. 4, pp. 760–765, 1989.