TWO APPLICATIONS OF COMBINATORIAL OPTIMIZATION

BY MATTHEW R. OSTER

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Operations Research

Written under the direction of

Professor Jonathan Eckstein

and approved by

_____

_____

_____

_____

_____

New Brunswick, New Jersey

May, 2014

ABSTRACT OF THE DISSERTATION

Two applications of combinatorial optimization

by Matthew R. Oster

Dissertation Director: Professor Jonathan Eckstein

This thesis presents two applications of combinatorial optimization. The first part contains a detailed description of a conference scheduling problem. We model the problem as a symmetric clustering problem, or a variant of **minimum k-partition** we call **capacitated k-partition**. This problem is proved to be $\mathcal{NP}$-hard to solve to optimality, and further, unless $\mathcal{P} = \mathcal{NP}$, no constant factor polynomial-time approximation algorithm exists. We also propose a branch-and-cut algorithm with semidefinite programming relaxations enhanced with polyhedral cuts found at each tree node. Many cutting planes are demonstrated to be satisfied, or provably close to being satisfied, by semidefinite matrices in the variable space $[-1/(k-1), 1]$, which is in contrast to basic linear programming relaxations. Our algorithm also relies on a novel heuristic strategy when attempting to generate feasible solutions at every tree node. We test an implementation of our algorithm on random k-partition instances as well as a particular conference data set which comes from the 13th Annual INFORMS Computing Society Conference and was solved to within 0.85% of optimum in under 4 hours. The results here are promising and provide a starting point for future projects.

In the second part, we describe a project called the Boat Allocation Module, where a team comprised of United States Coast Guard (USCG) analysts, and Command, Control, and Interoperability Center for Advanced Data Analysis researchers worked

together in building a decision support system for USCG analysts. The software was designed to solve the problem of allocating boats of the Coast Guard to the nation's coastal stations, so as to meet station requirements, while adhering to particular budget and capability limitations. We model the problem as a resource allocation problem and prove that it is $\mathcal{NP}$-hard to solve. We relax the problem slightly by allowing a single boat type to be shared, or assigned among disjoint subsets of stations rather than to individual stations, but show that implementing "seasonal" sharing is $\mathcal{NP}$-hard. A mixed integer linear programming formulation is proposed, and an implementation within a decision support system for USCG analysts is tested as per USCG Verification and Validation standards. The software provides an intuitive interface and allows for a variety of scenarios. Tests have shown that our tool may save the Coast Guard millions of dollars a year.

Acknowledgements

I would like to thank the members of my dissertation committee. I am especially indebted to my advisor, Professor Jonathan Eckstein, for initiating and guiding the research found in Part I, and for outlining the first reduction found in Appendix B. This work as a whole would not have been possible without his patience, support, and expertise. I would also like thank Professor Endre Boros for his mentorship during the project found in Part II, and for his overall energy brought to every discussion. I want to thank Professor Vladimir Gurvich for introducing me early on to many interesting research problems. I want to also thank Professors Farid Alizadeh and Miguel Anjos for their influential works in Semidefinite Programming, a critical tool used in Part I. I need to thank Professor Adi Ben-Israel for joining the committee, especially on such short notice.

Throughout the past six years as a Ph.D. student, I have encountered many inspiring individuals at Rutgers University. I need to thank Professor Fred Roberts for mentoring and believing in me, as well as for turning my interests towards all things Discrete Mathematics. Professor Paul Kantor has helped shape me as a researcher and was a key figure in guiding the successful research found in Part II. I would also like to thank Dr. Gene Fiorini for his support and his amiable demeanor. I want to thank Professor András Prékopa for his passionate teaching and his encouraging comments on a particular class project. Dr. Tami Carpenter has been a great source of support and continues to keep me on the path to success as an interdisciplinary researcher. I would like to thank Robert DeMarco and Jake Baron for providing a starting point for the second reduction found in Appendix B. Others include the RUTCOR staff: Terry Hart, Clare Smietana, Lynn Agre, Katie D'Agosta, and Chong; I have enjoyed and benefited from all of our conversations.

I could not have enjoyed the last six years of my studies as much as I had if not for the tremendous unconditional support of my wife and best friend Lauren Matricardi. I also want to acknowledge the supportive comradery felt among my fellow graduate students. In particular, I want to thank my good friend, office mate, and eXtremely knowlEdgeable individual, Aritanan Gruber. The light-hearted attitude of Minh Pham has always brought about a good laugh. Ozlem Cavus demonstrated an air of unmatched dedication and kindness. I also want to thank office mates Noam Goldberg, Chris Gaffney, Gianluca Gazolla, and Anh Ninh, as well as fellow class mates Dávid Papp, Kunikazu Yoda, Wang Yao, Mohammad Ranjbar, Olga Myndyuk, and Jinwook Lee for interesting discussions and many adventures.

This material is based upon work supported by the Command, Control, and Interoperability Center for Advanced Data Analysis (CCICADA) at Rutgers University, and the U.S. Department of Homeland Security, under Grant Award Numbers 2008-ST-104-000016 and 2009-ST-061-CCI002-05. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

Dedication

To my family, for their endless support.

Table of Contents

List of Tables

List of Figures

Part I

Optimizing conference schedules

Chapter 1

Introduction

The structure of a general conference schedule can be viewed as having sessions of grouped talks, time slots over many days, and various rooms spread across multiple nearby venues. The goal is typically to find a schedule, or an assignment of sessions to distinct times and places, which has no participant required to be in two places at once. Naturally, many more constraining factors arise, *e.g.* speakers preferring or forbidding to be scheduled at certain times, subsets of rooms unavailable at the end of a conference, or advisor-advisee pairs each requesting to be at the other's presentation.

Not all constraints, however, are achievable when constructing a schedule, and thus those deemed inessential are relaxed. For example, consider the INFORMS Annual Meeting 2012 scheduling problem, with over 1,000 sessions of technical presentations, 15 time slots, and nearly 70 rooms available [35]. While it was observed that no speaker was to give two talks at once, at least one conflict may have been avoidable: the two authors of this thesis gave two separate talks in parallel, one of which was on this very collaboration.

On the other hand, some properties arise automatically from the restrictions imposed upon the schedule structure. For instance, the tightly constrained 13th INFORMS Computing Society Conference (ICS-2013) exposed a "balanced" problem structure: there were 61 technical sessions to be scheduled over the course of 9 time slots, with at most 7 sessions allowed per time slot, and thus every time slot was forced to contain at least 5 sessions [36]. Our model for the conference scheduling problem is flexible enough to allow the inclusion of constraints similar to the above two types, *i.e.* it can produce balanced schedules without having participants assigned to be in two places at once.

A basic multi-track structure exists in the aforementioned conferences (the 2012 International Symposium on Mathematical Programming [91] is another example), where talks similar in topic are grouped and placed one after another in the same physical area of the conference. While a sequential ordering allows audience members—whom we suppose have concentrated interests—to potentially stay in close proximity throughout the duration of the conference, it may only incidentally produce schedules which reduce total *parallel topic overlap*, or a measure of similarity between sessions appearing together within some time slot.[1] We, however, are interested in guaranteeing that such parallel topic overlap is minimal. By doing so, we complementarily increase the variety and overall appeal of talks offered in each time slot.

In the sequel, we give a general definition of the conference scheduling problem, but primarily focus on a symmetric sub-problem arising from attempting to minimize total parallel topic overlap. Indeed, similarity between presented topics is independent of when and where such talks are given, and thus allows the elimination of time slot and room labels. Therefore, we are able to concentrate on the "core" structure of the scheduling problem: clustering sessions into time slots. Although this approach may limit the model's ability to satisfy certain secondary constraints (*e.g.* individuals' time preferences), it significantly reduces the size and complexity of the problem. Moreover, the symmetric problem can be modeled as a capacitated variant of the **minimum** k-**partition** problem—or the well-studied complementary **maximum** k-**cut** problem—since the set of clusters forms a partition of the sessions. We borrow from the structure of these known models, as well as other related combinatorial problems, and so we provide a detailed review of relevant literature, next.

1.1 Related work

There are many types of scheduling problems that have been successfully modeled and solved by combinatorial optimization methods, such as job shop scheduling [7] or flight

[1] Note that at quick glance of the ICS-2013 schedule [36], swapping the Critical Infrastructure session in the DeVargas room held during Sunday's B-slot with the "Other" session appearing in Sunset Room held on Sunday's C-slot, could potentially result in a reduced amount of parallel topic overlap, judging only by track titles and assuming that no conflicts would be introduced.

and airline crew scheduling [43]. Outside of the optimization community, there exist general methodologies such as timetabling for handling *e.g.* education-based scheduling problems (see surveys [89] and [84]). We give a review of results pertaining to several specific combinatorial problems which are related to our conference scheduling problem.

The **maximum cut** problem asks for a 2-partition of a given weighted graph G with a corresponding *cut*—the set of edges with endpoints in opposite parts of the partition—of maximum total weight. It is a combinatorial problem that has received significant attention over the last two decades, both theoretical [12, 28] and practical [13, 83]. The groundbreaking result of Goemans and Williamson [42] was the first use of semidefinite programming (SDP)[2] to achieve a (randomized) approximation guarantee for a combinatorial optimization problem. The algorithm, hereby denoted GW, is guaranteed on average to produce α^{GW}-approximations, *i.e.* solutions within a factor α^{GW} of optimality, for $\alpha^{\mathrm{GW}} \coloneqq 0.87856$. Karloff [60] showed that any additional valid inequalities included in the SDP relaxation of Goemans and Williamson could not improve upon this factor. Håstad [52] proved that unless $\mathcal{P} = \mathcal{NP}$, the best one can hope for in designing any α-approximation algorithm for **maximum cut**, is at most $\frac{16}{17} \approx 0.94118$. This upper limit can be reduced, under the assumption that the Unique Games Conjecture holds [62], to α^{GW}, showing that GW conditionally achieves the best quality guarantee in polynomial-time computable approximations.

The randomized rounding approach of Goemans and Williamson [42] was extended to the more general **maximum k-cut** problem in Frieze and Jerrum [38]. For a weighted graph G and integer $k \geq 2$, the problem seeks a maximum-weight k-*cut*, or set of edges with endpoints in separate parts of a corresponding k-partition. The α_k-approximation guarantee for **maximum k-cut**—which arises from a non-standard formulation with variables $Y \in \left\{\frac{-1}{k-1}, 1\right\}^{n \times n}$, which is discussed further in section 2.2)—can be asymptotically written as $\alpha_k \sim 1 - \frac{1}{k} + \frac{\log k}{2k^2}$. Kann *et al.* [59] showed that each α_k has limited room for improvement by proving that no α-approximation algorithm with factor larger

[2]Semidefinite programming is a sub-class of convex optimization problems and a generalization of linear programming; the first SDP-specific polynomial-time algorithm was developed by Alizadeh [4] (see also [44]).

than $1 - \frac{1}{34k}$ can exist, unless $\mathcal{P} = \mathcal{NP}$. They also proved that, contrary to **maximum k-cut**, the complementary **minimum k-partition** problem cannot be approximated to within a factor of $|E(G)|$—the number of edges of the corresponding graph—unless $\mathcal{P} = \mathcal{NP}$.

Related hardness results for **maximum k-cut** exist. For instance, Guruswami and Sinop [50] proved the problem to be $\mathcal{NP}$-hard even when the input graph G is known to be k-colorable—a problem sometimes called **maximum k-colorable subgraph**. For smaller values of k, de Klerk *et al.* [26] marginally improved upon the α_k-approximation ratio of Frieze and Jerrum [38], by using a similar randomized rounding procedure, only applied to an SDP relaxation of a problem related to the Lovász theta function [72] (see also the notes of Knuth [63]).

Despite the hardness results, many authors have successfully implemented exact algorithms for the above problems. For **maximum cut**, the *Bi*nary *q*uadratic and *Ma*x *c*ut solver, or *BiqMac* (available from [1]), can handle up to 100 nodes even when the input graph is dense, by using a branch-and-bound algorithm equipped with an SDP relaxation, which is further tightened by polyhedral cuts [86]. A more recent solver for **maximum cut** instances is *BiqCrunch* [64]. On the other hand, linear programming (LP) methods reportedly performed better on sparse instances and graphs with special structure (*e.g.* grids) arising from statistical physics application, *e.g.* the Ising spin glass problem in Barahona *et al.* [11].

For **maximum k-cut**—and therefore for the complementary problem **minimum k-partition**—with $k \geq 2$, Ghaddar *et al.* [41] developed a branch-and-cut algorithm using SDP relaxations and an iterative clustering heuristic (ICH), and were able to solve dense instances containing up to 60 nodes, and over 100 nodes for sparse grid-like graphs related to the Potts glass problem [71]. The runtimes were further improved by Anjos *et al.* [5], using a bundle method to solve the SDP relaxations. For small k and dense instances, the bundle method of Anjos *et al.* compared favorably against the LP-based branch-and-bound method of Kaibel *et al.* [58], which employed orbitopal fixing—a symmetry-breaking branching method based on the algebraic orbits of feasible solutions. Without symmetry breaking, it was observed in Ghaddar *et al.* [41] and Anjos

et al. [5] that the LP relaxations were very weak compared to the corresponding SDP relaxations, and resulted in almost complete enumeration in the search for optimal solutions.

Heuristic techniques without approximation guarantees have also been developed for **maximum cut**, such as the Rank Two heuristic of Burer *et al.* [19]. The methods of Festa *et al.* [34], which combine greedy randomized adaptive search procedure (GRASP), variable neighborhood search (VNS), and path-relinking (PR), have been shown to be practical improvements—both in solution quality and runtime performance—over the GW algorithm found in [42]. For the more general **maximum k-cut** problem, the ICH heuristic of [41] was seen to be competitive with the provably good randomized rounding procedure of [38].

The structure of the *cut polytope*, or the convex hull of all incidence vectors encoding feasible cuts of the **maximum cut** problem, and the structure of the *elliptope*, a convex object arising from a particular SDP relaxation of partition matrices, have both been extensively studied in the book of Deza and Laurent [28]. The polyhedral structure of the complementary, and more general problem, **minimum k-partition**, can be found in Chopra and Rao [20, 21]. Eisenblätter [31] discusses the strength of a related, truncated elliptope containing all r-partition matrices with $r \leq k$. These structures are described more in the forthcoming chapter (in particular, see §2.3).

The **minimum graph bisection** problem asks for a graph's maximum-weight cut with upper-bounded part sizes of the corresponding 2-partition. Polyhedral results for this problem can be found in Armbruster *et al.* [9], whereas computations comparing LP-based *vs.* SDP-based branch-and-cut algorithms are found in their follow-up paper [8]. Brunetta *et al.* [18] describe an algorithm for **minimum equicut**: a subclass of **minimum graph bisection** asking for a graph's minimum-weight cut with "balanced" parts—2-partition classes of equal size when the number of vertices is even, and differing by one otherwise. The algorithm was later improved by Anjos *et al.* [6].

The **minimum k-equipartition** problem, which can be seen as a generalized variant of **minimum equicut**, asks for a k-partition with "balanced" parts and corresponding k-cut of minimum weight. A polyhedral study and LP-based branch-and-cut

algorithm were analyzed in Mitchell [75], and an application of this algorithm to sports scheduling is found in Mitchell [74].

Labbé and Öszoy [65] analyzed the structure of the **constrained graph partition** problem's polytope; this problem asks for a partition of a graph's vertex set into an arbitary number of parts, each bounded above and below in size, so that the corresponding multi-cut is of optimal weight (maximum or minimum can be specified). This definition is general enough to include: **simple graph partition**, which asks for a graph partition with upper-bounded part sizes (*cf.* [92]); **clique partition**, which asks for a graph partition with no constraint on part sizes (*cf.* [45, 46]); **size constrained clique partition**, which asks for a graph partition with lower-bounded part sizes (*cf.* [57]); and the aforementioned k-**equipartition** and **equipartition** problems studied, respectively, by Mitchell [75] and Conforti *et al.* [22, 23].

Ferreira *et al.* [32, 33] studied a generalization to **simple graph partition**, known as **node capacitated graph partition**, where vertices have arbitrary weights and upper bounds are placed on the sum of each part's constituent weights. Some constraints developed in this work are covered in section §2.3.4, since node weights arise in certain subproblems solved by our algorithm (see Chapter 3).

There has been research on obtaining successively better relaxations of binary combinatorial problems. The well-known hierarchies of both linear and semidefinite relaxations were proposed by Sherali-Adams [90], Lovasz-Schrijver [73], and Lasserre [68]. A comparison of the three can be found in Laurent [69], where it was determined that the Lasserre hierarchy is strongest.

1.2 Our contributions

In this thesis, we study the problem of scheduling conferences, and are motivated by the seemingly *ad hoc* approaches apparent in example schedules. We first introduce a general definition of the **conference scheduling problem** (ConSP), but focus on a symmetric sub-problem, the **conference clustering problem** (ConCP), which arises when time slot and room labels are ignored. This particular structure was chosen to

accomodate a novel comparison measure of schedules; namely, parallel topic overlap. Several ways of computing this measure are proposed.

We prove that ConCP, and hence ConSP, is not only an $\mathcal{NP}$-hard combinatorial optimization problem, but that it is $\mathcal{NP}$-complete to determine whether a feasible solution exists. This, we prove, implies that no α-approximation algorithm exists, for any constant α.

After modeling ConCP as a variant of the familiar **maximum k-cut** problem, we explore the structure of our particular problem's polytope. We determine which closely-related polytopes provide valid inqualities for our polytope, and consider when these inequalities are facet-defining. On the other hand, we demonstrate the strength of a semidefinite programming formulation of the problem when compared to the polytope in $\{0,1\}$ space. In particular, we display known results showing when the SDP relaxations are provably "close" to satisfying the many valid inequality types, and use them to prove when in fact such inequalities are satisfied by the SDP relaxation. All cuts are displayed in both the natural $\{0,1\}$ variables space as well as the "shifted" space $\left\{\frac{-1}{k-1}, 1\right\}$.

We design a branch-and-bound algorithm for solving the computationally difficult clustering problem ConCP. We argue for our choice of SDP-based relaxations as opposed to LP methods. We also find evidence for our branching choice in a related algorithm solving the restricted subclass **minimum k-partition**. Since feasible solutions are $\mathcal{NP}$-hard to generate, we develop a bin-packing-based heuristic that works well in practice—it has been observed to find solutions often, and those found tend to be relatively close to optimal.

We test our algorithm on random data as well as on the ICS-2013 data set (see beginning of this chapter). The random data instances were solvable in reasonable amounts of time for instances with up to 60 nodes, and otherwise the SDP relaxations were strong at the root. Our case study showed particularly promising results. The algorithm provided a feasible conference schedule which was within 1% of the optimum, within a 4-hour period, and reduced the actual implemented schedule's total parallel similarity by more than 20%.

1.3 Outline

The outline of this part of the dissertation is as follows. In Chapter 2 we present a general definition of the **conference scheduling problem** (§2.1), as well as the symmetric subproblem, and main focus: the **conference clustering problem** (§2.1.1). This chapter also explores the computational complexity of ConSP (§2.1.2), showing that the problem is theoretically as "hard" to optimize as to generate feasible solutions. A particular model of ConCP based on the **maximum k-cut** formulation that yields a strong SDP relaxation is presented in section §2.2. Polyhedral structures and borrowed valid inequalities from problems related to ConCP appear in section §2.3.

In Chapter 3, we propose a branch-and-bound algorithm intended to solve instances of ConCP to optimality. The main points discussed are the structure of the tree nodes (§3.1.1), the model of a contracted subproblem (§3.1.2), a method of preprocessing subproblems (§3.2), the lower bounding procedure (§3.3), the upper bounding procedure (§3.4), and the algorithm's branching rule (§3.5). The lower bounding procedure section includes subsections discussing a subproblem's semidefinite relaxation (§3.3.1), as well as the accompanying cutting plane subroutine (§3.3.2). The upper bounding procedure includes two heuristic strategies for generating feasible solutions; namely a bin-packing and a bin-swapping subroutine, which are presented, respectively, in sections §3.4.1 and §3.4.2.

Chapter 4 presents our computational results. First we discuss our choice of software and implementation issues (§4.1). Then, testing on random data appears in section §4.2. We also include the tuning of our parameters and runs on the ICS-2013 data set (§4.3). Concluding remarks are recorded in the last section (§4.4). An accompanying appendix provides methods for constructing the similarity measures and other inputs to ConSP (§A).

In the sequel, we assume the reader is familiar with basic terminology of integer programming [76], semidefinite programming [102], graph theory [101], complexity theory [81], and the analysis of algorithms [70]. Definitions will be clarified as needed.

Chapter 2

The conference scheduling problem

This chapter defines and explores the properties of the optimization problem we call the **conference scheduling problem**, which can intuitively be described as that of finding an assignment of small groups of presentations to times and places, so that the total similarity between talks appearing at the same time is minimal.

2.1 Problem description

The input $\mathcal{I}$ to the **conference scheduling problem** is referred to as a *conference*, and contains the following: n *sessions*, or sets containing pre-assigned talks which are to be presented at distinct times and places; k *time slots*, or blocks of time reserved for the presentation of sessions; and m *rooms*, in which sessions are to appear. There are also integers α and β, which represent the lower and upper bounds, respectively, on the allowed number of sessions in any time slot. Further, a conference $\mathcal{I}$ contains a subset S consisting of all distinct pairs of sessions which cannot appear together in any identical time slot. For any pair $\{i, j\} \in S$, i and j are said to be in *conflict*. We refer to sessions, time slots, and rooms by index, *e.g.* the set of n sessions is represented by $[n] \coloneqq \{1, \ldots, n\}$. The parameters must satisfy $2 \leq k \leq n \leq km$, and $1 \leq \alpha \leq \frac{n}{k} \leq \beta \leq \min\{m, n\}$, for otherwise the problem is not meaningful or interesting (*e.g.* $k = 1$), is infeasible (*e.g.* $n > km$), or can be redefined to fit these inequalities (*e.g.* if $\beta > \min\{m, n\}$, we can reset $\beta \coloneqq \min\{m, n\}$, since no time slot can receive more sessions than this amount).

A *schedule* of a conference $\mathcal{I}$ is defined as an assignment of each session $i \in [n]$ to a distinct time-and-place pair $(t, r) \in [k] \times [m]$. More formally, a schedule is an *injective* map $f : [n] \to [k] \times [m]$, *i.e.* $f(i) = f(j)$ implies $i = j$ for each i, $j \in [n]$. A schedule

f is said to be *feasible* for $\mathcal{I}$, if (1) f assigns at least α and at most β sessions to each time slot $t \in [k]$, and (2) f does not assign any conflicting pair of S to the same time slot. We rewrite these constraints, respectively, as:

$$\alpha \leq |g_f(t)| \leq \beta \qquad \text{for all } t \in [k], \tag{2.1}$$

$$\binom{g_f(t)}{2} \cap S = \emptyset \qquad \text{for all } t \in [k], \tag{2.2}$$

where $g_f(t) \coloneqq \bigcup_{r \in [m]} f^{-1}(t, r)$ for each $t \in [k]$, and $\binom{X}{r}$ denotes the family of r-element subsets of X. Note that since $\alpha \geq 1$, a feasible schedule does not contain empty time slots.

Consider a schedule f of $\mathcal{I}$, and a pair of sessions $\{i, j\} \in \binom{[n]}{2}$. If we have $\{i, j\} \subseteq \binom{g_f(t)}{2}$, for some $t \in [k]$, then the sessions i and j are *grouped* by f or *together*, and are otherwise said to be *separated* by f. If a pair $\{i, j\} \in S$ is grouped by f, then f is said to have a *session conflict*.

The *cost* of a schedule f, $c(f)$, is defined to be the total similarity in topics appearing in parallel sessions. To calculate $c(f)$, we suppose a symmetric matrix $W \in \mathbb{R}^{n \times n}$ is given, where the value W_{ij} represents the *similarity* or overlapping topics between sessions i and j relative to that of all other pairs, for each distinct pair $i, j \in [n]$ (see Appendix §A for a method of constructing such measures). In other words, $W_{ij} < W_{h\ell}$ signifies that sessions i and j have less overlap in subject matter than do sessions h and ℓ. We normalize such values so that $0 \leq W_{ij} \leq 1$, and define the diagonal entries of W to be 0, *i.e.* $W_{ii} \coloneqq 0$. Thus the cost of any schedule is formally written as:

$$c(f) \coloneqq \sum_{t \in [k]} \sum_{\{i,j\} \in \binom{g_f(t)}{2}} W_{ij}. \tag{2.3}$$

The **conference scheduling problem** calls for a feasible schedule f with minimal $c(f)$. Letting $\mathcal{F}(\mathcal{I})$ be the set of schedules satisfying constraints (2.1)–(2.2), the problem can be restated in terms of solving the following optimization problem for a given conference $\mathcal{I} \coloneqq (n, k, m, \alpha, \beta, S, W)$ as described above:

$$\begin{aligned} \text{(ConSP)} \quad & \text{minimize} \quad c(f) \\ & \text{subject to} \quad f \in \mathcal{F}(\mathcal{I}). \end{aligned} \tag{2.4}$$

A solution f to (2.4) is said to be an *optimal schedule*. Next we show that there is much symmetry in the definition of a schedule, which allows a certain reformulation of the problem.

2.1.1 Symmetry

By definition of the set $\mathcal{F}(\mathcal{I})$ for any conference $\mathcal{I}$, and corresponding cost function c (see (2.3) and (2.4)), the labels $t \in [k]$ and $r \in [m]$ do not influence the cost of an optimal solution. More precisely, if we define the family $\mathcal{C}(f) \coloneqq \{g_f(t) \,|\, t \in [k]\}$ as the *session clustering* of f, then the following symmetric structure can be observed. First, for any positive integers x and y, $y \leq x$, note that the value $x^{\underline{y}} \coloneqq x \cdot (x-1) \cdots (x-y+1)$ counts the number of injective maps from a set of size y to a set of size x.

Proposition 2.1.1. *For any conference $\mathcal{I}$ and schedule f of $\mathcal{I}$, the total number of schedules f' satisfying $\mathcal{C}(f') = \mathcal{C}(f)$ is:*

$$k! \cdot \Pi_{P \in \mathcal{C}(f)} \, m^{\underline{|P|}} = \Theta\left(e^{k \log k} \cdot \Pi_{P \in \mathcal{C}(f)} \, m^{\underline{|P|}}\right). \tag{2.5}$$

Moreover, each such schedule is of equal cost, i.e. $c(f') = c(f)$ whenever $\mathcal{C}(f') = \mathcal{C}(f)$.

Proof. First notice that, by definition, any schedule f' is such that the session clustering $\mathcal{C}(f')$ is a partition of $[n]$ into k non-empty parts, each of size at most β. Further, since it is required that $\beta \leq \min\{m, n\}$, the part sizes of $\mathcal{C}(f')$ are at most the number of rooms available, m. Thus counting the number of schedules f' for which $\mathcal{C}(f') = \mathcal{C}(f)$, amounts to counting the number of distinct ways of assigning the k non-empty parts $P \in \mathcal{C}(f)$ to distinct time slots $t \in [k]$, and assigning the $|P|$ constituents of any cluster P to distinct rooms $r \in [m]$, *i.e.* count the number of bijections on k elements, and the number of injections from $|P|$ to m elements, for each $P \in \mathcal{C}(f)$. Since these assignments can be performed independently, the first statement is immediate.

The right-hand side of the first claimed equality follows from the well-known property

$$\log(x!\,) = \Theta(x \log x)$$

which holds for any positive integer x (see for instance [70]).

The final claim is trivial by definition of cost function c (*cf.* (2.3)) and the supposition $\mathcal{C}(f') = \mathcal{C}(f)$. □

Since the above result holds for all schedules, and thus those which are feasible, the number of optimal schedules of a given conference $\mathcal{I}$ is exponential in the input. However, the proof of Proposition 2.1.1 suggests an approach to finding optimal schedules which bypasses the symmetries of schedules: namely, by focusing on session clusterings. To this end, define a cost function $\hat{c}(\mathcal{C}) \coloneqq c(f)$ for any schedule f and session clustering $\mathcal{C}$ of a conference $\mathcal{I}$ with $\mathcal{C}(f) = \mathcal{C}$—this is well-defined by the above proposition. Further, denote a session clustering $\mathcal{C}$ of $\mathcal{I}$ as *feasible* if it satisfies the following constraints, which can be seen as analogous to a feasible schedule's defining constraints (2.1) and (2.2), respectively:

$$\alpha \leq |F| \leq \beta \qquad \text{for all } F \in \mathcal{C} \tag{2.6}$$

$$\binom{F}{2} \cap S = \emptyset \qquad \text{for all } F \in \mathcal{C}. \tag{2.7}$$

Now, letting $\hat{\mathcal{F}}(\mathcal{I})$ denote the set of all feasible session clusterings of conference $\mathcal{I}$, *i.e.* $\hat{\mathcal{F}}(\mathcal{I}) \coloneqq \left\{\mathcal{C} \in 2^{[n]} \,\middle|\, \exists f \in \mathcal{F}(\mathcal{I}) : \mathcal{C}(f) = \mathcal{C}\right\}$, where 2^X denotes the power set of a set X, the **conference clustering problem** is defined as the following optimization problem:

$$\begin{aligned} \text{(ConCP)} \quad & \text{minimize} && \hat{c}(\mathcal{C}) \\ & \text{subject to} && \mathcal{C} \in \hat{\mathcal{F}}(\mathcal{I}). \end{aligned} \tag{2.8}$$

Notice that any optimal session clustering of (2.8) can be converted to an optimal solution of ConSP (*cf.* (2.4)) in polynomial time, with respect to the same input conference $\mathcal{I}$. Indeed, by definition, every schedule f has a session clustering $\mathcal{C}(f)$, and every session clustering $\mathcal{C}$ maps to some schedule f. Furthermore, by Proposition 2.1.1, a session clustering $\mathcal{C}$ and all schedules f for which $\mathcal{C} = \mathcal{C}(f)$ are of equal cost, *i.e.* $\hat{c}(\mathcal{C}) = c(f)$. Thus any labeling of the time slots and rooms for an optimal session clustering $\mathcal{C}$ suffices. In other words, given optimal solution $\mathcal{C}$ to ConCP, an arbitrary bijection $\sigma : \mathcal{C} \to [k]$ and any injections $\gamma_F : F \to [n]$, for each $F \in \mathcal{C}$, are enough to form an optimal schedule to ConSP, in linear time.

It is clear now that the number-of-rooms parameter m of an input schedule for the **conference scheduling problem** is irrelevant for the purpose of solving ConCP, and so, with slight abuse of notation, the input tuple for the **conference clustering problem** will be a schedule $\mathcal{I}$ with m suppressed, *i.e.* $\mathcal{I} \coloneqq (n, k, \alpha, \beta, S, W)$. Next, we prove that in fact ConCP is $\mathcal{NP}$-hard to solve to optimality, and therefore so is ConSP.

2.1.2 Computational complexity

For the following complexity results, we restate the **conference clustering problem** as a constrained version of a known graph problem: **minimum k-partition**. Here the input is a tuple (n, k, W) where n and k are integers satisfying $2 \leq k \leq n$, and W is an $n \times n$ non-negative, symmetric matrix. The problem asks to find a *k-partition* $(V_1, \ldots, V_k)$ of vertices $V \coloneqq [n]$—a partition of V into k non-empty, pairwise disjoint subsets V_i called *classes* or *parts*—of minimum total cost or *weight* $\sum_{h\in[k]} \sum_{\{i,j\}\in\binom{V_h}{2}} W_{ij}$. Indeed, the session clusterings of a conference $\mathcal{I} \coloneqq (n,\ k,\ \alpha,\ \beta,\ S,\ W)$ can be viewed as a particularly structured k-partition of $[n]$, and for any session clustering $\mathcal{C} = (V_1, \ldots, V_k)$ of $[n]$, the total weight is written $\hat{c}(\mathcal{C}) = \sum_{F\in\mathcal{C}} \sum_{\{i,j\}\in\binom{F}{2}} W_{ij} = \sum_{h\in[k]} \sum_{\{i,j\}\in\binom{V_h}{2}} W_{ij}$. This capacitated k-partition problem is formally defined as follows.

Definition 2.1.2. ***capacitated* k-*partition***: *given input $(n, k, \alpha, \beta, S, W)$ where n, k, α, and β are integers satisfying $2 \leq k \leq n$ and $1 \leq \alpha \leq \frac{n}{k} \leq \beta \leq n$; S is a subset of* edges *or distinct pairs from vertex set $V \coloneqq [n]$; and W is an $n \times n$ non-negative, symmetric matrix; find a k-partition $(V_1, \ldots, V_k)$ of V so that $\hat{c}(V_1, \ldots, V_k) = \sum_{h\in[k]} \sum_{\{i,j\}\in V_h} W_{ij}$ is minimum, and the following constraints are satisfied:*

1. *$\alpha \leq |V_i| \leq \beta$, for $i \in [k]$;*
2. *$\binom{V_i}{2} \cap S = \emptyset$, for $i \in [k]$.*

A feasible solution to **capacitated k-partition**, *i.e.* a k-partition $(V_1, \ldots, V_k)$ of $[n]$ satisfying the two constraints of 2.1.2, is called a *capacitated k-partition*, and $\hat{c}(V_1, \ldots, V_k)$ is written in place of the more cumbersome $\hat{c}((V_1, \ldots, V_k)) = \hat{c}(\mathcal{C})$ for any k-partition $\mathcal{C} = (V_1, \ldots, V_k)$.

Proposition 2.1.3. *Given integers n, k, α, and β, where $2 \leq k \leq n$ and $1 \leq \alpha \leq \frac{n}{k} \leq \beta \leq n$, as well as a set of pairs $S \subseteq \binom{[n]}{2}$, forming graph $G \coloneqq ([n], S)$, any capacitated k-partition $(V_1, \ldots, V_k)$ of $[n]$ (see Definition 2.1.2) must satisfy:*

$$n - \beta \cdot (k-1) \leq |V_h| \leq \min\{n - \alpha \cdot (k-1), \alpha(G)\},$$

where $\alpha(G)$ is the stable set number of G.

Proof. Suppose on the contrary that some k-partition $(V_1, \ldots, V_k)$ has a part, without loss of generality V_k, of size less than $n - \beta(k-1)$. Since we have $\sum_{h \in [k]} |V_h| = n$, and the inequality $\sum_{h \in [k-1]} |V_h| > \beta(k-1)$ is implied. An application of the pigeonhole principle now yields $|V_\ell| > \beta$ for some part $\ell \neq k$. But, by definition, we must also have $\alpha \leq |V_h| \leq \beta$ for each $h \in [k]$, which is a contradiction. A similar argument follows for the case $|V_k| > n - \alpha(k-1)$.

Now suppose $|V_k| > \alpha(G)$. By definition of $\alpha(G)$, there must be some pair $\{i, j\} \in S = E(G)$ which is contained in V_k. But this contradicts the fact that in any capacitated k-partition, we have $\binom{V_k}{2} \cap S = \emptyset$, and thus the claim is proved. □

Proposition 2.1.3 provides a method of tightening either capacity bound α or β, by respectively resetting them to the dominant values of $\max\{\alpha, n - \beta \cdot (k-1)\}$ and $\min\{\beta, n - \alpha \cdot (k-1), \alpha(G)\}$, where $G \coloneqq ([n], S)$ and $\alpha(G)$ denotes the size of the largest stable set in G. Unfortunately, the exact computation of $\alpha(G)$ is $\mathcal{NP}$-hard in general [40], though a known polynomial-time computable upper bound called the Lovász-theta function, $\vartheta(G)$ [72], suffices as to potentially tighten β.

The **maximum k-cut** problem is a problem related to **minimum k-partition**, which takes similar input (n, k, W), and asks for a maximum-weight *k-cut* of W, *i.e.* a maximum total sum over W of all pairs $\{i, j\}$ which are separated by a k-partition of $[n]$, over all such partitions. In fact, the problems are complementary in the sense that the weight of a k-partition $(V_1, \ldots, V_k)$ of $[n]$ (*i.e.* $\sum_{h \in [k]} \sum_{\{i,j\} \in \binom{V_h}{2}} W_{ij}$) plus the weight of the corresponding k-cut (*i.e.* $\sum_{h < \ell} \sum_{(i,j) \in V_h \times V_\ell} W_{ij}$) is the constant value

$$\sum_{1 \leq i < j \leq n} W_{ij}.$$

Thus since **maximum k-cut** is an $\mathcal{NP}$-hard problem (*cf.* Karp [61] for the special case $k = 2$, **maximum cut**), **capacitated k-partition** is at least as computationally difficult. Moreover, the equivalent **conference clustering problem** and, ultimately, the **conference scheduling problem** must also be $\mathcal{NP}$-hard to solve to optimality. These observations can be strengthened as follows: first, define the *feasibility version* of optimization problem **capacitated k-partition** as the question asking whether or not the input $\mathcal{I} \coloneqq (n, k, \alpha, \beta, S, W)$ permits a feasible capacitated k-partition, *i.e.* a k-partition of $[n]$ satisfying the two constraints of 2.1.2.

Proposition 2.1.4. *The feasibility version of* ***capacitated*** *k-**partition**, for $k \geq 3$, is $\mathcal{NP}$-complete, even for input with fixed parameters $\alpha = 1$, and $\beta = n - k + 1$.*

Proof. The result follows by a reduction from the $\mathcal{NP}$-complete decision problem **vertex coloring** (*cf.* [61]), which asks to decide whether an input graph G is *k-colorable*, *i.e.* whether a function on the vertices $\xi\colon V(G) \to [k]$ exists so that $\xi(u) \neq \xi(v)$ for each $\{u, v\} \in E(G)$. Indeed, it is sufficient to produce such a reduction from the case $k \geq 3$, and to show that **capacitated k-partition** is in $\mathcal{NP}$.[1] The latter claim is trivial since a family of k subsets of $[n]$ of input $\mathcal{I}$ can be easily verified as being a capacitated k-partition or not, in time polynomial in n and k.

For the reduction, take any instance (k, G) of **vertex coloring**, let $n \coloneqq |V(G)|$, and define a tuple $\mathcal{I} \coloneqq (n, k, 1, n-k+1, E(G), W)$ as the input to the feasibility version of **capacitated k-partition**, where W can be arbitrary. Here, supposing $V(G) = [n]$, a given k-coloring ξ of G defines a k-partition $(V_1, \ldots, V_k)$ of $V(G)$ where $V_h \coloneqq \xi^{-1}(h)$, $h \in [k]$, and conversely any given k-partition $(V_1, \ldots, V_k)$ of $V(G)$ trivially yields the coloring function defined by $\xi(i) \coloneqq h$, $i \in V_h$, $h \in [k]$. Thus a k-coloring of G exists if and only if a k-partition of $[n]$ exists which satisfies both constraints of 2.1.2 (the first is satisfied trivially), and so the first claim holds.

The last claim now holds trivially, since $\alpha = 1$ and $\beta = n - k + 1$ were fixed in the reduction. □

[1] The decision problem **vertex coloring** with $k = 2$ is indeed in $\mathcal{P}$, as the problem asks whether a graph is 2-colorable, *i.e.* bipartite, which can be answered—and in fact a bipartition can be produced—in linear time via a breadth-first search (see *e.g.* [70]).

The above result now implies the following.

Proposition 2.1.5. *The* ***capacitated*** k*-****partition*** *problem, for* $k \geq 2$*, is* $\mathcal{NP}$*-hard. Moreover, assuming* $\mathcal{P} \neq \mathcal{NP}$ *and* $k \geq 3$*, for every constant* c*, there does not exist a polynomial time* c*-approximation algorithm.*

Proof. The first claim is obvious from 2.1.4.

As for the second claim, suppose instead that a polynomial-time c-approximation algorithm does exist for an instance $(n, k, \alpha, \beta, S, W)$. This implies that we have a polynomial-time procedure which generates feasible (and in fact, provably good) k-partitions of $[n]$, in polynomial time. But, under the assumption $\mathcal{P} \neq \mathcal{NP}$, Proposition 2.1.4 is then contradicted, and thus our supposition must be false. □

The above results show that not only is solving **capacitated** k**-partition** to optimality a computationally intractable problem, but even designing a heuristic procedure to generate feasible solutions, regardless of quality guarantee, is just as difficult. This justisfies our use of potentially exponential running time tools like branch-and-bound in our algorithm design, discussed in the next chapter.

2.2 Modeling the problem

A natural first step towards modeling an instance $\mathcal{I} \coloneqq (n, k, \alpha, \beta, S, W)$ of **capacitated** k**-partition** is to define decision variables which assign each vertex one of k labels, *i.e.* for $i \in [n]$ and $h \in [k]$, let $Z_{ih} \in \{0, 1\}$ be a variable encoding the labeling of vertex $i \in [n]$ with h if and only if $Z_{ih} = 1$. However, relating Proposition 2.1.1 to the terminology of **capacitated** k**-partition**, there are exponentially many choices of matrices $Z \in \{0, 1\}^{n \times k}$ which encode k-partitions of identical costs, since the number of ways of labeling parts is $k! = \Theta(e^{k \log k})$. Thus we reduce the number of variables by eliminating labels and focusing on choosing which vertices are grouped together, *i.e.* for each $i, j \in [n]$ let $X_{ij} \in \{0, 1\}$ be such that $X_{ij} = 1$ if and only if vertices i and j of $[n]$ are placed together—by convention, let $X_{ii} \coloneqq 1$. This change of variables yields a tradeoff in problem complexity, since now a quadratic relationship $X = ZZ^\top$ exists

between the two binary matrices $Z \in \{0,1\}^{n\times k}$ and $X \in \{0,1\}^{n\times n}$.

The encoding of 'TRUE' and 'FALSE' as integers '1' and '0', respectively, in the above decision variables is an intuitive and convenient choice, but perhaps arbitrary. In general one can define a decision variable $\zeta \in \mathbb{R}$ to take on a value of 'b' whenever 'TRUE' and 'a' whenever 'FALSE', so long as $a \neq b$. For example, consider the **maximum k-cut** problem, where we recall that the input consists of integers n and k, and a non-negative, symmetric matrix $W \in \mathbb{R}^{n\times n}$, and the objective is to find a maximum-weight k-cut with respect to W. For general $k \geq 2$, Frieze and Jerrum [38] modeled the problem geometrically, by focusing on assigning each vertex of V to an element of a set $A \subseteq \mathbb{R}^{k-1}$ of size k.[2] The particular choice of A enabled a reformulation of the problem in terms of pairing vertices together, where for each $i, j \in V$, a decision variable $Y_{ij} \in \left\{\frac{-1}{k-1}, 1\right\}$, was interpreted as $Y_{ij} = \frac{-1}{k-1}$ if and only if vertices i and j were separated by the k-partition. In other words, for the property "vertex pair is grouped together", 'TRUE' is represented by the usual $b = 1$, but 'FALSE' is encoded by the fractional value $a = -1/(k-1)$. As mentioned earlier in §1.1, a semidefinite programming (SDP) relaxation of the **maximum k-cut** model in the $\left\{\frac{-1}{k-1}, 1\right\}$ space results in an α_k-approximation algorithm, for each $k \geq 2$, where α_k is a provably "good" constant factor (*cf.* [38]). Furthermore, it was shown in [31] that this SDP relaxation is strong in relation to a corresponding LP relaxation, which is discussed further in the sequel.

Though we cannot guarantee any reasonable approximation for **capacitated k-partition** (*cf.* Proposition 2.1.5), the variable space mentioned above for the **maximum k-cut** problem is chosen, since it may still yield strong convex relaxations in practice. Thus we explicitly present the model for **maximum k-cut** in the next section, and follow up with a formulation for the capacitated problem.

---

[2]The set A contains k vectors of the form $a_i := b_i - c$, where $\{b_i \,|\, i \in [k]\}$ are the endpoints of an equilateral k-simplex with centroid c, and each a_i is scaled so that $\|a_i\| = 1$. This was proved in [38] to yield $a_i \cdot a_j = \frac{-1}{k-1}$ if and only if $i \neq j$.

2.2.1 The maximum k-cut model

Let $\mathcal{Y}(n,k)$ denote the set of all *k-partition matrices* $Y \in \left\{\frac{-1}{k-1}, 1\right\}^{n\times n}$, where, for any $i,j \in V \coloneqq [n]$, we have $Y_{ij} = \frac{-1}{k-1}$ if and only if vertices i and j are not grouped together. The **maximum k-cut** problem, with input (n,k,W), can then be written as:

$$\begin{array}{lll} \text{(MKC)} & \text{maximize} & \frac{k-1}{k}\sum_{\{i,j\}\in\binom{V}{2}} W_{ij}(1-Y_{ij}) \\ & \text{subject to} & Y \in \mathcal{Y}(n,k). \end{array} \tag{2.9}$$

Indeed, for any pair $\{i,j\} \in \binom{V}{2}$, the term $\frac{k-1}{k}W_{ij}(1-Y_{ij})$ reduces to W_{ij} whenever $Y_{ij} = \frac{-1}{k-1}$, and 0 otherwise. Eisenblätter [31] proved that the slightly larger set of matrices $\bigcup_{r\le k}\mathcal{Y}(n,r)$ is characterized by the set of $Y \in \left\{\frac{-1}{k-1}, 1\right\}^{n\times n}$ with $Y_{ii} = 1$, for each $i \in [n]$, and Y positive semidefinite, *i.e.* the eigenvalues of Y are non-negative. We write the latter property as $Y \in \mathcal{S}^n_+$, where $\mathcal{S}^n$ denotes the space of symmetric, $n \times n$ matrices and $\mathcal{S}^n_+$ denotes the convex cone containing all positive semidefinite matrices $Y \in \mathcal{S}^n$ which are positive semidefinite.

Defining $A \bullet B \coloneqq \sum_{i\in[m],j\in[n]} A_{ij}B_{ij}$ for any $m \times n$ matrices A and B, e as the vector of ones and $J \coloneqq ee^\top$ as the matrix of all ones (dimension can be found from context), and $\operatorname{diag}(A)$ to be the vector of diagonal entries $(A_{ii} : i \in [n])$, we can rewrite **maximum k-cut** as follows:

$$\text{(MKC)} \quad \text{maximize} \quad \tfrac{k-1}{2k}W \bullet (J-Y) \tag{2.10}$$

$$\text{subject to } \operatorname{diag}(Y) = e \tag{2.11}$$

$$Y \in \mathcal{S}^n_+ \tag{2.12}$$

$$Y \in \left\{\tfrac{-1}{k-1}, 1\right\}^{n\times n}. \tag{2.13}$$

The SDP relaxation of the above system (2.10)–(2.13) is obtained by replacing the binary constraint $Y \in \left\{\frac{-1}{k-1}, 1\right\}^{n\times n}$ with the continuous bounds $\frac{-1}{k-1} \le Y_{ij} \le 1$, for each $i,j \in [n]$. Notice that the upper bound is superfluous, as the semidefiniteness of Y implies that all off-diagonal entries Y_{ij} are bounded in magnitude by the diagonals Y_{ii} and Y_{jj}, which are both of value 1. The set of matrices of this relaxation coincides with the *truncated elliptope*

$$\mathcal{E}(n,k) \coloneqq \left\{Y \in \mathcal{S}^n_+ \,\middle|\, \operatorname{diag}(Y) = e, Y \ge \tfrac{-1}{k-1}J\right\},$$

or the intersection of half-spaces $Y \geq \frac{-1}{k-1}J$ with the *elliptope* $\mathcal{E}(n) \coloneqq \mathcal{E}(n,2) = \left\{Y \in \mathcal{S}^n_+ \,\middle|\, \operatorname{diag}(Y) = e\right\}$.

It is curious that solving the SDP relaxation of MKC in the $\left\{\frac{-1}{k-1}, 1\right\}$ space yields provably strong approximation guarantees. Frieze and Jerrum [38] prove that, in expectation, a randomized rounding procedure applied to the above SDP relaxation of MKC, yields a factor which is strictly greater than $1 - \frac{1}{k}$, but is asymptotically close to $1 - \frac{1}{k} + \frac{\log(k)}{2k^2}$. Intuitively, their rounding approach is as follows, for a given SDP solution matrix $Y \in \mathcal{E}(n,k)$: first find a set of unit vectors $\{v_i \,|\, i \in [n]\} \subseteq \mathbb{R}^n$ so that $v_i^\top v_j = Y_{ij}$, for each $i, j \in [n]$ (by, *e.g.* Cholesky factorization); then generate k random hyperplanes, *i.e.* a set of unit vectors $\{r_i \,|\, i \in [k]\} \subseteq \mathbb{R}^n$; and finally form a k-partition of $[n]$ by defining parts $V_i \coloneqq \left\{j \in [n] \,\middle|\, v_j^\top r_i \geq v_j^\top r_h, \forall h \in [k]\right\}$, breaking ties arbitrarily.

The intersection of the elliptope $\mathcal{E}(n,k)$ with valid inequalities, or linear inequalities satisfied by all feasible points $Y \in \mathcal{Y}(n,k)$, can only strengthen the quality of the actual SDP relaxation of MKC, yet there is theoretical evidence against the possibility of strict improvements [60, 59, 62], as well as practical evidence in support of the addition of inequalities [5, 31]. Towards the latter point, for any $Y \in \left\{\frac{-1}{k-1}, 1\right\}^{n \times n}$ with $\operatorname{diag}(Y) = e$, matrix Y is positive semidefinite if and only if it satisfies a set of (exponentially many) well-known inequalities—the triangle and clique inequalities.

The *triangle inequalities* encode the fact that the partitioning of vertices is transitive, *i.e.* if vertices i and j are together in some part, and j and h are also grouped together, then i and h must certainly be together. Thus, each $Y \in \mathcal{Y}(n,k)$ must satisfy:

$$\begin{aligned} Y_{ij} + Y_{jh} - Y_{hi} &\leq 1 \\ Y_{ij} - Y_{jh} + Y_{hi} &\leq 1 \quad \text{for } \{i,j,h\} \in \binom{[n]}{3} \\ -Y_{ij} + Y_{jh} + Y_{hi} &\leq 1. \end{aligned} \tag{2.14}$$

Indeed, these inequalities essentially cut away matrices $Y \in \left\{\frac{-1}{k-1}, 1\right\}^{n \times n}$, which have $Y_{ih} = Y_{jh} = 1$ and $Y_{ij} = \frac{-1}{k-1}$ for any triangle $\{i,j,h\} \in \binom{[n]}{3}$. We succinctly rewrite the $3\binom{n}{3}$ triangle inequalities as $\Delta(Y) \leq e$.

The *clique inequalities* impose the condition that for any subset $K \subseteq [n]$ of $k+1$

vertices, at least one pair must be grouped together, for otherwise K is a $(k+1)$-partition. Thus the following holds for each $Y \in \mathcal{Y}(n,k)$:

$$\sum_{\{i,j\} \in \binom{K}{2}} Y_{ij} \geq \frac{-k}{2} \quad \text{for } K \subseteq [n] : |K| = k+1. \tag{2.15}$$

To see this, the summand of matrix entries Y_{ij} over pairs $\{i,j\} \in K$ must be at least

$$\frac{-1}{k-1}\left(\binom{k+1}{2} - 1\right) + 1 = \frac{-k}{2}.$$

Thus, letting $\mathcal{K}(Y) \geq \frac{-k}{2}e$ denote the set of clique inequalities, we can rewrite MKC as a linear disjunctive programming formulation

$$\begin{aligned}
\text{(MKC)} \quad \text{maximize } & \tfrac{k-1}{2k} W \bullet (J - Y) && (2.16)\\
\text{subject to } & \operatorname{diag}(Y) = e && (2.17)\\
& \Delta(Y) \leq e && (2.18)\\
& \mathcal{K}(Y) \geq \tfrac{-k}{2}e && (2.19)\\
& Y \in \left\{\tfrac{-1}{k-1}, 1\right\}^{n \times n}. && (2.20)
\end{aligned}$$

Although the triangle and clique inequalities are strong enough to replace the nonlinear semidefinite constraint $Y \in \mathcal{S}^n_+$ in the MKC formulation above, this representation is unfortunately an exponential-sized system since the number of $k+1$-cliques amounts to $\binom{n}{k+1}$, which is at least $\left(\frac{n}{k+1}\right)^{k+1}$, or $\Omega(2^{k+1})$. However, we will show in the next section that the truncated elliptope $\mathcal{E}(n,k)$ provides a strong relaxation to MKC, *i.e.* it contains matrices which are provably "close" to satisfying the triangle and clique inequalities, as well as a more general class of inequalities encompassing these constraints. If we define $\mathcal{P}_{\text{MKC}}(n,k)$ to be the MKC *polyhedron*—the convex hull of all k-partition *incidence vectors* $x \in \{0,1\}^{\binom{n}{2}}$, where $x_{ij} = 1$ if and only if vertices i and j are grouped together—then these results also show that any $Y \in \mathcal{E}(n,k)$, when translated to $[0,1]^{\binom{n}{2}}$ in a natural way (details to follow), is much "closer" to satisfying the valid inequalities of $\mathcal{P}_{\text{MKC}}(n,k)$ then any corresponding LP relaxation in which $x = 0$ is a feasible point.

The above discussion motivates the formulation of our **capacitated k-partition** model in the $\left\{\frac{-1}{k-1}, 1\right\}$ variables, rather than in the typical $\{0,1\}$ setting, which is

discussed next. We conclude this section by displaying the SDP relaxation of MKC equipped with the triangle and clique inequalities below:

$$\begin{aligned}
\text{(SMKC)} \quad \text{maximize} \quad & \tfrac{k-1}{2k} W \bullet (J - Y) && (2.21)\\
\text{subject to} \quad & \operatorname{diag}(Y) = e && (2.22)\\
& \Delta(Y) \leq e && (2.23)\\
& \mathcal{K}(Y) \geq \tfrac{-k}{2} e && (2.24)\\
& Y \geq \tfrac{-1}{k-1} J && (2.25)\\
& Y \in \mathcal{S}^n_+. && (2.26)
\end{aligned}$$

2.2.2 Reformulation into capacitated *k*-partition

Now we are able to present the initial model of the **capacitated k-partition** problem as defined in Definition 2.1.2, by extending the MKC formulations found in the previous section, *i.e.* the SDP-based model (2.10)–(2.13), the IP-based formulation (2.16)–(2.20), and the SDP-relaxation (2.21)–(2.26). Recall that the input tuple is denoted $\mathcal{I} = (n, k, \alpha, \beta, S, W)$, and that we are searching for a particular k-partition matrix $Y \in \mathcal{Y}(n, k)$ of minimum cost: one whose corresponding k-partition has parts with sizes bounded between α and β, and forbids such parts from containing any pair $\{i, j\} \in S$. Let $\mathcal{Y}(n, k, \alpha, \beta, S)$ denote the set of feasible matrices. The cost of any such Y is complementary to its cost in MKC, and is written as:

$$\sum_{\{i,j\} \in \binom{[n]}{2}} W_{ij} - \left(\tfrac{k-1}{2k} W \bullet (J - Y)\right) = \tfrac{k-1}{2k} W \bullet \left(\tfrac{1}{k-1} J + Y\right).$$

Since the polyhedra described in the sequel arise from convex combinations of arbitrary partition incidence vectors $x \in \{0, 1\}^{\binom{n}{2}}$, we present valid inequalities in terms of such variables, and then linearly transform them to inequalities in terms of symmetric matrices $Y \in \mathcal{Y}(n, k) \subseteq \left\{\frac{-1}{k-1}, 1\right\}^{n \times n}$. Indeed, this is done via the linear map $f : \mathbb{R} \to \mathbb{R}$ defined as $f(z) \coloneqq \frac{k-1}{k} z + \frac{1}{k}$. In particular, notice that $f(0) = \frac{-1}{k-1}$ and $f(1) = 1$, and so applying this to any linear constraint satisfied by $x \in \{0, 1\}^{\binom{n}{2}}$, we have

$$\sum_{\{i,j\} \in \binom{[n]}{2}} A_{ij} x_{ij} \leq b \qquad (2.27)$$

if and only if the unique symmetric matrix $Y \in \left\{\frac{-1}{k-1}, 1\right\}^{n\times n}$ with $\mathrm{diag}(Y) = 1$, and satisfying $f(Y_{ij}) = x_{ij}$ for $\{i,j\} \in \binom{[n]}{2}$, is such that

$$\begin{aligned} & \sum_{\{i,j\}\in\binom{[n]}{2}} A_{ij} f(Y_{ij}) \leq b \\ \iff & \sum_{\{i,j\}\in\binom{[n]}{2}} A_{ij}\left(\tfrac{k-1}{k} Y_{ij} + \tfrac{1}{k}\right) \leq b \\ \iff & \sum_{\{i,j\}\in\binom{[n]}{2}} A_{ij} Y_{ij} \leq \tfrac{k}{k-1} b + \tfrac{-1}{k-1} \sum_{\{i,j\}\in\binom{[n]}{2}} A_{ij}. \end{aligned} \tag{2.28}$$

Now, given $\mathcal{I} = (n, k, \alpha, \beta, S, W)$, we represent the capacitated k-partition incidence vectors $x \in \{0,1\}^{\binom{n}{2}}$ as per Definition 2.1.2. There are two additional constraints which must be satisfied by x. The first constraint asks for the size of each part of a k-partition be bounded between α and β, which is clearly encoded by

$$\alpha - 1 \leq \sum_{j\in[n]\setminus\{i\}} x_{ij} \leq \beta - 1, \quad \text{for } i \in [n].$$

The second condition asks for every pair $\{i,j\} \in S$ to cross distinct parts, and so we have

$$x_{ij} = 0 \quad \text{for } \{i,j\} \in S.$$

With appropriate transformation, each $Y \in \mathcal{Y}(n,k)$ must now satisfy $\sum_{j\in[n]\setminus\{i\}} Y_{ij} \leq \frac{k}{k-1}(\beta-1) - \frac{1}{k-1}(n-1) = \frac{k\beta-n}{k-1} - 1$ and, similarly, $\sum_{j\in[n]\setminus\{i\}} Y_{ij} \geq \frac{k\alpha-n}{k-1} - 1$, for each $i \in [n]$, and further $Y_{ij} = \frac{-1}{k-1}$, for each $\{i,j\} \in S$. Noting that $Y_{ii} = 1$ for each $i \in [n]$, the former pair of constraints can be rewritten as

$$\tfrac{\alpha k-n}{k-1} e \leq Ye \leq \tfrac{\beta k-n}{k-1} e.$$

The model for **capacitated k-partition** can now be formed by intersecting the SDP-based constraint set of MKC, *i.e.* (2.11)–(2.13), with the two new types of constraints (note the change in objective):

$$\text{(CKP)} \quad \text{minimize} \quad \tfrac{k-1}{2k} W \bullet \left(\tfrac{1}{k-1} J + Y\right) \tag{2.29}$$

$$\text{subject to } \mathrm{diag}(Y) = e \tag{2.30}$$

$$Y_{ij} = \tfrac{-1}{k-1} \qquad \text{for } \{i,j\} \in S \tag{2.31}$$

$$\tfrac{k\alpha-n}{k-1}e \leq Ye \leq \tfrac{k\beta-n}{k-1}e \tag{2.32}$$

$$Y \in \mathcal{S}^n_+ \tag{2.33}$$

$$Y \in \left\{\tfrac{-1}{k-1}, 1\right\}^{n\times n}. \tag{2.34}$$

Similarly to the **maximum k-cut** problem, we can relax this formulation of CKP to a semidefinite program, by simply replacing the binary constraint (2.34) of (2.29)–(2.34), with the lower bound inequality

$$Y \geq \tfrac{-1}{k-1}J. \tag{2.35}$$

Furthermore, we can encode the problem as a linear IP by intersecting the constraint set (2.16)–(2.20) of MKC with the newly introduced constraints (2.31) and (2.32):

$$\begin{aligned}
\text{(CKP)} \quad \text{minimize} \quad & \tfrac{k-1}{2k}W \bullet \left(\tfrac{1}{k-1}J + Y\right) && && (2.36)\\
\text{subject to} \quad & \operatorname{diag}(Y) = e && && (2.37)\\
& Y_{ij} = \tfrac{-1}{k-1} && \text{for } \{i,j\} \in S && (2.38)\\
& \tfrac{k\alpha-n}{k-1}e \leq Ye \leq \tfrac{k\beta-n}{k-1}e && && (2.39)\\
& \Delta(Y) \leq e && && (2.40)\\
& \mathcal{K}(Y) \geq \tfrac{-k}{2}e && && (2.41)\\
& Y \in \left\{\tfrac{-1}{k-1}, 1\right\}^{n\times n}. && && (2.42)
\end{aligned}$$

The next section explores the known inequalities of polyhedra of optimization problems related to **capacitated k-partition**, which will be used when appropriate to partially describe its polyhedron $\mathcal{P}_{\text{CKP}}(n, k, \alpha, \beta, S)$—shortened to $\mathcal{P}_{\text{CKP}}$ when parameters are clear from context—*i.e.* the convex combination of all k-partition incidence vectors $x \in \{0,1\}^{\binom{n}{2}}$ which represent capacitated k-partitions. In turn, such inequalities will be used to strengthen the convex SDP relaxation of CKP, which will aid in the overall algorithm design for solving **capacitated k-partition** to optimality (Chapter 3).

2.3 Valid inequalities of capacitated k-partition polyhedron

In this section we review the structure of various polyhedra corresponding to optimization problems closely related to **capacitated** k**-partition** (2.1.2). First let $(n, k_1, k_2, \alpha, \beta, S, a)$ be a tuple where n, k_1, k_2, α, and β are positive integers satisfying $2 \leq k_1 \leq k_2 \leq n$ and $1 \leq \alpha \leq \beta \leq n$, S is a subset of pairs of vertices $[n]$, and $a = (a_i : i \in [n])$ is an n-vector with positive, integral entries. Now define the parameterized polyhedron

$$\mathcal{P}(n, k_1, k_2, \alpha, \beta, S, a) \subseteq \mathbb{R}^{\binom{n}{2}},$$

to be the convex hull of the set of all incidence vectors $x \in \{0,1\}^{\binom{n}{2}}$ corresponding to partitions $\mathcal{V}$ of $[n]$ with $k_1 \leq |\mathcal{V}| \leq k_2$, where each part $P \in \mathcal{V}$ has bounded total *weight*, *i.e.*

$$\alpha \leq \sum_{i \in P} a_i \leq \beta,$$

and no part contains any pair $\{i, j\} \in S$. Note that whenever $a = e$, *i.e.* the vector of all ones, this reduces to the case where the total weight of P is simply its cardinality $|P|$. We include the weighted case, as it will be needed in our algorithm.

This parameterized polyhedron encompasses many known structures. First notice that for any fixed $n \in \mathbb{Z}_+$ and weight vector $a \in \mathbb{Z}_+^n$, we have the following containment:

$$\mathcal{P}(n, k_1, k_2, \alpha, \beta, S, a) \subseteq \mathcal{P}(n, k_1', k_2', \alpha', \beta', S', a), \tag{2.43}$$

for any integers satisfying $k_1' \leq k_1$, $k_2 \leq k_2'$, $\alpha' \leq \alpha$, $\beta \leq \beta'$, and sets $S' \subseteq S$. Indeed, every feasible partition of the former polyhedron, is also feasible in the latter polyhedron. This will be helpful in generating valid inequalities for $\mathcal{P}_{\text{CKP}}$, since for any $\mathcal{P} \supseteq \mathcal{P}_{\text{CKP}}$, valid inequalities of $\mathcal{P}$ cannot be violated by points of $\mathcal{P}_{\text{CKP}}$.

In particular, whenever $k \coloneqq k_1 = k_2$, we have

$$\mathcal{P}_{\text{CKP}}(n, k, \alpha, \beta, S) = \mathcal{P}(n, k, k, \alpha, \beta, S, e),$$

and so the containment

$$\mathcal{P}_{\text{CKP}}(n, k, \alpha, \beta, S) \subseteq \mathcal{P}(n, k_1', k_2', \alpha', \beta', S', e),$$

holds for any integers $k'_1 \leq k \leq k'_2$, $\alpha' \leq \alpha$, $\beta \leq \beta'$, and $S' \subseteq S$. As an example, consider the MKC polyhedron:

$$\mathcal{P}_{\mathrm{MKC}}(n,k) = \mathcal{P}(n,k,k,1,n,\emptyset,e).$$

This clearly contains the CKP polyhedron for any integers n, k, $1 \leq \alpha \leq \beta \neq n$, and any set $S \subseteq \binom{[n]}{2}$. Our goal now is to explore polyhedra which contain $\mathcal{P}_{\mathrm{CKP}}$, and ultimately provide tight valid inequalities.

For each type of inequality presented, we describe at least one separation algorithm, which may be heuristic in nature—especially if the corresponding *separation problem* is $\mathcal{NP}$-hard, *i.e.* the problem of determining whether or not some inequality of the given type is violated by an arbitrary point $x^* \in \mathbb{R}^{\binom{n}{2}}$. We focus on inequalities which are both empirically and theoretically strong, *e.g.* those which are *facet-defining*, or necessary to the polyhedral description. We note that such inequalities will be first presented in terms of variables $x \in \{0,1\}^{\binom{n}{2}}$, and then transformed to the space $Y \in \left\{\frac{-1}{k-1}, 1\right\}^{n \times n}$, so that any valid inequality of $\mathcal{P}_{\mathrm{CKP}}(n,k,\alpha,\beta,S)$ yields a "shifted" valid inequality of $\mathcal{Y}(n,k,\alpha,\beta,S)$.

Since our algorithm (*cf.* Chapter 3) for solving CKP instances uses SDP relaxations equipped with cutting planes, we would like to compare the strength of semidefinite matrices in $\mathcal{E}(n,k)$ prior to the addition of any valid inequalities. We do so by linearly transforming solutions in the space $\left[\frac{-1}{k-1}, 1\right]^{n \times n}$ to the $[0,1]$ setting, and then quantifying a measure of violation for each valid inequality of $\mathcal{P}_{\mathrm{MKC}}(n,k)$. To this end, we first define a "shifted" set of vectors, using the previously defined linear function f:

$$\theta(n,k) \coloneqq \left\{ x \in \mathbb{R}^{\binom{n}{2}} \,\middle|\, x_{ij} = f(Y_{ij}), \forall \{i,j\} \in \binom{[n]}{2}, \forall Y \in \mathcal{E}(n,k) \right\}.$$

Notice that $\mathcal{E}(n,k)$ and $\mathcal{Y}(n,k)$ share the same set of k-partition matrices, and so the transformed set $\theta(n,k)$ and $\mathcal{P}_{\mathrm{MKC}}(n,k)$ share the same set of k-partition incidence vectors. Now recall the fact that $Y \in \mathcal{S}^n_+$ if and only if $b^\top Y b \geq 0$ for each $b \in \mathbb{R}^n$, and, letting $V_+ \coloneqq \{i \in [n] \,|\, b_i > 0\}$ and $V_- \coloneqq \{i \in [n] \,|\, b_i < 0\}$, we rewrite this inequality as:

$$\sum_{\{i,j\} \in \binom{V_-}{2} \cup \binom{V_+}{2}} Y_{ij}|b_i b_j| - \sum_{(i,j) \in V_- \times V_+} Y_{ij}|b_i b_j| \geq \tfrac{-1}{2} \sum_{i \in [n]} b_i^2, \tag{2.44}$$

for each $b \in \mathbb{R}^n$. This inequality holds for any $Y \in \mathcal{E}(n,k)$, and so the linearly transformed version:

$$\sum_{\{i,j\} \in \binom{V_-}{2} \cup \binom{V_+}{2}} x_{ij}|b_i b_j| - \sum_{(i,j) \in V_- \times V_+} x_{ij}|b_i b_j| \geq \tag{2.45}$$
$$\frac{k-1}{k}\left(\frac{-1}{2}\sum_{i \in [n]} b_i^2\right) + \frac{1}{k}\left(\sum_{\{i,j\} \in \binom{V_-}{2} \cup \binom{V_+}{2}} |b_i b_j| - \sum_{(i,j) \in V_- \times V_+} |b_i b_j|\right),$$

must hold for each $x \in \theta(n,k)$. We use this generic inequality to derive bounds on the amount each valid inequality of $\mathcal{P}_{\text{MKC}}(n,k)$ is violated. In particular, if a valid inequality is of the form $\sum_{\{i,j\} \in \binom{[n]}{2}} A_{ij} x_{ij} \leq B$, and we can decompose $A_{ij} = b_i b_j$, for each $\{i,j\} \in \binom{[n]}{2}$, and some $b \in \mathbb{R}^n$, then the amount this valid inequality is violated is precisely the value on the right-hand side of (2.45), less B. Fortunately, the form of the above inequality is general enough contain many of the valid inequalities we present in the sequel as sub-classes, thus enabling such a comparison of $\mathcal{E}(n,k)$ to $\mathcal{Y}(n,k)$.

2.3.1 Triangle and clique inequalities

For any positive integers n and k such that $2 \leq k \leq n$, define the polytopes $\mathcal{P}_{\leq \text{KP}}(n,k)$ and $\mathcal{P}_{\geq \text{KP}}(n,k)$ to be the convex hull of all r-partition incidence vectors, for $2 \leq r \leq k$ and $k \leq r \leq n$, respectively. Thus these two polyhedra can be rewritten as special cases of the parameterized polyhedron $\mathcal{P}$ introduced in the beginning of the previous section:

$$\mathcal{P}_{\leq \text{KP}}(n,k) = \mathcal{P}(n,2,k,1,n,\emptyset,e)$$
$$\mathcal{P}_{\geq \text{KP}}(n,k) = \mathcal{P}(n,k,n,1,n,\emptyset,e).$$

With this equivalence, it is clear that the following series of containments holds, where (n,k,α,β,S,W) is the typical input tuple for **capacitated k-partition**:

$$\mathcal{P}_{\text{CKP}}(n,k,\alpha,\beta,S) \subseteq \mathcal{P}_{\text{MKC}}(n,k) = \mathcal{P}_{\leq \text{KP}}(n,k) \cap \mathcal{P}_{\geq \text{KP}}(n,k)$$

Thus, any valid inequality for $\mathcal{P}_{\leq \text{KP}}(n,k)$ or $\mathcal{P}_{\geq \text{KP}}(n,k)$ is valid for $\mathcal{P}_{\text{CKP}}(n,k,\alpha,\beta,S)$. However, it can seen that r-partitions with large r are desirable when minimizing the function $\sum_{i<j} W_{ij} x_{ij}$, for given entries $W_{ij} \geq 0$ and variables $x_{ij} \geq 0$. Indeed, the

bounds in Turán's Theorem 2.3.1 (below) tells us that the minimum number of non-zero entries of any r-partition incidence vector $x \in \{0,1\}^{\binom{n}{2}}$, *i.e.* the minimum size of the *support* $\operatorname{supp}(x) \coloneqq \left\{ \{i,j\} \in \binom{[n]}{2} \,\middle|\, x_{ij} \neq 0 \right\}$, is strictly less than than that of any r'-partition with $r' < r$. Furthermore, since a majority of the valid inequalities discussed in [20] for $\mathcal{P}_{\geq \mathrm{KP}}(n,k)$ are of the form $\sum_{i<j} A_{ij} x_{ij} \leq b$ with $A_{ij} \geq 0$ and $b \geq 0$, it is unlikely that these types of inequalities with be violated (see also [27]). Hence we only focus on $\mathcal{P}_{\leq \mathrm{KP}}(n,k)$, and in our branch-and-bound algorithm, any arising k'-partitions with $k' > k$ are found and eliminated in the branching procedure (§3.5).

Proposition 2.3.1 (Turán's Theorem [95])**.** *For any graph G, let $n \coloneqq |V|$, $q \coloneqq \left\lfloor \frac{n}{k} \right\rfloor$, and $r \coloneqq n \mod k$. If the maximum size of a clique in G is at most k, i.e. $\omega(G) \leq k$, then*

$$\begin{aligned} |E(G)| &\leq \frac{nq(k-1)}{2} + \binom{r}{2} \\ &= \frac{n^2}{2}\left(1 - \frac{1}{k}\right) - \frac{r\left(k(n-r+1)-n\right)}{2k} \\ &\leq \frac{n^2}{2}\left(1 - \frac{1}{k}\right). \end{aligned}$$

Equivalently, if the maximum size of a stable in the complement $\overline{G}$ is at most k, i.e. $\alpha(\overline{G}) \leq k$, then

$$\begin{aligned} |E(\overline{G})| &\geq \binom{q}{2} k + qr \\ &= \frac{n(n-k)}{2k} + \frac{r(k-r)}{2k} \\ &\geq \frac{n(n-k)}{2k}. \end{aligned}$$

The first bound in each is tight when $\overline{G}$ is the disjoint union of $k-r$ cliques of size q, and r cliques of size $q+1$, and the second bound in each is tight when further k divides n, i.e. $r = 0$. □

The triangle inequalities

$$x_{ih} + x_{jh} - x_{ij} \leq 1 \quad \text{for distinct } i,j,h \in [n],$$

and lower bounds $x_{ij} \geq 0$, for $\{i,j\} \in \binom{[n]}{2}$ are satisfied by each incidence vector $x \in \mathcal{P}_{\leq \mathrm{KP}}(n,k)$, and are therefore valid for $\mathcal{P}_{\mathrm{CKP}}(n,k,\alpha,\beta,S)$. Indeed, the triangle

inequalities encode that fact that grouping vertices into parts is a transitive binary relation (see also (2.14)). Such inequalities are completely enumerable in polynomial time, since there are $3\binom{n}{3}$ in total, and thus we have a simple separation algorithm.

The clique inequalities, defined as

$$\sum_{\{i,j\}\in\binom{K}{2}} x_{ij} \geq 1 \quad \text{for } K \subseteq [n] : |K| = k+1, \tag{2.46}$$

are another class of valid inequalities for $\mathcal{P}_{\leq \mathrm{KP}}(n,k)$, since any $k+1$ vertices cannot all lie in separate parts of a partition (see also (2.15)).

It was shown in [20] that the above three types of inequalities—lower bound, triangle, and clique—are facet-defining and enough to completely describe $\mathcal{P}_{\leq \mathrm{KP}}(n,k)$. However, as mentioned earlier (§2.2.1), the number of clique inequalities becomes prohibitively large even for small values of k, since their number is $\binom{n}{k+1} \geq \left(\frac{n}{k}\right)^k$, or $\Omega(n^k)$ for fixed k. Moreover, the corresponding separation problem is $\mathcal{NP}$-hard. Indeed, determining whether or not a point $x^* \in [0,1]^{\binom{n}{2}}$ violates some clique inequality is equivalent to determining if the graph on n vertices with edge weights $x^*_{ij} \geq 0$ for each $\{i,j\} \in \binom{[n]}{2}$ permits a $(k+1)$-clique of minimum weight at most 1, *i.e.* the right-hand side of (2.46). Testing whether or not a clique of bounded weight exists is a well-known $\mathcal{NP}$-hard problem [40].[3]

The clique inequalities were shown by Eisenblätter [31] to be the "least dense" facet-defining inequalities of $\mathcal{P}_{\leq \mathrm{KP}}(n,k)$ for which the zero-vector $x = 0$ is not feasible. In other words, for any facet $\sum_{\{i,j\}\in\binom{[n]}{2}} A_{ij}x_{ij} \leq b$ with $b < 0$, must have $|\operatorname{supp} A| \geq \binom{k+1}{2}$. Thus, we attempt to separate them in our algorithm's cutting plane procedure using a simple "greedy" heuristic of Ghaddar *et al.* [41]. The algorithm, known as Greedy Clique Cover (GCC), contains a sub-routine (Algorithm 1) meant to find a single violated clique inequality, if one exists. Given positive integers n and k, an index $v \in [n]$, and a fractional point $x^* \in \mathbb{R}^{\binom{n}{2}}$—*e.g.* a feasible point of $\operatorname{conv} \mathcal{P}_{\mathrm{KP}}(n,k)$—initially set $K \coloneqq \{v\}$, and until $|K| = k+1$, iteratively find a vertex outside of K, *i.e.* $u \in [n]\setminus K$, for which the value $\delta_K(u) \coloneqq \sum_{ij\in\binom{K\cup\{u\}}{2}} x^*_{ij} - \sum_{ij\in\binom{K}{2}} x^*_{ij}$ is minimum, and replace K

[3] On the other hand, complete enumeration of the $\Theta(n^2)$ lower bounds and $\Theta(n^3)$ triangle inequalities can be performed in polynomial time.

Algorithm 1: greedy clique covering

Input : positive integers n and k, index $v \in [n]$, and $x \in \mathbb{R}^{\binom{n}{2}}$

Output: a set $K \subseteq [n]$ of size $k+1$

Initialize $K \coloneqq \{v\}$;

while $|K| < k+1$ **do**

  Find an index $i \in [n] \backslash K$ which minimizes $\sum_{j \in K} x_{ij}$;

  Add index i to set K;

with $K \cup \{u\}$. The resulting K encodes a violated clique inequality if $\sum_{\{i,j\} \in \binom{K}{2}} x^*_{ij} < 1$. This procedure can be called for each vertex $v \in [n]$—which is exactly what GCC does—testing for up to n possible violated clique inequalities. If ties are broken arbitrarily at any iteration when selecting $u \in \arg\min_{i \in [n] \backslash K} \delta_K(i)$, then the sub-routine may result in more than n possible violated inequalities. GCC was successful in practice in [41], and further tested with positive results in Anjos *et al.* [5].

The above valid inequalities of $\mathcal{P}_{\text{CKP}}(n, k, \alpha, \beta, S)$ are transformed to the $\left\{\frac{-1}{k-1}, 1\right\}$ space using (2.27) and (2.28). It is easy to see that the triangle inequalities remain unchanged in form, *i.e.* $\Delta(Y) \leq e$ (2.14) holds for any capacitated k-partition matrix Y. However, the lower bounds appear as $Y \geq \frac{-1}{k-1} J$, and the clique inequalities, take on the form

$$\sum_{\{i,j\} \in \binom{K}{2}} Y_{ij} \geq \tfrac{-k}{2} \qquad \text{for } K \subseteq [n] : |K| = k+1, \tag{2.47}$$

which confirms their appearances as such in CKP (*cf.* (2.36)–(2.42)).

To determine the strength of an SDP relaxation of MKC solved without the triangle or clique inequalities, we first bound the quantity $-x_{ij} + x_{ih} + x_{jh}$ from above, for any distinct indices $i, j, h \in [n]$, and $x \in \theta(n, k)$—the set of vectors corresponding to "shifted" points of $\mathcal{E}(n, k)$. Recalling that inequality (2.45) is valid for any $x \in \theta(n, k)$, and choosing a particular vector $b \in \mathbb{R}^n$ with $b_i = b_j = 1$ and $b_h = -1$ for any distinct three vertices $i, j, h \in [n]$, and setting the remaining entries set to zero, we can simplify the expression to:

$$-x_{ij} + x_{ih} + x_{jh} \leq 1 + \left(\tfrac{1}{2} - \tfrac{1}{2k}\right), \qquad \text{for } x \in \theta(n, k).$$

This shows that for $k \geq 2$, any semidefinite matrix $Y \in \mathcal{E}(n,k)$ yields a corresponding $x = f(Y) \in \theta(n,k)$, which violates the triangle inequalities by a "distance" of at most $\frac{1}{2} - \frac{1}{2k}$. This bound is can be tightened, as the next proposition shows.

Proposition 2.3.2 (Prop. 5, Eisenblätter [31])**.** *For any integers k and n satisfying $4 \leq k \leq n$, and any triangle $\{i,j,h\} \in \binom{[n]}{3}$, every $x \in \theta(n,k)$ must satisfy*

$$-x_{ij} + x_{ih} + x_{jh} \leq 1 + \frac{\sqrt{2(k-2)(k-1)} - (k-2)}{k} \quad \left[< \sqrt{2}\right]$$

and the bound is tight. □

Now to compare any $x \in \theta(n,k)$ against the clique inequalities, consider a set $K \subseteq [n]$ of size $k+1$ and defining $b_i = 1$ for each $i \in K$, and 0 otherwise. Simplifying (2.45), the following inequality is satisfied by each $x \in \theta(n,k)$:

$$\sum_{\{i,j\} \in \binom{K}{2}} x_{ij} \geq 1 - \left(\tfrac{1}{2} - \tfrac{1}{2k}\right).$$

This bound is in fact tight, which is also demonstrated in a more general setting (to be discussed) by Eisenblätter [31].

Similar to the triangle constraints, the above bound proves that no $x \in \theta(n,k)$, for $k \geq 2$, can violate the clique inequalities by more than $\frac{1}{2} - \frac{1}{2k} < \frac{1}{2}$. Also, since $x = 0$ is trivially a solution to the LP relaxation of MKC in the $\{0,1\}$ setting—*i.e.* the problem (2.10)–(2.13) with both the semidefinite constraint removed and the binary constraints $x_{ij} \in \{0,1\}$ replaced with continuous interval constraint $x_{ij} \in [0,1]$—we see that any $x' \in \theta(n,k)$ is at least twice as "close" as x is to satisfying the clique inequalities of $\mathcal{P}_{\text{MKC}}(n,k)$. In fact, this is true even when all constraints not cutting the 0-vector are included; the set of triangle inequalities are one example.

2.3.2 2-partition inequalities

For positive integers n, α, and β, such that $1 \leq \alpha \leq \beta \leq n$, $\mathcal{P}_{\text{CGP}}(n,\alpha,\beta) := \mathcal{P}(n,2,n,\alpha,\beta,\emptyset,e)$ denote the polyhedron arising from the instance (n,α,β) of **constrained graph partition**, a problem studied in Labbé and Öszoy [65]. In other words, this polytope is the convex hull of all r-partition incidence vectors $x \in \mathcal{P}_{\text{MKP}}(n,r)$ with

corresponding r-partition having parts of size bounded between α and β, for $2 \leq r \leq n$. Obviously, the only integers r for which such an r-partition exists are those values satisfying $r\alpha \leq n \leq r\beta$, *i.e.* $\left\lceil \frac{n}{\beta} \right\rceil \leq r \leq \left\lfloor \frac{n}{\alpha} \right\rfloor$. Thus we must have

$$\mathcal{P}_{\mathrm{CKP}}(n, r, \alpha, \beta, S) \subseteq \mathcal{P}_{\mathrm{CGP}}(n, \alpha, \beta) \qquad \text{for } \left\lceil \tfrac{n}{\beta} \right\rceil \leq r \leq \left\lfloor \tfrac{n}{\alpha} \right\rfloor .$$

We also observe

$$\mathcal{P}_{\mathrm{CGP}}(n, \alpha, \beta) \subseteq \mathcal{P}_{\mathrm{KP}}(n, r) \qquad \text{for } \; r \geq \left\lfloor \tfrac{n}{\alpha} \right\rfloor ,$$
$$\mathcal{P}_{\mathrm{CGP}}(n, \alpha, \beta) \subseteq \mathcal{P}_{\mathrm{MKC}}(n, k) \iff k = \left\lceil \tfrac{n}{\beta} \right\rceil = \left\lfloor \tfrac{n}{\alpha} \right\rfloor .$$

The above containments indicate that the **constrained graph partition** problem is defined generally enough to encompass other known problems as subclasses. Indeed, it is shown in [65] that the following problems previously described briefly in §1.1 arise from CGP for particular input tuples (n, α, β): **simple graph partition** [92], **clique partition** [45, 46], **size constrained clique partition** [57], and k-**equipartition** [75, 22, 23]. General conditions for full-dimensionality, as well as valid inequalities and criteria for when they are facet-defining can be found in [65]. Here we recall a few such relevant facts.

A k-partition $(V_1, \ldots, V_k)$ of $[n]$ is said to be 2-*loose* if $\beta - \alpha \geq 2$ and there exist two parts V_i and V_j satisfying the strict inequalities:

$$\alpha < |V_i| < \beta,$$
$$\alpha < |V_j| < \beta.$$

Proposition 2.3.3 ([65]). *If input parameters (n, α, β) to **constrained graph partition** permit a 2-loose k-partition, for $\left\lceil \frac{n}{\beta} \right\rceil \leq k \leq \left\lfloor \frac{n}{\alpha} \right\rfloor$, then $\mathcal{P}_{CGP}(n, \alpha, \beta)$ is full-dimensional.* □

Note that in fact there exists a 2-loose partition if and only if $\alpha k + 1 < n < \beta k - 1$. We record the following result since it demonstrates the strength of triangle inequalities in a capacitated class of partition problems. Furthermore, we notice that the data of our case study ICS-2013 (*cf.* §4)—a conference with $n = 61$, $\alpha = 5$, $\beta = 7$, and $k = 9$—allows 2-loose partitions (see §2.1 for a description of the original ConSP problem).

Proposition 2.3.4 ([65]). *If the input parameters* (n, α, β) *to* ***constrained graph partition*** *permit a 2-loose k-partition and* $\lfloor \frac{n}{\alpha} \rfloor \geq 2$, *i.e.* $n \geq 2\alpha + 2$, *then the triangle inequalities are facet-defining for* $\mathcal{P}_{CGP}(n, \alpha, \beta)$. □

Proposition 2.3.4 was only a remark in [65], appearing after it was proved that the following more general 2-partition inequalities can be facet-defining, under some fairly modest conditions. The 2-*partition inequalities* of Grötschel and Wakabayashi [46] are defined for any non-empty, disjoint subsets $S, T \subseteq [n]$, as:

$$\sum_{(i,j)\in S\times T} x_{ij} - \sum_{\{i,j\}\in\binom{S}{2}\cup\binom{T}{2}} x_{ij} \leq \min\{|S|, |T|\} \tag{2.48}$$

Intuitively, assuming without loss of generality $|S| \leq |T|$, the summand on the left-hand side of (2.48) is maximum over all r-partition incidence vectors $x \in \{0,1\}^{\binom{n}{2}}$, $1 \leq r \leq n$, when the parts P of partition $\mathcal{V}$ which intersect S also intersects T in the same number of vertices (*i.e.* $|P \cap S| = |P \cap T|$), and the remaining $|T| - |S|$ vertices of T are in distinct parts of $\mathcal{V}$. With some manipulation, such a vector x yields the following summand

$$\sum_{P\in\mathcal{V}} \left(|P \cap S| \cdot |P \cap T| - \binom{|P \cap S|}{2} - \binom{|P \cap T|}{2} \right) = |S|^2 - 2\binom{|S|}{2} = |S|,$$

and $|S| = \min\{|S|, |T|\}$ (see [46] for details). Thus the 2-partition inequalities are valid for $\mathcal{P}_{\leq \mathrm{KP}}(n, n)$, or equivalently, $\mathcal{P}_{\mathrm{MKP}}(n, k)$ for each $k = 2, \ldots, n$, and hence for polyhedra $\mathcal{P}_{\mathrm{CKP}}$.

Indeed, 2-partition inequalities are general enough to contain the triangle inequalities, particularly when $|S| = 1$ and $|T| = 2$. The 2-partition inequalities were introduced by Grötschel and Wakabayashi [46] for the **simple graph partition** or **clique partition** problem which asks for a maximum-weight r-partition, for any r. Oosten *et al.* [80] generalized the 2-partition inequalities further, and also showed the $\mathcal{NP}$-hardness of the separation problem: given a set S of fixed size (even the case $|S| = 1$), determine if there exists a disjoint set T such that the corresponding 2-partition inequality is violated.[4] The inequalities have been show to be facet-defining, for example, whenever $|S| \neq |T|$.

[4] As far as the author knows, the general separation problem, *i.e.* where $|S|$ is not fixed, still remains an open problem as presented in [80].

Algorithm 2: 2-partition randomized heuristic

Input : positive integer n, index $s \in [n]$, and non-negative vector $x \in \mathbb{R}_+^{\binom{n}{2}}$

Output: a set $T \subseteq [n]$

Initialize $T \coloneqq \emptyset$, $V \coloneqq \{i \in [n] \backslash \{s\} \,|\, x_{is} > 0\}$;

Let $\pi \colon \{1, \ldots, |V|\} \to V$ be an arbitrary permutation of V;

for $i = 1, \ldots, |V|$ **do**

  Let $u \coloneqq \pi(i)$;

  if $x_{us} - \sum_{v \in T} x_{uv} > 0$ **then**

    Add vertex u to set T;

Grötschel and Wakabayashi [45] present a useful separation heuristic algorithm, denoted GWA, which has the following sub-routine (Algorithm 2) for finding a violated 2-partition inequality with S fixed to be of size $|S| = 1$. Given a vertex $s \in [n]$, and fractional point $x \in \mathbb{R}^{\binom{n}{2}}$, define $S \coloneqq \{s\}$, $T \coloneqq \emptyset$, let V be all remaining indices i for which $x_{si} \neq 0$, *i.e.* $V \coloneqq \{i \in [n] \backslash \{s\} \,|\, x_{si} > 0\}$, and let $\pi \colon \{1, \ldots, |V|\} \to V$ be some permutation of V. For $i = 1, \ldots, |V|$, if $u \coloneqq \pi(i)$ satisfies

$$x_{us} > \sum_{v \in T} x_{uv},$$

then add u to the set T. After the sub-procedure finishes, the set T is output, and $(\{s\}, T)$ defines a violated 2-partition inequality if

$$\sum_{i \in T} x_{is} - \sum_{\{i,j\} \in \binom{T}{2}} x_{ij} > 1.$$

The overall algorithm GWA calls this subroutine for each vertex $s \in [n]$ and some random permutation π_s with respect to the computed set V, and then again for each vertex, but where V is searched in the reverse order determined by π_s. Thus, $O(n)$ violated 2-partition inequalities may be generated.

Using the transformation in (2.27) and (2.28), any k-partition matrix $Y \in \mathcal{Y}(n, k)$ must satisfy the following, for any disjoint subsets $S, T \subseteq [n]$ with $s \coloneqq |S|$ and $t \coloneqq |T|$:

$$\sum_{(i,j) \in S \times T} Y_{ij} - \sum_{\{i,j\} \in \binom{S}{2} \cup \binom{T}{2}} Y_{ij} \leq \tfrac{k}{k-1} \min\{s, t\} + \tfrac{-1}{k-1} \left(st - \binom{s}{2} - \binom{t}{2} \right), \qquad (2.49)$$

the right-hand side of which simplifies to

$$\min\{s,t\} + \frac{1}{k-1}\binom{|s-t|}{2}$$

where, by convention, $\binom{i}{j} \coloneqq 0$ whenever $i < j$.

To determine the strength of an SDP relaxation of MKC solved without the 2-partition inequalities, consider any two disjoint sets $S, T \subseteq [n]$, and define a vector $b \in \mathbb{R}^n$ so that $b_i = 1$ for $i \in S$, $b_i = -1$ for $i \in T$, and $b_i = 0$ for $i \in [n] \setminus (S \cup T)$. Substituting b into inequality (2.45), and reversing the inequality by multiplying both sides by -1, we have the following inequality satisfied by all $x \in \theta(n,k)$:

$$\sum_{(i,j)\in S\times T} x_{ij} - \sum_{\{i,j\}\in\binom{S}{2}\cup\binom{T}{2}} x_{ij} \leq \tfrac{-(k-1)}{k}\left(\frac{-(s+t)}{2}\right) + \tfrac{-1}{k}\left(st - \binom{s}{2} - \binom{t}{2}\right), \quad (2.50)$$

where the right-hand side simplifies to

$$\min\{s,t\} + \tfrac{1}{2k}|t-s|\Big(k - |t-s|\Big).$$

The bound here is tight under particular conditions, which are reported in the following proposition.

Proposition 2.3.5 (Prop. 7, Eisenblätter [31])**.** *For any integers k and n satisfying $4 \leq k \leq n$, and any disjoint subsets $S, T \subseteq [n]$ with $s \coloneqq |S|$, $t \coloneqq |T|$, and $s \leq t$, every $x \in \theta(n,k)$ must satisfy*

$$\sum_{(i,j)\in S\times T} x_{ij} - \sum_{\{i,j\}\in\binom{S}{2}\cup\binom{T}{2}} x_{ij} \leq \min\{s,t\} + \tfrac{1}{2k}|t-s|\Big(k - |t-s|\Big).$$

The bound is tight if either

1. *$s = 1$ and $t \geq k-1$, or*
2. *$s \geq 2$, $s + t \leq k$, either*
   - *$t \leq s^2$, or*
   - *$t > s^2$ and $k \leq \frac{t^2-s^2}{t-s^2}$.* □

We see now that any $x \in \theta(n,k)$ violates any 2-partition inequality by at most $\frac{k}{8}$, which by the above proposition is attained when $|t-s|(k-|t-s|)$ is maximized,

i.e. whenever $|t-s| \in \left\{\left\lfloor \frac{k}{2} \right\rfloor, \left\lceil \frac{k}{2} \right\rceil\right\}$.[5] Furthermore, whenever $|t-s| = 0$ or $|t-s| \geq k$, the corresponding 2-partition inequality is already satisfied, and in the case $|t-s| > k$, the satisfaction is always strict. This implies that for any SDP-based cutting plane algorithm with matrix variables in the $\left\{\frac{-1}{k-1}, 1\right\}$ setting, it suffices to only generate 2-partition inequalities with $1 \leq |t-s| \leq k-1$ for $k \geq 2$, where the potentially "deepest" cuts are those for which $\left\lfloor \frac{k}{2} \right\rfloor \leq |t-s| \leq \left\lceil \frac{k}{2} \right\rceil$. This new result is recorded in the following proposition.

Proposition 2.3.6. *Let n and k be integers satisfying $2 \leq k \leq n$, Y be a matrix in $\mathcal{E}(n,k)$, and $S, T \subseteq [n]$ be disjoint subsets with $s := |S|$, $t := |T|$. If the 2-partition inequality* (2.49) *with respect to S and T is violated by Y, then $1 \leq |s-t| \leq k-1$* □

For the special case where $s = 1$ and $2 \leq t \leq k$, the 2-partition inequalities are potentially violated most when $\left\lfloor \frac{k}{2} \right\rfloor + 1 \leq t \leq \left\lceil \frac{k}{2} \right\rceil + 1$. The next proposition tightens this bound, and immediately yields Proposition (2.3.2), for the case $s = 1$, and $t = 2$.

Proposition 2.3.7 (Prop. 8, Eisenblätter [31])**.** *For any integers k and n satisfying $4 \leq k \leq n$, and any disjoint subsets $S, T \subseteq [n]$ with $|S| = 1$, $t := |T|$, and $2 \leq t \leq k-2$, every $x \in \theta(n,k)$ must satisfy*

$$\sum_{(i,j) \in S \times T} x_{ij} - \sum_{\{i,j\} \in \binom{S}{2} \cup \binom{T}{2}} x_{ij} \leq 1 + \frac{\sqrt{t(k-t)(k-1)} - (k-t)}{k} \quad < \sqrt{t}.$$

Furthermore, this bound is tight. □

2.3.3 Lower and upper clique inequalities

In [65], lower and upper clique inequalities are discussed; the former generalizes the clique inequalities (2.46). For any two integers n and k, a *lower clique inequality* with respect to a set $K \subseteq [n]$ of size $c := |K| \geq k+1$, is of the form:

$$\sum_{\{i,j\} \in \binom{K}{2}} x_{ij} \geq \binom{q}{2} k + qr, \tag{2.51}$$

[5] In fact, $k/8$ is the largest violation possible for the class of so-called hypermetric inequalities, which are known to contain the triangle, clique, and 2-partition inequalities [31].

where $q \coloneqq \left\lfloor \frac{c}{k} \right\rfloor$ and $r \coloneqq c \mod k$. Notice that if k divides c, *i.e.* $r = 0$, then the right-hand simplifies to $k\binom{c/k}{2} = \frac{1}{2k}c(c-k)$. To see that this inequality holds for any k-partition $x \in \mathcal{P}_{\leq \mathrm{KP}}(n,k)$, recall Turán's theorem (2.3.1), which yields

$$|\mathrm{supp}(x)| \geq \binom{q}{2}k + qr \quad \text{for } x \in \mathcal{P}_{\leq \mathrm{KP}}(n,k).$$

It can now be seen that we have a generalized version of the original clique inequalities (2.46). Indeed, when the set $K \subseteq [n]$ is of size $c \coloneqq |K| = k+1$, we have $c = qk + r$ with $q = 0$ and $r = 1$, which yields a simplified right-hand side value in (2.51) of 1. Thus the $\mathcal{NP}$-hardness of the separation of original clique inequalities immediately implies the $\mathcal{NP}$-hardness of separating the upper clique inequalities. Note that Algorithm 1, which heuristically attempts to separate clique inequalities, can be altered to handle the more general inequalities: simply stop the 'while' loop after c iterations, where we can now have $c \geq k+1$. Lower clique inequalities are facet-defining for **simple graph partition** [20] polytope whenever $1 \leq r < k$, and for **size constrained clique partition** [75] polytope whenever $r > 1$, or $r = 1$ and $c = n$.

The lower inequalities are both valid for any capacitated k-partition vector $x \in \mathcal{P}_{\leq \mathrm{KP}}(n,k)$, and can be transformed using (2.27) and (2.28) to a valid inequality of $\mathcal{Y}(n,k)$ as follows. Let $K \subseteq [n]$ be a subset of size $c \coloneqq |K| \geq k+1$, and let $q \coloneqq \left\lfloor \frac{c}{k} \right\rfloor$ and $r \coloneqq c \mod k$. Now we have

$$\begin{aligned} \sum_{\{i,j\} \in \binom{K}{2}} Y_{ij} &\geq \tfrac{k}{k-1}\left(\binom{q}{2}k + qr\right) + \tfrac{-1}{k-1}\binom{c}{2} \\ &= \frac{-c}{2} + \frac{r(k-r)}{2(k-1)}, \end{aligned} \tag{2.52}$$

for any $Y \in \mathcal{Y}(n,k)$. The strength of such constraints in the $\{0,1\}$ setting is found by choosing a vector $b \in \mathbb{R}^n$ with $b_i = 1$ for $i \in K$ and $b_i = 0$ for $i \in [n] \backslash K$, and simplifying the expression in (2.45) to:

$$\sum_{\{i,j\} \in \binom{K}{2}} x_{ij} \geq \tfrac{k-1}{k}\left(\tfrac{-1}{2}c\right) + \tfrac{1}{k}\binom{c}{2}. \tag{2.53}$$

The above right-hand side simplifies further to

$$\frac{1}{2k}c(c-k),$$

and is tight by the following proposition (which also implies the claimed strength of the original clique inequality (2.46)).

Proposition 2.3.8 (Prop. 6, [31])**.** *For any integers k and n satisfying $3 \leq k \leq n$, and subset $K \subseteq [n]$, with $c \coloneqq |K| > k$, every $x \in \theta(n,k)$ must satisfy*

$$\sum_{\{i,j\} \in \binom{K}{2}} x_{ij} \geq \frac{1}{2k} c(c-k).$$

This bound is tight. □

Now we rewrite the right-hand side, $\binom{q}{2}k + qr$, of the valid lower clique inequality (2.51) of $\mathcal{P}_{\mathrm{MKC}}(n,k)$, in the form

$$B \coloneqq \tfrac{1}{2k} c(c-k) - \xi$$

for some ξ, where we recall $q = \left\lfloor \frac{c}{k} \right\rfloor$ and $qk = c - r$. The expression B now simplifies to

$$B = \tfrac{1}{2k}(c-r)(c-r-k),$$

which, after simple algebra, yields

$$\xi = \tfrac{1}{2k} r(2c-k-r).$$

The maximum value of ξ denotes the largest absolute amount any $x \in \theta(n,k)$ can violate a lower clique inequality (2.51). Clearly, $r = 0$ implies that every lower clique inequality with corresponding clique K of size divisible by k is satisfied by every $Y \in \mathcal{E}(n,k)$. Now, for any K with $c = |K| > k$, we must have $r = c \mod k < k$, and so the value of ξ, is maximized when r is as large as possible, *i.e.* $r = k-1$. When fixing $r = k-1$, the value of ξ is maximized for the largest size of a clique $c \leq n$ with $c \mod k = k-1$. Thus the largest potential violation of the inequality (2.51) for any $Y \in \mathcal{E}(n,k)$ occurs when K has size $c = qk + k - 1 = (q+1)k - 1$, for the largest possible integer q, *i.e.* $q + 1 = \left\lfloor \frac{n+1}{k} \right\rfloor$, which yields $c = \left\lfloor \frac{n+1}{k} \right\rfloor k - 1$.

Proposition 2.3.9. *For any two integers k and n satisfying $4 \leq k \leq n$, matrix $Y \in \mathcal{E}(n,k)$, and subset $K \subseteq [n]$, if the corresponding lower clique inequality* (2.52) *is violated by Y, then $1 \leq |K| \mod k \leq k-1$.* □

For any instance $(n, k, \alpha, \beta, S, W)$ of CKP, with $S = \emptyset$, and for any subset $K \subseteq [n]$, the *upper clique inequalities* defined in [65] encode the fact that any capacitated k-partition $(V_1, \ldots, V_k)$ must have an upper bound on the total number of pairs $\{i, j\} \in \binom{[n]}{2}$ contained in any subset $V_h \cap K$, $h \in [k]$. Let us define

$$\phi_i \coloneqq \max\Big\{\phi \in \mathbb{Z}_+ \,\Big|\, \phi \leq \beta, \tfrac{R_i - \phi}{k-i} \geq \alpha\Big\},$$

where $\phi_0 \coloneqq 0$, $R_0 \coloneqq n$ and $R_i \coloneqq R_{i-1} - \phi_i$. Here ϕ_i denotes the ith largest part in capacitated k-partition of $[n]$. Indeed, this recursive procedure is correct, since $\frac{n}{k} \geq \alpha$ if and only if $\frac{n-\phi}{k-1} \geq \alpha$ for some $\phi \leq \beta$, which implies $\alpha \leq \phi_i \leq \beta$, for $i \in [k]$, and $\sum_{i \in [k]} \phi_i = n$. Further, $(\phi_i : i \in [k])$ must be lexicographically maximum, *i.e.* for any capacitated k-partition $(\phi'_i : i \in [k])$, the smallest $\ell \in [k]$ for which $\phi_\ell \neq \phi'_\ell$ must yield $\phi_\ell > \phi'_\ell$, and so ϕ_ℓ must be the ℓth largest part in any capacitated k-partition of $[n]$. Thus, for any $Q \subseteq [n]$, if we let $k_Q \coloneqq \max\Big\{i \in [k] \,\Big|\, \sum_{j \leq i} \phi_i \leq |Q|\Big\}$ and $n_Q \coloneqq |Q| - \sum_{i \in [k_Q]} \phi_i$, we have the following valid inequality for $\mathcal{P}_{\mathrm{CKP}}(n, k, \alpha, \beta, \emptyset)$:

$$\sum_{\{i,j\} \in \binom{Q}{2}} x_{ij} \leq \sum_{i \in [k_Q]} \binom{\phi_i}{2} + \binom{n_Q}{2}. \tag{2.54}$$

Since it can be heuristically argued, using Turán's Theorem 2.3.1, that the objective function of **capacitated k-partition**—$\sum_{\{i,j\} \in \binom{[n]}{2}} W_{ij} x_{ij}$—attains small values at r-partition vectors for large r, and which are "balanced", *i.e.* each part is of size $\lfloor \frac{n}{r} \rfloor$ or $\lceil \frac{n}{r} \rceil$, we see that the lower clique inequalities (2.51) which give lower bounds on $\sum_{\{i,j\} \in \binom{[n]}{2}} X_{ij}$ are more likely to be violated by an optimal fractional $x \in \mathcal{P}_{\leq \mathrm{KP}}(n, k)$ than the above upper bounds. Furthermore, we expect the upper clique inequalities to be $\mathcal{NP}$-hard to separate.

Transforming the inequality (2.54) by using (2.28), we have the following inequality which is valid for any $Y \in \mathcal{Y}(n, k, \alpha, \beta, S)$ and $Q \subseteq [n]$ with $(\phi_i : i \in [k])$ defined as above:

$$\sum_{\{i,j\} \in \binom{Q}{2}} x_{ij} \leq \tfrac{k}{k-1} \left(\sum_{i \in [k_Q]} \binom{\phi_i}{2} + \binom{n_Q}{2} \right) + \tfrac{-1}{k-1} \binom{|Q|}{2}. \tag{2.55}$$

We note here that we cannot prove the strength of $\mathcal{E}(n, k)$ with respect to this last inequality—as well as those discussed in the remaining subsections—using the valid

inequality of the form (2.45) for $\theta(n,k)$. It would be interesting to explore this area further.

2.3.4 Generalized inequalities for node-weighted case

So far we have only discussed valid inequalities for polyhedra whose corresponding partitions were not weighted, in the sense that Ferreira *et al.* [32, 33] considered when studying the polytope of the **node capacitated graph partition**, denoted $\mathcal{P}_{\mathrm{NCGP}}(n,\beta,a) \coloneqq \mathcal{P}(n,1,n,1,\beta,\emptyset,a)$—recall that $a=(a_i : i\in[n])$ is a vector of positive, integral vertex weights. The problem asks for a maximum-weight r-partition $\mathcal{V}$ of $[n]$, for any $r\in[n]$, with each part $P\in\mathcal{V}$ bounded above in weight, *i.e.* $\sum_{i\in P} a_i \leq \beta$. This generalizes the **simple graph partition** problem studied by Sørensen [92], where vertex weights are restricted to be 1, *i.e.* the unweighted case. We present a few valid inequalities of [32], since it can be seen that

$$\mathcal{P}_{\mathrm{CKP}}(n,k,\alpha,\beta,S) \subseteq \mathcal{P}_{\mathrm{NCGP}}(n,\beta,e),$$

for any input (n,k,α,β,S,W) to **capacitated k-partition**. Furthermore, as we will see we in the following chapter (Chapter 3), our algorithm contains a subroutine where solving a weighted version of **capacitated k-partition**—known as **weight capacitated k-partition**—is necessary. Thus we define the corresponding polyhedron as $\mathcal{P}_{\mathrm{WCKP}}(n,k,\alpha,\beta,S,a)$, where $a=(a_i : i\in[n])$ encodes the vertex weights of $[n]$, and noting that

$$\mathcal{P}_{\mathrm{CKP}}(n,k,\alpha,\beta,S) = \mathcal{P}_{\mathrm{WCKP}}(n,k,\alpha,\beta,S,e).$$

The inequalities of [32] are displayed below in their original form, where the variables z_{ij} for $\{i,j\}\in\binom{[n]}{2}$ are complementary to k-partition incidence vectors $x\in\mathcal{P}_{\leq\mathrm{KP}}(n,k)$, *i.e.* for any $i,j\in[n]$, $z_{ij}=1$ if and only if vertices i and j are separated by the corresonding k-partition; denote z a *k-cut* incidence vector. Nevertheless, we can transform any valid inequality of $\mathcal{P}_{\mathrm{NCGP}}(n,\beta,a)$ into one valid for $\mathcal{P}_{\mathrm{WCKP}}(n,k,\alpha,\beta,S,a)$, by applying the unique linear function $f\colon\mathbb{R}\to\mathbb{R}$ with $f(1)=0$ and $f(\frac{-1}{k-1})=1$. Here,

we have $f(x) \coloneqq \frac{k-1}{k} - \frac{k-1}{k}x$. Thus, for any k-cut vector z satisfies

$$\sum_{\{i,j\} \in \binom{[n]}{2}} A_{ij} z_{ij} \leq b, \tag{2.56}$$

if and only if the unique k-partition matrix $Y \in \mathcal{Y}(n,k)$ such that $z_{ij} = f(Y_{ij})$, for each $\{i,j\} \in \binom{[n]}{2}$, satisfies:

$$\begin{aligned} & \sum_{\{i,j\} \in \binom{[n]}{2}} A_{ij} f(Y_{ij}) \leq b, \\ \iff & \sum_{\{i,j\} \in \binom{[n]}{2}} A_{ij} \left(\tfrac{k-1}{k} - \tfrac{k-1}{k} Y_{ij} \right) \leq b, \\ \iff & \sum_{\{i,j\} \in \binom{[n]}{2}} A_{ij} Y_{ij} \geq \sum_{\{i,j\} \in \binom{[n]}{2}} A_{ij} - \tfrac{k}{k-1} b \end{aligned} \tag{2.57}$$

The following are valid inequalities for $\mathcal{P}_{\mathrm{NCGP}}(n,\beta,a)$—and thus are valid for the contained polytope $\mathcal{P}_{\mathrm{WCKP}}(n,k,\alpha,\beta,S,a)$. First, note that a sufficient condition for full-dimensionality of the NCGP is that $n < k\alpha - 1$. The paper [32] also presents mild sufficient conditions in which all the inequalities presented below are facet-defining.

Let n and β be integers satisfying $1 \leq \beta \leq n$, and $a = (a_i : i \in [n])$ a positive, integral vector denoting weights of a vertex set $V \coloneqq [n]$. For any positive integer q, define a *q-cover* of V to be a set $C \subseteq V$ such that $\sum_{i \in C} a_i > q\beta$. In other words, by the pigeon-hole principle, any partition of V into parts with weights at most β, is such that the vertices of a q-cover must be split into at least $q+1$ parts. Further denote a q-cover C as *minimal* if for each $j \in C$, we have $\sum_{i \in C \setminus \{j\}} a_i \leq q\beta$, *i.e.* if the removal of any vertex from the q-cover causes the remaining set to not have the q-cover property.

For any q-cover C of V and subset $E_C \subseteq \binom{C}{2}$, if the graph (C, E_C) defines a tree, then

$$\sum_{\{i,j\} \in E_C} z_{ij} \geq q, \tag{2.58}$$

is valid, whereas if the graph (C, E_C) is a cycle or an ear decomposition (see [101]), then the following inequality is valid:

$$\sum_{\{i,j\} \in E_C} z_{ij} \geq q + 1. \tag{2.59}$$

To see the first inequality (2.58), note that any partition of C into $q+1$ or more parts must cut at least q edges of any connected graph over C—where the latter (2.59) follows since each part of a partition must cut at least two edges of C.

A strengthening of inequality (2.58) is also found in [32]: whenever $C \subseteq V$ is a 1-cover and the tree (C, E_C) is a *star*, *i.e.* all $|E_C| = |C| - 1$ edges of the tree are incident upon a single vertex $r \in C$, called the *root*, the following inequality holds for any k-cut vector z:

$$\sum_{\{i,j\} \in E_C} z_{ij} \geq |E_C| - \max \left\{ |U| \;\middle|\; U \subseteq C \backslash \{r\}, \sum_{i \in U} a_i + a_r \leq \beta \right\} \tag{2.60}$$

Furthermore, given a q-cover, the cycle inequalities above can be strengthened by the so-called cycle-with-tails inequality; see [32] for details. We omit the remaining facet-defining inequalities of Ferreira *et al.* due to their rarely being violated on our "side" of the polyhedron, *i.e.* the transformations of the above inequalities yield upper bound on non-negative combinations of non-negative variables which appear in a minimization problem, and hence tend to be small in value in relaxation solutions.

The inequalities (2.58), (2.59), and (2.60), can be transformed using (2.56) and (2.57) into valid inequalities with respect to k-partition matrices $Y \in \left\{ \frac{-1}{k-1}, 1 \right\}^{n \times n}$, as follows. First let $C \subseteq [n]$ be a q-cover of $[n]$ with given vertex weight vector $a = (a_i : i \in [n]) \in \mathbb{Z}_+^n$. For any subset of edges E_C so that (C, E_C) is a tree on C, Y must satisfy:

$$\sum_{\{i,j\} \in E_C} Y_{ij} \leq |E_C| - \tfrac{k}{k-1} q, \tag{2.61}$$

whereas if E_C is such that (C, E_C) is a cycle of C, then

$$\sum_{\{i,j\} \in E_C} Y_{ij} \leq |E_C| - \tfrac{k}{k-1} (q+1), \tag{2.62}$$

must hold. Notice that the right-hand side of former inequality simplifies to

$$|C| - \tfrac{k(q+1)}{k-1} + \tfrac{1}{k-1},$$

since $|E_C| = |C| - 1$, while the right-hand side of the latter inequality becomes

$$|C| - \tfrac{k(q+1)}{k-1},$$

since $|E_C| = |C|$.

Now suppose C is a 1-cover, and E_C is a set of edges so that (C, E_C) is a star of C with root $r \in C$. Letting $n_r \coloneqq \max\{|U| \,|\, U \subseteq C \backslash \{r\}, \sum_{i \in U} a_i + a_r \leq \beta\}$, we can now transform (2.60) into:

$$\sum_{\{i,j\} \in E_C} Y_{ij} \leq |E_C| - \tfrac{k}{k-1} \left(|E_C| - n_r\right), \tag{2.63}$$

where the right-hand side can be simplified to

$$\tfrac{k}{k-1} n_r + \tfrac{-1}{k-1} \left(|C| - 1\right),$$

since $|E_C| = |C| - 1$.

The upper capacity constraints $\sum_{j \in [n] \backslash \{i\}} a_j x_{ij} \leq \beta$, for each $i \in [n]$, of our polyhedron $\mathcal{P}_{\mathrm{CKP}}(n, k, \alpha, \beta, S)$ are in the form of a knapsack constraint, and we saw above that knapsack-like valid inequalities arose. Since the knapsack polytope is well-studied (see *e.g.* [10, 100]), and our variables $x_{ij} \in \{0, 1\}$, we transform the lower capacity constraints $\sum_{j \in [n] \backslash \{i\}} a_j x_{ij} \geq \alpha$, for each $i \in [n]$, into the alternate form:

$$\sum_{j \in [n] \backslash \{i\}} a_j z_{ij} \leq m - \alpha, \tag{2.64}$$

where $z_{ij} \coloneqq 1 - x_{ij}$ for each $\{i, j\} \in \binom{[n]}{2}$, and $m \coloneqq \sum_{j \in [n]} a_j$.

The knapsack polytope was first characterized by Balas [10] using the set of all strong minimal covers, where in our setting, for a given index $i \in [n]$, a *cover* of the variables in (2.64) is a set $C \subseteq N_i \coloneqq [n] \backslash \{i\}$, such that $\sum_{j \in C} a_j > m - \alpha$. Clearly, for any cover C of N_i, we must have

$$\sum_{j \in C} z_{ij} \leq |C| - 1.$$

In other words, for any cover C, not all edges of $\{\{i, j\} \,|\, j \in C\}$ can be cut. When translated back to variables $x = e - z$, this tells us that at least some entry x_{ij}, $j \in C$, is of value 1. Since $\sum_{i \in [n]} a_i = m$, this statement must be true for each $i \in [n] \backslash C$, *i.e.*

$$\sum_{j \in C} x_{ij} \geq 1, \qquad \text{for } i \in [n] \backslash C \tag{2.65}$$

is a valid inequality of $\mathcal{P}_{\mathrm{WCKP}}(n, k, \alpha, \beta, S, a)$, which we denote as a *star cover inequality*.

A cover C is *minimal* if $\sum_{j\in Q} a_j \leq m-\alpha$ for each proper subset $Q \subsetneq C$. The *extension* of C is denoted $E(C) \coloneqq C \cup C'$ where $C' = \{j \in N_i \backslash C \,|\, a_j \geq \max_{h\in C}\{a_h\}\}$. Thus an extension of a cover C yields the following strengthening of (2.65):

$$\sum_{j\in E(C)} x_{ij} \geq |E(C)|-|C|+1 \quad \text{for } i \in [n]\backslash E(C). \tag{2.66}$$

Denote this inequality as an *extended star cover* inequality. A minimal cover C is said to be *strong* if no other minimal cover C' of the same cardinality $|C|= |C'|$ is such that $E(C') \subseteq E(C)$. Unfortunately, generating all such covers is $\mathcal{NP}$-hard, and there may be exponentially many such covers in the worst case.

A simple heuristic for separating valid star covering inequalities with respect to x^* is as follows. First choose a vertex $v \in [n]$ with $a_v < \alpha$, and let $C \coloneqq [n] \setminus \{v\}$. Then iteratively remove vertices from C until $\sum_{j\in C} a_j > m-\alpha$, and $\sum_{j\in C} a_j - \min_{h\in C}\{a_h\} \leq m-\alpha$. Vertex selection here can either be random, or greedy with respect to the cover (*e.g.* remove from C the index $i \in C$ for which a_i is the maximum weight satisfying $\sum_{j\in C} a_j - a_i > m-\alpha$), or greedy with respect a point x^* we attempt to separate (*e.g.* remove $\ell \in \left\{ i \in C \,\middle|\, \sum_{j\in C} a_j - a_i > m-\alpha \right\}$ which maximizes x^*_{ij}). The former greedy strategy will potentially yield a larger right-hand side of (2.66), whereas the latter greedy strategy is more likely to end with a small sum of the left-hand side of (2.66). Clearly, C will be a minimal cover, and so we construct $E(C)$, and test whether x^* violates any of the $m-|E(C)|$ extended star cover inequalities (2.66), *i.e.* test

$$\sum_{j\in E(C)} x^*_{ij} < |E(C)|-|C|+1 \quad \text{for } i \in [n] \setminus E(C).$$

By the transformation in (2.27) and (2.28), the cover inequalities with respect to a cover $C \subseteq [n]$ of $m-\alpha$, take on the following form, and must be valid for any $Y \in \mathcal{Y}(n,k,\alpha,\beta,S,a)$:

$$\sum_{j\in C} Y_{ij} \geq \frac{k}{k-1} + \frac{-1}{k-1}|C| \quad \text{for } i \in [n] \setminus C, \tag{2.67}$$

where the right-hand side simplifies to

$$1 - \frac{|C|-1}{k-1}.$$

This is strengthened whenever C is minimal, to

$$\sum_{j \in E(C)} Y_{ij} \geq \frac{k}{k-1}\left(|E(C)|-|C|+1\right) + \frac{-1}{k-1}|E(C)| \quad \text{for } i \in R(P), \tag{2.68}$$

with right-hand side simplifying to

$$|E(C)|-|C|+1 - \frac{|C|-1}{k-1}.$$

If $\alpha > 1$, one type of inequality which bypasses the use of the knapsack polyhedron is the following slight generalization of the *flower inequalities* of Ji and Mitchell [57]. Here, we consider the unweighted case first, *i.e.* $a = e$, and take a partition of the set $[n]$ into two parts F and $[n]\backslash F$. The inequalities

$$\sum_{\{i,j\} \in \binom{F}{2}} x_{ij} + \sum_{(i,j) \in F \times [n]\backslash F} x_{ij} \geq \binom{\alpha}{2} q + \binom{r}{2} + r(\alpha - r), \tag{2.69}$$

are valid for each $x \in \mathcal{P}_{\mathrm{CKP}}(n, k, \alpha, \beta, S, e)$, where $q \coloneqq \left\lfloor \frac{|F|}{\alpha} \right\rfloor$ and $r \coloneqq |F| \mod \alpha$. To see this, consider an incidence vector $x \in \{0,1\}^{\binom{n}{2}}$ of a k-partition of $[n]$ in which F is the disjoint union of q cliques of size α, and 1 clique of size r—for this argument we ignore k. The total number of edges here is precisely the right-hand side of (2.69). This value is minimum since any edge removed from within a part in F must be replaced by 2 new edges in $\binom{F}{2} \cup (F \times [n]\backslash F)$, to keep part sizes at least α. Further, any edge removed from the part spanning F and $[n]\times F$ must be replaced by another edge, again, to keep the part size of the resulting partition above α. This bound of valid inequality (2.69) can only be attained when n and the size of $F \subseteq [n]$ permit a "packing" of α-cliques, *i.e.* when $|F| \leq \alpha k_\alpha$ where $k_\alpha \coloneqq \max\{i \in [k] \mid \phi_{k-i+1} = \alpha\}$, and $(\phi_i : i \in [k])$ is as in the previous section—ϕ_{k-i+1} here is the ith smallest part size in any capacitated k-partition. Note that $k_\alpha \geq 1$ by prop 2.1.3. It would be interesting to tighten the bound whenever $|F| > \alpha k_\alpha$.

A heuristic proposed by Ji and Mitchell for separating the flower inequalities is to partition the set $[n]$ into components, and then check for violation of the corrseponding inequalities. Inuitively, for a given fractional solution $x^* \in [0,1]^{\binom{n}{2}}$, the components of the graph $G = (V, E)$ with $V = [n]$ and $E = \left\{ \{i,j\} \in \binom{[n]}{2} \;\middle|\; x_{ij} > \delta \right\}$, where $\delta \in [0,1]$,

are likely to violate the flower inequality, since the edges between components have $x_{ij} \leq \delta$.

The flower inequalities can be transformed into valid inequalities for $Y \in \mathcal{Y}(n,\, k,\, \alpha,\, \beta,\, \emptyset,\, e)$, via (2.27) and (2.28), for any $F \subseteq [m]$, with:

$$\sum_{\{i,j\} \in \binom{F}{2}} Y_{ij} + \sum_{(i,j) \in F \times [m] \setminus F} Y_{ij} \geq \tfrac{k}{k-1} \left(\binom{\alpha}{2} q + \binom{r}{2} + r(\alpha - r) \right) + \tfrac{-1}{k-1} \left(\binom{c}{2} + c(n-c) \right), \tag{2.70}$$

where $c \coloneqq |F|$, $q \coloneqq \left\lfloor \frac{c}{\alpha} \right\rfloor$, and $r \coloneqq c \mod \alpha$.

It is easy to see that, for any m-partition of $[n]$ with parts of size at most β and having weights a_i, for $i \in [m]$, the flower inequalities imply the following inequality which is valid for all $Y \in \mathcal{Y}(m, k, \alpha, \beta, \emptyset, a)$, and any $F \subseteq [n]$:

$$\sum_{\{i,j\} \in \binom{F}{2}} a_i a_j Y_{ij} + \sum_{(i,j) \in F \times [n] \setminus F} a_i a_j Y_{ij} \geq \tfrac{k}{k-1} \left(\binom{\alpha}{2} q + \binom{r}{2} + r(\alpha - r) - \sum_{i \in [m]} \binom{a_i}{2} \right) + \tfrac{-1}{k-1} \left(\binom{c}{2} + c(n-c) \right), \tag{2.71}$$

where $c \coloneqq \sum_{i \in F} a_i$, $n \coloneqq \sum_{i \in [m]} a_i$, $q \coloneqq \left\lfloor \frac{c}{\alpha} \right\rfloor$, and $r \coloneqq c \mod \alpha$. Indeed, $a_i a_j$ represents the number of pairs between parts i and j of the m-partition, and $\binom{a_i}{2}$ is the number of pairs within part $i \in [m]$. Since $Y_{ii} = 1$, we simply subtract the summand $\sum_{i \in [m]} \binom{a_i}{2}$ from both sides of the original inequality (2.69).

Chapter 3

A branch-and-cut algorithm for ConSP

In this chapter we describe in detail the main components of our proposed branch-and-cut algorithm for solving instances of **capacitated k-partition** (2.2.2), and ultimately, for producing optimal schedules in the **conference scheduling problem** (*cf.* 2.1). First we provide a summary of our algorithm in procedural form.

Let $\mathcal{I} \coloneqq (n, k, \alpha, \beta, D, W)$ be a tuple denoting an arbitrary input to **capacitated k-partition**, where:

- n, k, α, and β are integers satisfying $2 \leq k \leq n$ and $1 \leq \alpha \leq \frac{n}{k} \leq \beta \leq n$;
- $D \subseteq \binom{[n]}{2}$;
- W is an $n \times n$ nonnegative, symmetric matrix—typically $\operatorname{diag}(W) = 0$;

and the optimization problem asks for a minimum-weight capacitated k-partition of vertices $[n]$, *i.e.* a k-partition $(V_1, \ldots, V_k)$ of V satisfying

- $\alpha \leq |V_h| \leq \beta$, for each $h \in [k]$;
- $\binom{V_h}{2} \cap D = \emptyset$, for each $h \in [k]$;
- $\sum_{h \in [k]} \sum_{\{ij\} \in \binom{V_h}{2}} W_{ij}$ is minimum.

A *subproblem* $\mathcal{J}$ of $\mathcal{I}$ is a tuple (S, T, z) encoding a subset of feasible capacitated k-partitions of $\mathcal{I}$. Here $S, T \subseteq \binom{[n]}{2}$ are disjoint subsets of pairs of vertices which are to be separated and grouped together, respectively, in any capacitated k-partition of $\mathcal{I}$ found in $\mathcal{J}$, and $z \in \mathbb{R}$ denotes a lower bound on the total weight of any such feasible partition. If a subproblem $\mathcal{J}$ encompasses at least one feasible solution, we say it is *feasible* or *non-empty*, otherwise *infeasible* or *empty*. Whenever $\mathcal{J}$ is infeasible, we let

$+\infty$ be a lower bound, by convention. We write $R(\mathcal{I}, \mathcal{J})$ as a formal instance (to be discussed) to a relaxation of an optimization problem which encodes the solution set of $\mathcal{J}$ with respect to $\mathcal{I}$—thus $\mathcal{R}(\mathcal{I}, \mathcal{J})$ provides lower bounds for $\mathcal{J}$. The algorithm, hereafter denoted as *semidefinite branch-and-cut* or SBC, can be summarized as follows, with details provided in the sequel.

0. *Initialization.* Initialize *global solution variables* $\mathcal{V}_{\text{ub}}$ and z_{ub} to be any *a priori* capacitated k-partition and its cost, respectively; if none are known, then assign $\mathcal{V}_{\text{ub}} \coloneqq \emptyset$ and $z_{\text{ub}} \coloneqq \infty$. Initialize the *pool* $\mathfrak{P} \coloneqq \{\mathcal{J}^*\}$ with the *root* subproblem $\mathcal{J}^* = (\emptyset, \emptyset, -\infty)$ of $\mathcal{I}$. Go to Step 1.

1. *Node selection.* If $\mathfrak{P} = \emptyset$, exit and output $\mathcal{V}_{\text{ub}}$ and z_{ub}; otherwise, select a subproblem $\mathcal{J} \coloneqq (S, T, z) \in \mathfrak{P}$ which minimizes a real-valued function $f(z)$ (*e.g.* $f(z) \coloneqq z$) and remove it from the pool, *i.e.* $\mathfrak{P} \coloneqq \mathfrak{P} \backslash \{\mathcal{J}\}$.

2. *Preprocessing.* If $z \geq z_{\text{ub}}$ or $\mathcal{J}$ is proved to be infeasible, then go back to Step 1; otherwise, go to Step 3.

3. *Lower bounding.* Solve $R(\mathcal{I}, \mathcal{J})$, producing a solution $\mathcal{V}^*$ with value z^*. If $z^* \geq z_{\text{ub}}$, go back to Step 1; otherwise test if $\mathcal{V}^*$ encodes a feasible solution to $\mathcal{J}$: if so, update $\mathcal{V}_{\text{ub}} \coloneqq \mathcal{V}^*$ and $z_{\text{ub}} \coloneqq z^*$, and go back to Step 1; otherwise update $z \coloneqq z^*$, and go to Step 4.

4. *Upper bounding.* Attempt to generate a *global* feasible solution to $\mathcal{I}$, or a *local* solution to subproblem $\mathcal{J}$. If a solution $\mathcal{V}^*$ of value $z^* < z_{\text{ub}}$ is found, update $\mathcal{V}_{\text{ub}} \coloneqq \mathcal{V}^*$ and $z_{\text{ub}} \coloneqq z^*$, and if further $z^* \leq z$, then go back to Step 1. Otherwise, go to Step 5.

5. *Branching.* Find a pair of vertices $e \in \binom{[n]}{2}$ to *branch* on, *i.e.* a pair $e \notin S \cup T$, which is used to form two subproblems $\mathcal{J}_0 \coloneqq (S \cup \{e\}, T, z)$ and $\mathcal{J}_1 \coloneqq (S, T \cup \{e\}, z)$, and finally update the pool $\mathfrak{P} \coloneqq \mathfrak{P} \cup \{\mathcal{J}_0, \mathcal{J}_1\}$. Go back to Step 1.

3.1 Subproblems

In this section we discuss an alternate representation of the subproblems of our algorithm, and how this affects the modeling of the optimization problem solved at each subproblem.

3.1.1 Representation

Suppose $\mathcal{I} \coloneqq (n,\, k,\, \alpha,\, \beta,\, D,\, W)$ is some input to our SBC algorithm, and let $\mathcal{J} \coloneqq (S, T, z) \in \mathfrak{P}$ be an arbitrary subproblem of $\mathcal{I}$. Recall that the set T represents the subset of pairs $i, j \in [n]$ which must be grouped together in any feasible solution found in $\mathcal{J}$. Since grouping pairs is a transitive relation, T must form an m-partition of $[n]$ with $k \leq m \leq n$, where each part is a *connected component* of T, *i.e.* a maximally connected subset of vertices in T. Thus, we can represent T by an *m-partition map* τ of $[n]$, *i.e.* an map $\tau\colon [m] \mapsto 2^{[n]}$, satisfying $\tau(i) \cap \tau(j) = \emptyset$ for distinct $i, j \in [m]$ and $\cup_{i\in[m]}\tau(i) = [n]$. For any $i \in [m]$, we call $\tau(i) \subseteq [n]$ the ith τ *component* or *part* of the m-partition $(\tau(1), \ldots, \tau(m))$ of $[n]$.

For a given m-partition map τ and subset $S \subseteq \binom{[n]}{2}$, we can *contract* S with respect to τ, *i.e.* replace $S \subseteq \binom{[n]}{2}$ with $S' \subseteq \binom{[m]}{2}$, where $\{i, j\} \in S'$ if and only if $S \cap (\tau(i) \times \tau(j)) \neq \emptyset$. In other words, we treat each of the m τ-components as vertices, and connect vertex pairs in S' whenever the corresponding τ-components are adjacent in S. We can also form S' by defining a *vertex placement* map $\rho\colon [n] \to [m]$, where $\rho(v)$ is the unique index $h \in [m]$ with $v \in \tau(h)$. In this setting, $\{\rho(u), \rho(v)\} \in S'$ only if $\{u, v\} \in S$ and $\rho(u) \neq \rho(v)$.

We now write each subproblem $\mathcal{J}$ as a tuple (m, S, τ, z), where $k \leq m \leq n$, $S \subseteq \binom{[m]}{2}$, τ is an m-partition map of $[n]$, and z is a lower bound on the value of all capacitated k-partitions encompassed by $\mathcal{J}$. Note that the corresponding vertex placement map of τ can be formed in linear time, and is thus not part of the input of $\mathcal{J}$. To ensure that each such subproblem of $\mathcal{I}$ has a unique representation, we choose $\tau\colon [m] \mapsto 2^{[n]}$ so that the ith part $\tau(i)$ has the ith smallest *representative* $\mathrm{rep}(\tau(i))$, where we define $\mathrm{rep}(\tau(i)) \coloneqq \min\{v \in [n] \,|\, v \in \tau(i)\}$, and write $\mathrm{rep}(i) \coloneqq \mathrm{rep}(\tau(i))$ whenever

τ is clear from context. In the next section, we show how this representation yields a weighted version of the original **capacitated** k**-partition** problem, which may be of smaller size.

3.1.2 Model: a weighted capacitated *k*-partition problem

Let $(m, k, \alpha, \beta, S', W', a)$ be a tuple where m, k, α, and β are integers satisfying $2 \leq k \leq m$ and $1 \leq \alpha \leq \frac{m}{k} \leq \beta \leq m$; $a = (a_i : i \in [k])$ is a positive, integral vector; $S' \subseteq \binom{[m]}{2}$; and W' is an $m \times m$ nonnegative, symmetric matrix—note that $\mathrm{diag}(W)$ is not required to be 0-valued. The **weight capacitated** k**-partition** problem asks for a minimum cost *weight-capacitated k-partition*, *i.e.* a k-partition $\mathcal{V} = (V_1, \ldots, V_k)$ of $[m]$ satisfying:

- $\alpha \leq \sum_{i \in V_h} a_i \leq \beta$ for each $h \in [k]$
- $\binom{V_h}{2} \cap S' = \emptyset$ for each $h \in [k]$
- $\sum_{h \in [k]} \sum_{\{i,j\} \in \binom{V_h}{2}} W'_{ij}$ is minimum

Now it is easy to check that any k-partition matrix $Y \in \left\{\frac{-1}{k-1}, 1\right\}^{m \times m}$ which encodes a feasible weight-capacitated k-partition, or *weight-capacitated k-partition matrix*, must satisfy the following knapsack-like constraint—analogous to the cardinality constraint (2.39) of CKP, where here $n \coloneqq \sum_{i \in [m]} a_i$:

$$\tfrac{k\alpha - n}{k-1} e \leq \sum_{j \in [m]} a_j Y_{ij} \leq \tfrac{k\beta - n}{k-1} \quad \text{for } i \in [m].$$

Thus a model for the weighted **capacitated** k**-partition** problem, given arbitrary input (m, k, α, β, S', W', a), is:

$$\begin{aligned}
\text{(WCKP)} \quad \text{minimize} \quad & \tfrac{k-1}{2k} W' \bullet \left(\tfrac{1}{k-1} J + Y\right) && && (3.1)\\
\text{subject to} \quad & \mathrm{diag}(Y) = e && && (3.2)\\
& Y_{ij} = \tfrac{-1}{k-1} && \text{for } \{i,j\} \in S' && (3.3)\\
& \tfrac{k\alpha - n}{k-1} e \leq Ya \leq \tfrac{k\beta - n}{k-1} e && && (3.4)\\
& Y \in \mathcal{S}^m_+ && && (3.5)\\
& Y \in \left\{\tfrac{-1}{k-1}, 1\right\}^{m \times m}. && && (3.6)
\end{aligned}$$

Now for an input $\mathcal{I} \coloneqq (n, k, \alpha, \beta, D, W)$ to our SBC algorithm—*i.e.* an unweighted **capacitated k-partition** instance—any subproblem $\mathcal{J} \coloneqq (m, S, \tau, z)$ of $\mathcal{I}$, can be bounded by solving a relaxation of the **weight capacitated k-partition** problem with input tuple $\mathcal{R}(\mathcal{I}, \mathcal{J}) \coloneqq (m, k, \alpha, \beta, S', W', a)$, where

- $a = (a_i : i \in [m])$ is defined $a_i \coloneqq |\tau(i)|$, for each $i \in [m]$—note $\sum_{i \in [m]} a_i = n$;
- $S' \subseteq \binom{[m]}{2}$ is defined to be the set $D' \cup S$, where D' is such that $\{i, j\} \in D'$ if and only if $(\tau(i) \times \tau(j)) \cap D \neq \emptyset$; and
- $W' \in \mathbb{R}^{m \times m}$ is the symmetric matrix where $W'_{ij} \coloneqq \sum_{(h,\ell) \in \tau(i) \times \tau(j)} W_{h\ell}$, for each $i, j \in [m]$.

Essentially, we are forcing the weights of our vertices in the instance $\mathcal{R}(\mathcal{I}, \mathcal{J})$ to correspond with the size of each τ component. We also want pairs of vertices $\{i, j\} \in \binom{[m]}{2}$ separated, *i.e.* in the set S', if and only if there exists an adjacency between corresponding parts $\tau(i)$ and $\tau(j)$ in D or the pair $\{i, j\}$ is already in S. Finally, we define W' so that entry W'_{ij} denotes the cost of including the weight edges between $\tau(i)$ and $\tau(j)$ if i and j are grouped together, which can be rewritten as

$$W'_{ij} = \chi_{\tau(i)}^{\top} W \chi_{\tau(j)}, \quad \text{for } i, j \in [m],$$

with $\chi_R \in \{0, 1\}^m$ defined to be the *characteristic vector* of $R \subseteq [m]$, where $\chi_{R,r} = 1$ if and only if $r \in R$. Note that since $W_{hh} = 0$ for each $h \in [m]$, the diagonal entries of W' can be written

$$W'_{ii} = 2 \sum_{\{h,\ell\} \in \binom{\tau(i)}{2}} W_{ij}, \quad \text{for } i \in [m].$$

Thus the total cost $\frac{1}{2} W' \bullet Y'$ of any k-partition $(V_1, \ldots, V_k)$ of $[m]$, represented by $Y' \in \{0, 1\}^{m \times m}$, is the total cost of the corresponding k-partition $(V'_1, \ldots, V'_k)$ of $[n]$ represented by $Y \in \{0, 1\}^{n \times n}$, where $V'_h \coloneqq \bigcup_{i \in V_h} \tau(i)$. To see this, we have the following equalities:

$$\begin{aligned} \tfrac{1}{2} W' \bullet Y' &= \tfrac{1}{2} \sum_{i,j \in [m]} W'_{ij} Y'_{ij} \\ &= \sum_{\{i,j\} \in \binom{[m]}{2}} W'_{ij} Y'_{ij} + \tfrac{1}{2} \sum_{i \in [m]} W'_{ii} \end{aligned}$$

$$
\begin{aligned}
&= \sum_{h\in[k]} \sum_{\{i,j\}\in\binom{V_h}{2}} W'_{ij} + \tfrac{1}{2} \sum_{h\in[k]} \sum_{i\in V_h} W'_{ii} \\
&= \sum_{h\in[k]} \left(\sum_{\{i,j\}\in\binom{V_h}{2}} \sum_{(r,s)\in\tau(i)\times\tau(j)} W_{rs} + \sum_{i\in V_h} \sum_{\{r,s\}\in\binom{\tau(i)}{2}} W_{rs} \right) \\
&= \sum_{h\in[k]} \left(\sum_{\{u,v\}\in\binom{V'_h}{2}} W_{uv} \right) \\
&= \tfrac{1}{2} W \bullet Y.
\end{aligned}
$$

This argument easily translates to the setting in which partition matrices Y have entries in $\left\{\frac{-1}{k-1}, 1\right\}$.

We note that the use of graph contractions was used by Anjos *et al.* [5] in the design of an algorithm similar to our SBC, but for the **minimum k-partition** , or **maximum k-cut** problem. Vertex weights in this case are not needed, since no capacity on original part sizes is required.

3.2 Preprocessing

In section §2.1.2 we showed that determining whether an instance $\mathcal{I}$ of **capacitated k-partition** has a feasible solution is $\mathcal{NP}$-complete, and thus so is the slightly more general weighted version (described in the previous section). This implies that it is hard to decide if a given subproblem $\mathcal{J}$ of $\mathcal{I}$ is empty or not. Since, further, we solve multiple SDP relaxation problems $R(\mathcal{I},\mathcal{J})$ at each $\mathcal{J}$—the bottleneck procedure of our algorithm (see the next section)—we propose a few heuristic tests which are at least as efficient as the SDP relaxation problem, and aim to either find the best solution of $\mathcal{J}$, or determine that none exists. If one is able to accomplish such a task, then no further exploration of $\mathcal{J}$ is needed, and thus we avoid solving multiple SDPs.

Let $\mathcal{J} \coloneqq (m, S, \tau, z)$ be a subproblem of $\mathcal{I} \coloneqq (n, k, \alpha, \beta, D, W)$, and let $\mathcal{R} \coloneqq \mathcal{R}(\mathcal{I},\mathcal{J}) \coloneqq (m, k, \alpha, \beta, S', W', a)$ be the associated **weight capacitated k-partition** instance. Recall that $\mathcal{V}_{\text{ub}}$ and z_{ub} denote the best found global solution and its associated value, respectively. We start preprocessing the input by adding to S' any known *implied separations*, or those pairs $e \in \binom{[m]}{2} \setminus S'$ which can be shown as not appearing together

in any feasible solution of $\mathcal{J}$ which has value better than z_{ub}. We give two simple ways of checking for such adjacencies. For each $\{i,j\} \in \binom{[m]}{2} \setminus S'$:

1. if $a_i + a_j > \beta$, append $\{i,j\}$ to S';
2. if $W'_{ij} + \frac{1}{2}\operatorname{tr}(W') \geq z_{\text{ub}}$, append $\{i,j\}$ to S'.

Here we define the *trace* $\operatorname{tr}(A) \coloneqq \sum_{i\in[m]} A_{ii}$ for any square $m \times m$ matrix A. Indeed, any pair $\{i,j\} \in \binom{[m]}{2}$ having total weight more than the maximum capacity, β, cannot appear together in any feasible solution of $\mathcal{J}$. Furthermore, if $\{i,j\}$ costs more than the residual cost of the current m-partition of $[n]$ represented by τ, that is, $z_{\text{ub}} - \frac{1}{2}\operatorname{tr}(W')$, then grouping i and j cannot yield a feasible solution cheaper than z_{ub}.

Once S' has been updated to include all implied separations, we check whether $m = k+1$. If so, then the best solution of subproblem $\mathcal{J}$ can be found by the following exhaustive search. Note first that the above preprocessing step implies S' must have the following properties:

$$\begin{aligned}
&a_i \leq \beta && \text{for } i \in [m],\\
&a_i + a_j \leq \beta && \text{for } \{i,j\} \in \binom{[m]}{2} \setminus S',\\
&W'_{ij} + \tfrac{1}{2}\operatorname{tr}(W') < z_{\text{ub}}, && \text{for } \{i,j\} \in \binom{[m]}{2} \setminus S',
\end{aligned}$$

and so our search reduces to scanning the $\binom{m}{2} = O(k^2)$ pairs $\{i,j\} \in \binom{[m]}{2} \setminus S'$, and testing whether:

$$\begin{aligned}
&a_h \geq \alpha && \text{for } h \in [m] \setminus \{i,j\},\\
&a_i + a_j \geq \alpha,
\end{aligned}$$

while keeping track of the pair for which W'_{ij} is smallest. Clearly, if any solution is found, then a best solution has been found for $\mathcal{J}$ which must be cheaper than z_{ub}, and we can therefore update the global solution. Otherwise we have proved that no better solution exists in $\mathcal{J}$. In either case, we exit this preprocessing phase and move on to the next step of SBC (Step 3; Lower bounding; see next section). Note that by not branching on such subproblems with $m = k+1$, no subproblem of $\mathcal{I}$ will proceed past the preprocessing step with size $m \leq k+1$.

Now, if $m > k+1$, we continue by attempting to *prune* subproblem $\mathcal{J}$, *i.e.* determine that no feasible solution exists of cost less than z_{ub}, and proceed back to Step 1 of SBC. First, we associate a graph $G \coloneqq ([m], S')$ with the pairs or edges of S'. Since a capacitated k-partition can be viewed as finding a restrictive type of coloring of the edges in G—namely, a k-coloring in which color classes have bounded total vertex weights (see, for the unweighted vertex instance, the reduction in Proposition 2.1.4)—we can prune $\mathcal{J}$ whenever it is determined that we cannot color G with k or fewer colors. Denoting $\chi(G)$ as the minimum number of colors needed to color the edges of G, *i.e.* the *chromatic number*, we see that this sufficient condition for the pruning of $\mathcal{J}$ is equivalent to testing whether or not

$$\chi(G) > k.$$

Since computing $\chi(G)$ is $\mathcal{NP}$-hard in general [40], we recall a few known bounds on $\chi(G)$.

Graph G is clearly not k-colorable if it has a clique of size $k+1$. Furthermore, letting $\overline{G} \coloneqq \left(V(G), \binom{V(G)}{2} \setminus E(G)\right)$ denote the *complement* graph of G, any clique of G is also a *stable set*—a subset of pairwise non-adjacent vertices—in $\overline{G}$. The maximum size of a clique in G is denoted $\omega(G)$, and is known as the *clique number*, while the maximum size of a stable set is G is denoted $\alpha(G)$, and is known as the *stable set number* (see West [101]). Thus we have the following relations between the above graph quantities:

$$\chi(G) \geq \omega(G) = \alpha(\overline{G}).$$

Unfortunately, each quantity is $\mathcal{NP}$-hard to compute, so we look for further bounds on $\alpha(\overline{G})$ and $\omega(G)$.

Applying Turán's Theorem 2.3.1 to G, we have

$$\omega(G) \leq k \quad \Rightarrow \quad |E(G)| \leq \frac{mq(k-1)}{2} + \binom{r}{2}, \tag{3.7}$$

where we define $q \coloneqq \left\lfloor \frac{m}{k} \right\rfloor$ and $r \coloneqq m \mod k$—recall $|V(G)| = m$. Thus, the contrapositive tells us that whenever enough edges are present, *i.e.* $|E(G)| > \frac{mq(k-1)}{2} + \binom{r}{2}$, there exists a subgraph of G, namely a $k+1$-clique, which is too large for k colors. This observation is formalized in the following proposition, where we substitue $m(m-r)(1-1/k)$

for $mq(k-1)$ since $q = qk/k$ and $qk = m - r$.

Proposition 3.2.1. *For any subproblem $\mathcal{J}$ of input $\mathcal{I}$ to* ***capacitated*** *k-**partition**, if $\mathcal{R}(\mathcal{I}, \mathcal{J}) = (m, k, \alpha, \beta, S', W', a)$ is such that*

$$|S'| > \frac{m(m-r)}{2}\left(1 - \frac{1}{k}\right) + \binom{r}{2},$$

where $r \coloneqq m \mod k$, then $\mathcal{J}$ is infeasible. □

The bound in the above proposition is easy to calculate, but perhaps too strong a condition, since Proposition 2.3.1 bounds the right-hand side of (3.7) from below by

$$\frac{m^2}{2}\left(1 - \frac{1}{k}\right),$$

or, roughly, a fraction of $1 - 1/k$ of the total number of possible edges.

A bound dominant to that found in Proposition 3.2.1 is one provided by the random process in the following proposition.

Proposition 3.2.2. *Let G be a graph with $n \coloneqq |V(G)|$, and $\pi : [n] \mapsto V(G)$ be a random permutation of the vertices of G. Consider the iterative process where we initially set $S \coloneqq \emptyset$ and create duplicate graph $G' \coloneqq G$. Then, at the ith step, if vertex $v \coloneqq \pi(i) \in V(G')$, append v to S, and remove both the closed neighborhood $N_{G'}[v]$ from $V(G')$ and all incident edges, i.e. $\bigcup_{u \in N_{G'}[v]} \delta_{G'}(u)$—proceed to step $i+1$. This process yields the property that S is a stable set of G with expected size:*

$$\mathbb{E}[S] = \sum_{v \in V(G)} \frac{1}{d_G(v) + 1},$$

where $d_G(v)$ denotes the degree of v in G.

Proof. S is clearly a stable set in G, since exactly one vertex from any closed neighborhood $N_G[v]$ of $v \in V(G)$ is selected to be in S. This, along with the choice of π, implies that any $u \in N_G[v]$ has equal probability $\frac{1}{d_G(v)+1}$ of being selected to be in S, where $d_G(v)$ denotes the degree, or the size of the open neighborhood $|N_G(v)|$. Thus the expected size of S immediately follows by linearity of expectation. □

Applying the above proposition to the complement of $G = ([m], S')$ yields a similar process in which a clique of G is produced, and is of expected size $\gamma(G)$:

$$\gamma(G) \coloneqq \sum_{v \in V(G)} \frac{1}{n - d_G(v)} \leq \chi(G),$$

since $d_G(v) + d_{\overline{G}}(v) = m - 1$ for each $v \in V(G)$. Thus, if $\gamma(G) > k$—which is easy to test—then there must exist a clique of size $k+1$ in G, proving then that $\chi(G) > k$, and ultimately implying that the corresponding subproblem $\mathcal{J}$ is infeasible. Otherwise, we can run the random procedure multiple times, *e.g.* $O(m)$ times, and prune $\mathcal{J}$ as soon as a stable set S of $\overline{G}$ of size $|S| > k$ is found.

If the previous heuristic checks were not enough to prune the current subproblem $\mathcal{J}$, there is a more computationally expensive, but tighter bound: the Lovász-theta function of graph $\overline{G}$, which is the only known polynomial time computable bound sandwiched between the two $\mathcal{NP}$-hard numbers $\alpha(\overline{G})$ and $\chi(G)$ [72, 63]. The value $\vartheta(G)$ satisfies

$$\chi(G) \geq \vartheta(\overline{G}) \geq \alpha(\overline{G}),$$

and can be computed by solving the following semidefinite program:

$$\vartheta(\overline{G}) := \text{maximize} \;\; J \bullet X \tag{3.8}$$

$$\text{subject to } X_{ij} = 0 \quad \text{for } \{i,j\} \in E(\overline{G}) \tag{3.9}$$

$$\operatorname{tr}(X) = 1 \tag{3.10}$$

$$X \in \mathcal{S}^m_+. \tag{3.11}$$

Thus if $\vartheta(\overline{G}) > k$, then $\chi(G) > k$, and we can prune $\mathcal{J}$. If not, then we proceed to the next step in the main branch-and-bound algorithm. Note that it may not be wise to solve for $\vartheta(G)$ at each subproblem, since for sparse graphs (*e.g.* any ancestor of an initially sparse root), the bound may be weak and the time spent solving an SDP may be non-negligible. For this reason we calculate $\vartheta(G)$ only when the maximum or average degrees are "large enough", *e.g.* $\max_{v \in G} d_G(v) \geq (1 - \epsilon_1)m$, or $\sum_{v \in [m]} d_G(v) = 2|E(G)| \geq (1 - \epsilon_2) \cdot 2\binom{m}{2}$ for small fractional values $\epsilon_1, \epsilon_2 \in (0, 1)$. Intuitively, the more edges present, the more likely there is to be large clique, or at least a large value of $\vartheta(G)$. However, in practice, we observed that the amount of time spent solving for

$\vartheta(G)$ at any subproblem was negligible when compared to the main lower bounding procedure. This is due, in part, to the initial set of pairs $D \subseteq \binom{[n]}{2}$ of $\mathcal{I}$ being sparse, and to the strength of the SDP relaxations which tend to prune subproblems before branching deeply, *i.e.* the subproblem pairs $S' \subseteq \binom{[m]}{2}$, and thus $G = ([m], S')$, remain sparse.

We summarize this subprocedure applied at Step 2 of SBC to a subproblem $\mathcal{J}$ of $\mathcal{I}$ as follows:

1. Add all implied adjacencies S' of $R(\mathcal{I}, \mathcal{J}) = (m, k, \alpha, \beta, S', W', a)$, and go to Step 2.

2. If $m = k+1$, then exhaustively search the $O(k^2)$ possible solutions of $\mathcal{J}$ and update global structures appropriately, and go back to the Step 1 of SBC. Otherwise, let $G \coloneqq ([m], S')$ and go to Step 3 below—note $m > k+1$.

3. If $|S'| > \frac{m(m-r)}{2}\left(1 - \frac{1}{k}\right) + \binom{r}{2}$ (*cf.* Proposition 3.2.1), then go back to Step 1 of SBC; otherwise go to Step 4 below.

4. If $\gamma(G) > k$ (*cf.* Proposition 3.2.2), then go back to Step 1 of SBC; otherwise go to Step 5 below.

5. If a random stable set C of $\overline{G}$ is generated as in Proposition 3.2.2, and is of size $|C| > k$, after at most $O(m)$ tries, then go back to Step 1 of SBC; otherwise, if G is "dense enough", then go to Step 6 below; else go back to Step 1 of SBC.

6. If $\vartheta(\overline{G}) > k$ (*cf.* 3.8–3.11), the go back to Step 1 of main algorithm; otherwise continue to Step 3 in SBC (described in next section).

3.3 Lower bounding procedure

In this section, we present a detailed explanation of the lower bounding subprocedure of our algorithm (Step 3) which is used to generate strong local bounds to any subproblem $\mathcal{J}$ of an input $\mathcal{I}$ to **capacitated k-partition**. In particular, we first describe the semidefinite programming, or SDP, relaxation of the **weight capacitated**

k-**partition** problem, which takes input $R(\mathcal{I}, \mathcal{J}) \coloneqq (m, k, \alpha, \beta, S', W', a)$ (see §3.1.2). This is followed by a discussion of how to strengthen the relaxation with valid linear inequalities.

We choose to solve the SDP-relaxation of WCKP (3.1)–(3.6), as opposed to the corresponding LP relaxation—obtained by simply removing the semidefinite restriction on variables—since we have observed the latter relaxation to be much weaker in our tests, *i.e.* the number of subproblems explored was much larger. Eisenblätter [31] presents compelling evidence for choosing the $\left\{\frac{-1}{k-1}, 1\right\}$ setting over the $\{0, 1\}$ for solving such SDP relaxation; this is described in some detail in §2.3. Furthermore, SDPs such as ours can be solved in polynomial time using potential reduction methods [4] and primal-dual interior-point methods [53] (see also Vandenberghe and Boyd [96] for more information on SDP, and Nesterov *et al.* [77] for more on general convex programming interior-point methods).

Unfortunately, however, available SDP software tends to be slow when compared to standard LP solvers. Notable slowdown can occur in a setting such as ours—branch-and-bound with SDP relaxations solved at each subproblem—due to the general lack of warm-starting abilities in SDP solvers, as opposed to the non-polynomial-time simplex method [37] which is still widely used in practice in [25, 9] due to efficiencies provided by warm-starting. Another disadvantage present in SDP solvers is the lack of stability due to degeneracy, which can be caused by the problem not satisfying a strict complementarity condition, *e.g.* Slater's condition (see more on strong duality in Ramanal *et al.* [85]). Fortunately, for our particular binary optimization problems, stability is not an issue (see Tunçel [94] and Pataki and Tunçel *et al.* [82]). Our particular implementation details are found in the next chapter (*cf.* §4.1).

3.3.1 Semidefinite programming relaxations

Our lower-bounding procedure for a given subproblem $\mathcal{J}$ of $\mathcal{I}$, where we suppose $R(\mathcal{I}, \mathcal{J}) = (m, k, \alpha, \beta, S', W', a)$, and z is a lower bound on the optimum solution of $\mathcal{J}$, is comprised of solving multiple, progressively stronger SDP relaxations of WCKP. In particular, we attempt to improve upon the bound z by solving the following optimization

problem "SWCKP" iteratively. Here, we let $\mathcal{C}$ be a subset of $\mathcal{C}^* \coloneqq \mathcal{C}(\mathcal{I}, \mathcal{J})$, the set of valid inequalities for all weight-capacitated k-partition matrices $Y \in \mathcal{Y}(m, k, \alpha, \beta, S', a)$, *i.e.* $(A, b) \in \mathcal{C}^*$ if and only if $\sum_{\{i,j\} \in \binom{[m]}{2}} A_{ij} Y_{ij} \leq b$.

$$\text{(SWCKP)} \quad \text{minimize } \tfrac{k-1}{2k} W' \bullet \left(\tfrac{1}{k-1} J + Y \right) \tag{3.12}$$

$$\text{subject to } \operatorname{diag}(Y) = e \tag{3.13}$$

$$Y_{ij} = \tfrac{-1}{k-1} \qquad \text{for } \{i,j\} \in S \tag{3.14}$$

$$\sum_{\{i,j\} \in \binom{[m]}{2}} A_{ij} Y_{ij} \leq b \quad \text{for } (A, b) \in \mathcal{C} \tag{3.15}$$

$$Y \in \mathcal{S}^m_+ \tag{3.16}$$

A summary of the iterative procedure is as follows. First recall that global variables $\mathcal{V}_{\text{ub}}$ and z_{ub} represent the best known capacitated k-partition of $\mathcal{I}$ and its associated objective value, respectively. Also, define a matrix $Y \in \mathbb{R}^{m \times m}$ to be *integral* if it represents a feasible weight-capacitated k-partition of the associated WCKP problem, and otherwise it is *fractional*. Further, we define $\mathcal{D}$ to be a set of triples (A, b, δ), where $(A, b) \in \mathcal{C}^*$ is a valid inequality which has not been violated or *binding*—satisfied with equality—in the last δ consecutive iterations, *e.g.* $(A, b, 2) \in \mathcal{D}$ tells us that the inequality $\sum_{\{i,j\} \in \binom{[m]}{2}} A_{ij} Y_{ij} \leq b$ is valid, and was *non-binding*—satisfied with strict inequality—with repect to either of the last two iterations' corresponding optimum matrices.

0. *Initialization.* Choose $\epsilon \in (0, 1)$ and $c_{\min}, c_{\max}, d_{\max}, t_{\max} \in \mathbb{Z}_+$, and define $\mathcal{C}_0, \mathcal{D}_0, \mathcal{D} \coloneqq \emptyset$, and $z_0 \coloneqq -\infty$. Set counter $t \coloneqq 0$, and go to Step 1.

1. *Solving SDP.* Solve SWCKP with given valid inequalities $\mathcal{C} \subseteq \mathcal{C}(\mathcal{I}, \mathcal{J})$, letting Y^* be the solution and z^* its cost. If $z^* \geq z_{\text{ub}}$, exit and go back to Step 1 of SBC; otherwise, check if Y^* is integral. If so, update $z_{\text{ub}} \coloneqq z^*$ and $\mathcal{V}_{\text{ub}}$ appropriately, then exit and go back to Step 1 of SBC; if not, then test further if the solution sequence is *tailing off*, *i.e.* if $\frac{z^* - z_0}{\max\{|z_0|, 1\}} \leq \epsilon$, or $t = t_{\max}$. If so, update $z \coloneqq z^*$, then exit and go to next step of SBC (Step 3); otherwise increment $t \coloneqq t + 1$ go to Step 2 below.

2. *Removing non-binding inequalities.* Let $\mathcal{D}_0 \subseteq \mathcal{C}$ be the set of non-binding inequalities with respect to Y^*, *i.e.* inequalities $(A, b) \in \mathcal{C}$ for which

$$\sum_{\{i,j\} \in \binom{[m]}{2}} A_{ij} Y^*_{ij} < b.$$

   Then remove $\mathcal{C} \coloneqq \mathcal{C} \backslash \mathcal{D}_0$, and go to Step 3 below.

3. *Additional cutting planes.* Let $\mathcal{C}_0 \coloneqq \emptyset$, and go to Step 3a.

   (a) *From the pool $\mathcal{D}$.* For each element $(A, b, \delta) \in \mathcal{D}$: if (A, b) is violated by Y^*, then update $\mathcal{C}_0 \coloneqq \mathcal{C}_0 \cup \{(A, b, \delta)\}$ and remove $\mathcal{D} \coloneqq \mathcal{D} \backslash \{(A, b, \delta)\}$; else if (A, b) is binding, then reset $\delta \coloneqq 0$; otherwise (A, b) is non-binding, so we increment $\delta \coloneqq \delta + 1$, and if now $\delta \geq d_{\max}$, then remove $\mathcal{D} \coloneqq \mathcal{D} \backslash \{(A, b, \delta)\}$. After search is complete, update $\mathcal{D} \coloneqq \mathcal{D} \cup \mathcal{D}_0$, and if $|\mathcal{C}_0| \geq \min\{c_{\min}, c_{\max} - |\mathcal{C}|\}$, then update $\mathcal{C} \coloneqq \mathcal{C} \cup \mathcal{C}_0$ and go back to Step 1; otherwise go to Step 3b.

   (b) *Generated (details in §3.3.2).* For each type of valid inequality T in $\mathcal{C}$: find a subset $\mathcal{C}_T \subseteq \mathcal{C}^* \backslash \mathcal{C}$ of *cuts*, or violated inequalities, of type T, and update $\mathcal{C}_0 \coloneqq \mathcal{C}_0 \cup \mathcal{C}_T$; if $|\mathcal{C}_0| \geq \min\{c_{\min}, c_{\max} - |\mathcal{C}|\}$, then update $\mathcal{C} \coloneqq \mathcal{C} \cup \mathcal{C}_0$ and immediately go back to Step 1. After search is complete, if $\mathcal{C}_0 = \emptyset$, then update $z \coloneqq z^*$ and exit (proceeding to Step 3 of SBC); otherwise, update $\mathcal{C} \coloneqq \mathcal{C} \cup \mathcal{C}_0$ and go back to Step 1.

Indeed, the above procedure will improve the bound z on $\mathcal{J}$ until either the problem is proved infeasible ($z \geq z_{\text{ub}}$), a better integral solution is found, tailing off is observed, or not enough cuts are being generated. Note that after each iteration we remove all non-binding inequalities—placing them into a pool $\mathcal{D}$—due to the increase in computational demand correlated with solving SDPs with large constraint sets. For this same reason, we limit the total number of cuts per SDP solve, *e.g.* $c_{\max} = 2,000$; we resolve an SDP as soon as a minimum number of new cuts have been generated, *e.g.* $c_{\min} = 300$; we leave inequalities in the pool $\mathcal{D}$ for $d_{\max} = 5$ rounds; and we limit the number of consecutive SDP relaxations solved, *e.g.* $t_{\max} \coloneqq 20$. The improvement at each iteration is dependent upon the quality of the relaxation, *i.e.* the strength of the cuts found in Step 3. This is discussed more in the next section.

3.3.2 Strengthening with linear inequalities

For any subproblem $\mathcal{J}$ of $\mathcal{I}$ defining input $R(\mathcal{I}, \mathcal{J}) = (m, k, \alpha, \beta, S', W', a)$ to the **weight capacitated k-partition** (see §3.1.2), we solve multiple semidefinite programming relaxations of the form SWCKP (3.12)–(3.16). The quality of each depends on the set of cuts $\mathcal{C}$ maintained in each step of the procedure summarized in the previous section. Since each subproblem comes equipped with an initial set $\mathcal{C}$—the formulation of which is discussed below—we present a description of how we attempt to generate cuts at a single, arbitrary iteration. In particular, since in Step 3a, all violated inequalities of $\mathcal{D}$ are added to $\mathcal{C}$, we give the separation procedures of each inequality type (Step 3b), for a given fractional solution $Y^* \in \mathbb{R}^{m \times m}$. Note that some inequalities may be duplicated using our cut generation techniques, in which case any implementation of our algorithm must check whether a given violated inequality $(A, b) \in \mathcal{C}^*$ has already been included in $\mathcal{C}$; a standard hash table would suffice. Also, recall that a detailed description of each of the valid inequalities can be found in §2.3.

1. *Lower bound inequalities*: $Y_{ij} \geq \frac{-1}{k-1}$. Let $L \subseteq \binom{m}{2} \setminus S'$ denote the (largest) set of pairs for which the lower bound inequalities are violated by the point Y^*, *i.e.*

$$Y^*_{ij} < \tfrac{-1}{k-1} \quad \text{for } \{i, j\} \in L.$$

   All $O(m^2)$ inequalites can be exhaustively scanned in polynomial time, which allows us to eliminate the storage of such cuts in $\mathcal{D}$. If $|L| \geq c^+ := c_{\max} - (|\mathcal{C}| + |\mathcal{C}_0|)$, then add to $\mathcal{C}_0$ only the c^+ *most violated* such inequalities, *i.e.* those corresponding to the c^+ largest values of $\frac{-1}{k-1} - Y^*_{ij} > 0$. Otherwise, add all $|L|$ inequalities to $\mathcal{C}_0$. In general, obtaining the most violated inequalities is accomplished by pushing each cut and its residual onto a priority queue, and popping the desired number of entries.

   If at the root, and $\mathcal{C}_0 = \emptyset$, *i.e.* during the first cutting plane iteration, we generate $\frac{1}{4}\binom{m}{2}$ random lower bound inequalities.

2. *Triangle inequalities*: $-Y_{ij} + Y_{ih} + Y_{jh} \leq 1$. Let $T \subseteq [n]^3$ denote the (largest) set of distinct triples—triangles—(i, j, h), with $i < j$, for which the triangle inequalities

Algorithm 3: greedy lower clique coverings

Input : positive integers m, k, index $v \in [m]$, and symmetric $Y \in \mathbb{R}^{m \times m}$

Output: a family $\mathcal{F} \in 2^{[m]}$ of size $|F| \leq n - k - 1$, with $|F| > k$ for each $F \in \mathcal{F}$

Initialize $\mathcal{F} \coloneqq \emptyset$, $K \coloneqq \{v\}$, $V \coloneqq [n] \backslash \{v\}$, and $z \coloneqq 0$;

while $V \neq \emptyset$ **do**

  Find index $\ell = \arg\min_{i \in V} \left\{ \sum_{j \in K} Y_{ij} \right\}$;

  Increase $z \coloneqq z + \sum_{j \in K} Y_{\ell j}$;

  Update $K \coloneqq K \cup \{\ell\}$, and remove $V \coloneqq V \backslash \{\ell\}$;

  Let $r \coloneqq |K| \mod k$;

  if $|K| > k$ *and* $z < \frac{-|K|}{2} + \frac{r(k-r)}{2(k-1)}$ **then**

    Update $\mathcal{F} \coloneqq \mathcal{F} \cup \{K\}$;

are violated by Y^*, *i.e.*

$$-Y^*_{ij} + Y^*_{ih} + Y^*_{jh} > 1 \quad \text{for } (i,j,h) \in T.$$

Scanning all $O(n^3)$ inequalities is not particularly prohibitive—hence, we also eliminate triangles from being stored in the pool $\mathcal{D}$. If $|T| \geq c^+ \coloneqq c_{\max} - (|\mathcal{C}| + |\mathcal{C}_0|)$, then add to $\mathcal{C}_0$ only the c^+ *most violated* such inequalities; otherwise, include all $|T|$ inequalities in $\mathcal{C}_0$.

If at the root, and $\mathcal{C}_0 = \emptyset$, *i.e.* during the first cutting plane iteration, we generate up to $\frac{1}{4}\binom{m}{2}$ random triangle inequalities, and immediately resolve the SDP.

Of the inequalities we generate, the lower and triangle inequalities above are the only polynomial-time separable cuts for WCKP. Since they are facet-defining for the polyhedra which contain that of the **weight capacitated k-partition** problem, we include as many as possible. Recall that the procedure which calls the cutting plane subroutine exits the cut generation process as soon as "enough" cuts are generated, *i.e.* when $|\mathcal{C}_0| \geq \min\{c_{\min}, c_{\max} - |\mathcal{C}|\}$, which is implicit to our current presentation. The remaining inequalities have $\mathcal{NP}$-hard separation problems, and thus our algorithms are heuristic in nature.

3. *(Lower) clique inequalities* (2.52). Here, we attempt to generate cuts of the form:

$$\sum_{\{i,j\}\in\binom{K}{2}} Y_{ij} \geq \frac{-|K|}{2} + \frac{r(k-r)}{2(k-1)}, \tag{3.17}$$

where $K \subseteq [m]$ is of size $|K| > k$, and we define $r \coloneqq |K| \mod k$. In particular, we call the greedy lower clique covering heuristic (GLCC), or Algorithm 3—a modification of GCC (Algorithm 1)—on the solution matrix $Y^* \in \mathbb{R}^{m\times m}$ and each vertex $v \in [m]$, which outputs a family $\mathcal{F} \in 2^{[m]}$ of subsets $K \subseteq [m]$, each of which defines a violated lower clique inequality (3.17). The algorithm essentially grows a set K starting with the single vertex v, and at each iteration it finds a vertex $\ell \in [m] \setminus K$ which increases the expression

$$\sum_{\{i,j\}\in\binom{K\cup\{\ell\}}{2}} Y^*_{ij} - \sum_{\{i,j\}\in\binom{K}{2}} Y^*_{ij} \tag{3.18}$$

by the least amount. When K is large enough ($|K| > k$) and the sum $\sum_{\{i,j\}\in\binom{K}{2}} Y^*_{ij}$ is small enough—less than the right-hand side of (3.17)—then K is appended to the family $\mathcal{F}$.

This clique separation heuristic runs in $O(m^3)$ time. To see this, note that GLCC can be implemented to run in $O(m^2)$ time by maintaining the value of (3.18) at each iteration. Indeed, for each of the $O(m)$ iterations within GLCC, we can update this value for each of $O(m)$ vertices in constant time. Since the algorithm is called for each vertex in $[m]$, the claim holds. The total number of distinct clique inequalities produced is at most $m(m-k) - (m-1) = O(m^2)$, since the clique $K = [m]$ may be duplicated $m-1$ times. In fact, this bound can be reduced by an amount of $m\left\lfloor \frac{m}{k} \right\rfloor$: if Y^* satisfies the lower bound inequalities[1], then we can invoke Proposition 2.3.9, which tells us that whenever k divides $|K|$, the corresponding lower clique inequality is satisfied automatically by Y^*. Notice that the bound on the total number of cliques becomes $m\left(m - \left\lfloor \frac{m}{k} \right\rfloor - k - 1\right) + 1$, or roughly $m(m-k)\left(1 - \frac{1}{k}\right)$.

[1]This condition may be relaxed, as the semidefiniteness of Y^* implies $\chi_K^\top Y \chi_K \geq 0$ for any subset $K \subseteq [m]$, *i.e.* $\sum_{\{i,j\}\in\binom{K}{2}} Y^*_{ij} \geq \frac{-|K|}{2}$.

Note that if the GLCC procedure is altered to only output the first violated lower clique inequality, rather than all of them, then the running time of our separation heuristic is still $O(m^3)$ in the worst case, but no more than m cliques may be generated. Thus we choose to implement the former, since furthermore, Proposition 2.3.8 tells us that the elliptope $\mathcal{E}(m,k)$ contains matrices which violate all lower clique inequalities, barring the case where k divides $|K|$. Letting $\mathcal{K}$ denote the final set of clique inequalities generated, if $|\mathcal{K}| \geq c^+ := c_{\max} - (|\mathcal{C}|+|\mathcal{C}_0|)$, then add to $\mathcal{C}_0$ only the c^+ *most violated* such inequalities; otherwise, include all $|\mathcal{K}|$ inequalities in $\mathcal{C}_0$.

4. *2-Partition Inequalities* (2.49). We attempt to generate valid inequalities of the following form, for disjoint sets $S, T \subseteq [m]$, defining $s := |S|, t := |T|$:

$$\sum_{(i,j)\in S\times T} Y_{ij} - \sum_{\{i,j\}\in\binom{S}{2}\cup\binom{T}{2}} Y_{ij} \leq \min\{s,t\} + \tfrac{1}{k-1}\binom{|t-s|}{2}. \tag{3.19}$$

In particular, since the separation problem is $\mathcal{NP}$-hard even when $s = 1$, we choose S and T so that we have sizes $s = 1$ and $3 \leq t \leq k$. Indeed, Proposition 2.3.6 tells us that whenever $|t-s| = 0$ or $|t-s| \geq k$, and Y^* is in the truncated elliptope $\mathcal{E}(m,k)$, then the corresponding 2-partition inequalities are satisfied automatically. In fact, the inequalities cannot be binding for $|t-s| > k$. On the other hand, Proposition 2.3.7 states that there exist matrices in $\mathcal{E}(m,k)$ which violate each 2-partition for $1 \leq |t-s| \leq k-1$. Note that the triangle inequalities arise whenever $s = 1$ and $t = 2$, and thus we restrict attention to $t \geq 3$. Furthermore, the 2-partition inequalities are only generated when $k \geq 3$ (though the triangle inequalities are still separated above for $k = 2$; see point 2. above).

A heuristic procedure similar to GWA (Algorithm 2), is our randomized partition search (RPS), displayed as Algorithm 4. We call RPS for each vertex $s \in [m]$, and aim to form a family of valid 2-partition inequalities that Y^* violates, *i.e.* find $T \subseteq [m] \setminus \{s\}$, $3 \leq |T| \leq k$, so that the following holds:

$$\sum_{j\in T} Y^*_{sj} - \sum_{\{i,j\}\in\binom{T}{2}} Y^*_{ij} > 1 + \tfrac{1}{k-1}\binom{t-1}{2}. \tag{3.20}$$

Algorithm 4: randomized partition search

Input : positive integers m and k, index $s \in [m]$, and non-negative, symmetric $Y \in \mathbb{R}^{m \times m}$

Output: A family $\mathcal{T} \in 2^{[n] \setminus \{s\}}$ of size $|\mathcal{T}| \leq m - 4$, in which $3 \leq |T| \leq k$ for each $T \in \mathcal{T}$

Initialize $T \coloneqq \emptyset$, $\mathcal{T} \coloneqq \emptyset$, and $z \coloneqq 0$;

Define $V \coloneqq \left\{ i \in [m] \setminus \{s\} \,\middle|\, Y_{is} > \frac{-1}{k-1} \right\}$;

Let $\pi : \{1, \ldots, |V|\} \to V$ be an arbitrary permutation of V;

for $i = 1, \ldots, |V|$ **do**

  Let $v \coloneqq \pi(i)$;

  if $|T| < k$ **then**

    Define $\delta \coloneqq Y_{vs} - \sum_{j \in T} Y_{vj}$;

    if $\delta > 0$ **then**

      Increase $z \coloneqq z + \delta$;

      if $|T| \geq 3$ *and* $z > 1 + \frac{1}{k-1}\binom{|T|-1}{2}$ **then**

        Update $\mathcal{T} \coloneqq \mathcal{T} \cup \{T\}$;

  else

    for $\ell \in T$ **do**

      Define $\delta_\ell \coloneqq \left(Y_{vs} - \sum_{j \in T \setminus \{\ell\}} Y_{vj}\right) - \left(Y_{\ell s} - \sum_{j \in T \setminus \{\ell\}} Y_{\ell j}\right)$;

    Let $t \coloneqq \arg\max_{\ell \in T} \{\delta_\ell\}$;

    if $\delta_t > 0$ **then**

      Update $T \coloneqq (T \setminus \{t\}) \cup \{v\}$;

      Increase $z \coloneqq \delta_t$;

      `/* Recall that here,` $|T| = k$ `*/`

      if $z > 1 + \frac{1}{k-1}\binom{|T|-1}{2} = \frac{k}{2}$ **then**

        Update $\mathcal{T} \coloneqq \mathcal{T} \cup \{T\}$

In RPS, the vertices $[m]\backslash\{s\}$ are searched in a random order, and at each iteration, the considered vertex $\ell \in [m]\backslash T$ is appended to T only if $|T|< k$ and the expression

$$Y_{\ell s} - \sum_{j\in T} Y_{uv} > 0 \tag{3.21}$$

holds, or, when $|T|= k$, if there exists a vertex in $u \in T$ such that replacing u with ℓ results in a net increase, *i.e.*

$$\left(Y^*_{s\ell} + \sum_{j\in T\backslash\{u\}} Y^*_{j\ell}\right) - \left(Y^*_{su} + \sum_{j\in T\backslash\{u\}} Y^*_{ju}\right) > 0. \tag{3.22}$$

If the latter holds, then $u \in T$ is selected so that (3.22) yields the largest net increase. Each time the set T is altered, and $|T|\geq 3$, RPS adds T to the family of 2-partition inequalities $\mathcal{T}$ only if (3.20) holds for Y^*, *i.e.* if the corresponding 2-partition inequality is violated.

This separation algorithm runs in $O(m^2k^2)$, since we call RPS n times, and each call takes $O(mk^2)$. Indeed, RPS scans $O(m)$ vertices, each time comparing the left-hand side sum of (3.21)—which is the sum of $O(|T|)$ entries—to that of at most $|T|$ vertices of T. Since $|T|= O(k)$, the claim follows. The total number of distinct 2-partitions generated is at most $m(m-4) = O(m^2)$. If RPS is altered to only output the first violated 2-partition found, the overall separation procedure would still run in $O(m^2k^2)$, but not more than m violated inequalities would be produced. We choose the former routine, since Proposition 2.3.7 implies that it is possible for all inequalities with $|S|= 1$ and $3 \leq |T|\leq k$ to be violated by Y^*.

5. *Star cover inequalities* (2.65). Here, we attempt to separate the extended start cover inequality:

$$\sum_{j\in C} Y_{ij} \geq |E(C)|-|C|+1 - \frac{|C|-1}{k-1}, \tag{3.23}$$

for some minimal cover $C \subseteq [m]$ and its extension

$$E(C) = C \cup \left\{\ell \in [m]\backslash C \;\middle|\; a_\ell \geq \max_{h\in C} a_h\right\},$$

where index $i \in [m] \setminus C$. The heuristic separation of such inequalities is as follows. For each $v \in [m]$, find a minimal cover $C \subseteq [m] \setminus \{v\}$, by iteratively appending a vertex $\ell \in [m] \setminus (C \cup \{v\})$ for which $Y^*_{v\ell}$ is maximum over

$\left\{h \in [m] \setminus (C \cup \{v\}) \,\middle|\, \sum_{i \in C} a_i - a_h > n - \alpha\right\}$, *i.e.* at each step, choose the vertex external to C for which $C \setminus \{\ell\}$ is still a cover, and minimizes $\sum_{i \in C} Y^*_{vi} - Y^*_{v\ell}$. Once the set is found, extend it to $E(C)$, and test the inequalities 3.23 for violation, for each $u \in [m] \setminus E(C)$.

Note that this star covering separation heuristic is not entered if $\alpha \leq \min_{i \in [m]} a_i$.

6. *Flower inequalities* (2.71) Finally, we attempt to separate the weighted flower inequalities:

$$\sum_{\{i,j\} \in \binom{F}{2}} a_i a_j Y_{ij} + \sum_{(i,j) \in F \times [m] \setminus F} a_i a_j Y_{ij} \geq \tfrac{k}{k-1} \left(\binom{\alpha}{2} q + \binom{r}{2} + r(\alpha - r) - \sum_{i=1} m \binom{a_i}{2} \right) + \tfrac{-1}{k-1} \left(\binom{c}{2} + c(n-c) \right), \tag{3.24}$$

for any set $F \subseteq [m]$ which satisfies $|F| \leq \alpha k_\alpha$, where $c \coloneqq \sum_{i \in F} a_i$, $q \coloneqq \left\lfloor \frac{c}{\alpha} \right\rfloor$, $r \coloneqq c \mod \alpha$, and $n = \sum_{i \in [m]} a_i$, and where k_α denotes the largest number of parts of minimum size (*i.e.* α) in any k-partition of $[n]$—where n is the original size of the SBC input $\mathcal{I}$—with parts bounded in size between α and β. By Proposition 2.1.3, we must have $k_\alpha > 0$. Since the number of coefficients of the flower inequalities are dense, we proceed in a complementary fashion to the greedy clique covering (GCC), or Algorithm 1. In particular, for each vertex $v \in [m]$, we grow the initial set $F \coloneqq \{v\}$, of cost $\sum_{u \in [m] \setminus \{v\}} Y^*_{uv}$, in a greedy fashion, *i.e.* at each step, we find an external vertex $u \in [m] \setminus F$ which minimizes

$$\sum_{i \in [m] \setminus (F \cup \{u\})} a_i a_u Y^*_{iu}.$$

Continue until the first violated flower inequality (3.24) is found, or the maximum number of allowed iterations has been met—there is to be only a small number of iterations, *e.g.* $\min\{\alpha, k\}$.

Note that this separation heuristic is not called if $\alpha \leq \min_{i \in [m]} a_i$.

In preliminary tests, it was seen that, unless $\left\lceil \frac{n}{k} \right\rceil = \beta$, many of the valid inequalities of the form $\sum_{\{i,j\} \in \binom{[m]}{2}} A_{ij} Y_{ij} \leq b$ with positive coefficients A_{ij} and positive right-hand side b discussed in the previous sections turn out to be too weak for our problem,

i.e. are almost never violated. Intuitively, it can be argued that since our objective is to minimize $cW' \bullet Y + d$ for non-negative c and W', implies that entries Y_{ij} strive to be close to the lower bound $\frac{-1}{k-1}$. Hence these types of inequalities were not included in our algorithm SBC.

3.4 Upper bounding procedure

In this section, we present a heuristic algorithm called by SBC when attempting to bound the **capacitated** k**-partition** instance $\mathcal{I} \coloneqq (n, k, \alpha, \beta, D, W)$ from above, *i.e.* when generating feasible weight-capacited k-partitions of $[n]$. Intuitively, the procedure, denoted BIN, aims to first form any feasible weight-capacitated k-partition $\mathcal{V}$ of $\mathcal{I}$ by "packing" vertices one-by-one into k distinct parts or *bins*, and further "swapping" vertices between bins. Once the set of bins forms a feasible k-partition, vertices are further swapped whenever the resulting partition realizes a smaller total weight.[2]

For the input $\mathcal{I}$ to SBC, we define the following auxiliary global variables. At the start of the main algorithm SBC, we initially let $\mathcal{R}_{\text{glo}} \coloneqq \mathcal{R}_{\text{loc}} \coloneqq \mathcal{R}(\mathcal{I}, \mathcal{I})$ be respective tuples denoting *global* and *local* instances of **weight capacitated** k**-partition** problem. We also let $c_{\text{glo}} \coloneqq c_{\text{loc}} \coloneqq 0$ be integer counters of the following subroutines, with $c_{\max}$ denoting an upper bound on both values, and $t_{\max}$ a separate global counter limit (to be discussed). Further initialize $\mathcal{V}_{\text{glo}} \coloneqq \mathcal{V}_{\text{loc}} \coloneqq (\emptyset, \ldots, \emptyset)$, which are to be, respectively, global and local *k-packings* of their corresponding problems $\mathcal{R}_{\text{glo}}$ and $\mathcal{R}_{\text{loc}}$, *i.e.* each is of the form $(V_1, \ldots, V_k)$ where subsets V_i are disjoint and possibly empty. Define $\cup\mathcal{V} \coloneqq \bigcup_{i \in [k]} V_i$, for any k-packing $\mathcal{V} = (V_1, \ldots, V_k)$.

Once a given a subproblem $\mathcal{J} = (m, S, \tau, z)$ of $\mathcal{I}$ has reached this upper bounding procedure, we call BIN, which consists of calling first the global subroutine GBIN (Algorithm 5) on global input $\mathcal{I}$, and then local subroutine LBIN (Algorithm 6) on local input $\mathcal{J}$. The output of the former algorithm provides a k-packing of $\mathcal{R}(\mathcal{I}, \mathcal{I})$ (*i.e.* a global k-packing), whereas the latter provides a k-packing of $\mathcal{R}(\mathcal{I}, \mathcal{J}')$, for some

[2]There exist successful heuristics for many subcases of our problem, *e.g.* **maximum** k**-cut**—the case $G = \emptyset, \alpha = 1$, and $\beta = n$—has a provably good approximation algorithm, in expectation [38]. In such cases, the associated tailored methods may work better than our more general procedure, and can therefore be called where appropriate.

Algorithm 5: global subroutine of BIN: GBIN

Let n be number of vertices of $\mathcal{R}_{\text{glo}}$;

if $c_{glo} = c_{\max}$ **then**

  reset $c_{\text{glo}} \coloneqq 0$, and $\mathcal{V}_{\text{glo}} \coloneqq (\emptyset, \ldots, \emptyset)$;

else

  increment $c_{\text{glo}} \coloneqq c_{\text{glo}} + 1$;

if $\cup \mathcal{V}_{glo} \subsetneq [n]$ **then**

  call PACK on $(\mathcal{R}_{\text{glo}}, \mathcal{V}_{\text{glo}})$, resulting in solution $\mathcal{V}'$;

  if $\cup \mathcal{V}' = [n]$ **then**

    call SWAP on $(\mathcal{R}_{\text{glo}}, \mathcal{V}')$, resulting in solution $\mathcal{V}' \coloneqq \mathcal{V}''$, and reset $c_{\text{glo}} \coloneqq 0$;

else

  call SWAP on $(\mathcal{R}_{\text{glo}}, \mathcal{V}_{\text{glo}})$, resulting in solution $\mathcal{V}'$;

define $\mathcal{V}_{\text{glo}} \coloneqq \mathcal{V}'$;

subproblem possibly distinct from $\mathcal{J}$. Thus, if either output represents a feasible weight-capacitated k-partition of $\mathcal{I}$, and is of cheaper cost than the best known solution $\mathcal{V}_{\text{ub}}$, then we update $\mathcal{V}_{\text{ub}}$ and z_{ub} appropriately. Otherwise, the global and local k-packings found, are stored for future manipulation, respectively, as $\mathcal{V}_{\text{glo}}$ and $\mathcal{V}_{\text{loc}}$. The limit $c_{\max}$ denotes the maximum number of consecutive attempts BIN makes at minipulating a k-packing into a k-partition[3], and similarly, from a k-partition to a weight-capacitated k-partition. This ensures that neither $\mathcal{V}_{\text{glo}}$ or $\mathcal{V}_{\text{loc}}$ will be "stuck", and in the local case, it allows the exploration of k-packings and k-partitions with respect at least a fraction of $\frac{1}{2c_{\max}}$ distinct subproblems $\mathcal{J}'$ generated by SBC.

All that remains is an explanation of the subroutines PACK and SWAP found in both GBIN and LBIN, which are discussed in the following sections.

[3]Note that, indeed, if the vertex set is $[m]$ and k-packing $\mathcal{V}$ of $[m]$ is such that $\cup \mathcal{V} = [m]$, then $\mathcal{V}$ can be manipulated into a k-partition by taking any empty bin and "filling" it with half of another non-empty bin's constituents. Furthermore, the total weight cannot increase when doing so, since $W' \in \mathbb{R}^{m \times m}$ is nonnegative.

Algorithm 6: local subroutine of BIN: LBIN

Input : subproblem $\mathcal{J}$ of $\mathcal{I}$

Let n be number of vertices of $\mathcal{R}_{\text{loc}}$;

if $c_{loc} = c_{\max}$ **then**

  reset $c_{\text{loc}} \coloneqq 0$, $\mathcal{V}_{\text{loc}} \coloneqq (\emptyset, \ldots, \emptyset)$, and $\mathcal{R}_{\text{loc}} \coloneqq R(\mathcal{I}, \mathcal{J})$;

else

  increment $c_{\text{loc}} \coloneqq c_{\text{loc}} + 1$;

if $\cup \mathcal{V}_{loc} \subsetneq [n]$ **then**

  call PACK on $(\mathcal{R}_{\text{loc}}, \mathcal{V}_{\text{loc}})$, resulting in solution $\mathcal{V}'$;

  if $\cup \mathcal{V}' = [n]$ **then**

    call SWAP on $(\mathcal{R}_{\text{glo}}, \mathcal{V}')$, resulting in solution $\mathcal{V}' \coloneqq \mathcal{V}''$, and reset $c_{\text{loc}} \coloneqq 0$;

else

  call SWAP on $(\mathcal{R}_{\text{loc}}, \mathcal{V}_{\text{loc}})$, resulting in solution $\mathcal{V}'$;

define $\mathcal{V}_{\text{loc}} \coloneqq \mathcal{V}'$;

3.4.1 Capacitated bin packing heuristic

Supposing that $\mathcal{R} \coloneqq \mathcal{R}(\mathcal{I}, \mathcal{J}) = (m, k, \alpha, \beta, S', W', a)$ is the **weight capacitated** k-**partition** input with respect to subproblem $\mathcal{J} = (m, S, \tau, z)$, and $\mathcal{V} = (B_1, \ldots, B_k)$ is a k-packing of $[m]$, we demonstrate the subroutine PACK (Algorithm 7) with respect to input $(\mathcal{R}, \mathcal{V})$. The algorithm essentially takes the vertices of $[m] \setminus \cup \mathcal{V}$, and in some order, it attempts to place each into some bin B_i, until either $\mathcal{V}$ becomes a k-partition of $[m]$—possibly not feasible to $\mathcal{R}$—or the number of attempts has reached the limit $t_{\max}$. In the former case, we proceed to call SWAP (see next section) which further attempts to manipulate $\mathcal{V}$ into a weight-capacitated k-partition of $[m]$. In either case, the call to PACK ends in a k-packing being output and stored in one of the global variables, $\mathcal{V}_{\text{glo}}$ or $\mathcal{V}_{\text{loc}}$. We present a more detailed version of the procedure next. First, recall that $a = (a_i : i \in [n])$, where $\sum_{i \in [m]} a_i = n$. For convenience, define, for any subset $S \subseteq [m]$, $a(S) \coloneqq \sum_{i \in [m]} a_i$.

The algorithm PACK (Algorithm 7) is not guaranteed to stop with a k-partition

Algorithm 7: packing subroutine of BIN: PACK

Input : input tuple $\mathcal{R} \coloneqq (m, k, \alpha, \beta, S', W', a)$ of weighted **capacitated k-partition**, and k-packing $\mathcal{V} = (B_1, \ldots, B_k)$ of $[m]$

Output: a k-packing $(B_1, \ldots, B_k)$ of $[m]$

Define counter $t \coloneqq 0$;

Let $Q \coloneqq [m] \backslash \cup \mathcal{V}$ be a "first-in first-out" queue;

while $Q \neq \emptyset$ *and* $t < t_{\max}$ **do**

  select v as "top" of Q, reset $Q \coloneqq Q \backslash \{v\}$ and increment $t \coloneqq t + 1$;

  define $P \coloneqq \left\{ i \in [k] \,\middle|\, \binom{B_i \cup \{v\}}{2} \cap S' = \emptyset \right\}$;

  if $P \neq \emptyset$ **then**

    choose $\ell \in \arg\min_{i \in P} \{a(B_i \cup \{v\})\}$;

    if $a(B_\ell \cup \{v\}) \leq \beta$ **then**

      update $B_\ell \coloneqq B_\ell \cup \{v\}$;

    else

      find minimal $F \subseteq B_\ell$ such that $a(B_\ell \backslash F \cup \{v\}) \leq \beta$;

      update $B_\ell \coloneqq B_\ell \backslash F \cup \{v\}$;

      "push" each $u \in F$ onto Q;

  else

    choose $\ell \in \arg\min_{i \in [k]} \left\{ |\binom{B_i \cup \{v\}}{2} \cap S'| \right\}$;

    define $F \coloneqq \binom{B_\ell}{2} \cap S' \neq \emptyset$;

    if $a(B_\ell \backslash F \cup \{v\}) \leq \beta$ **then**

      update $B_\ell \coloneqq B_\ell \backslash F \cup \{v\}$;

      "push" each $u \in F$ onto Q;

    else

      find minimal $F' \subseteq B_i \backslash F$ such that $a(B_i \backslash (F \cup F') \cup \{v\}) \leq \beta$;

      reset $B_\ell \coloneqq B_\ell \backslash (F \cup F') \cup \{v\}$;

      "push" each $u \in S \cup F$ onto Q;

of $[m]$, and thus not a weight-capacitated k-partition (*cf.* Proposition 2.1.4). Indeed, consider any iteration and a vertex $v \in [m] \setminus \cup_{i \in [k]} B_i$ (*i.e.* the "top" of Q). If there exists a bin B_i with no neighbor of v in S' (*i.e.* $S' \cap \binom{B_i \cup \{v\}}{2} = \emptyset$) and is of small enough weight to include v (*i.e.* $a(B_i \cup \{v\}) \leq \beta$), then v can be placed in B_i. In such a case, we choose the bin B_i for which $a(B_i \cup \{v\})$ is smallest, since, intuitively, this encourages an even distribution of bin weights at each iteration. Otherwise, we must remove vertices from B_i to make "room" for v, and so we attempt to remove only a small amount of vertices (*i.e.* minimal $F \subseteq B_i$ so that $a(B_i \setminus F \cup \{v\}) + a_v \leq \beta$ and $S' \cap \binom{B_i \setminus F \cup \{v\}}{2} = \emptyset$). This choice is made so that the queue Q does not grow too large after "pushing" the removed vertices back (*i.e.* $Q \coloneqq Q \cup F$). Once PACK exits, the resulting k-packing $\mathcal{V}'$ may be a k-partition of $[m]$, in which case BIN calls SWAP (see next section) on $(\mathcal{R}, \mathcal{V}')$; otherwise, BIN is exited entirely. Note that feasibility of $\mathcal{V}'$ reduces to checking if

$$a(B_i) \geq \alpha \quad \text{for } i \in [k],$$

since the remaining properties of a weight-capacitated k-partition

$$a(B_i) \leq \beta \quad \text{for } i \in [k],$$
$$\binom{B_i}{2} \cap S' = \emptyset \quad \text{for } i \in [k],$$

are maintained throughout each iteration. The order in which vertices are placed onto the queue is randomized, as to allow for variability in the search for a k-partition.

3.4.2 Capacitated bin swapping extension

Once a k-partition $\mathcal{V} = (V_1, \ldots, V_k)$ of $[n]$ is found by PACK (see previous section), for a given subproblem $\mathcal{J} = (m, S, \tau, z)$ defining **weight capacitated k-partition** instance $\mathcal{R} \coloneqq \mathcal{R}(\mathcal{I}, \mathcal{J}) = (m, k, \alpha, \beta, S', W', a)$, we proceed to the following SWAP procedure (Algorithm 8). Essentially, this subroutine of BIN locally swaps vertices between bins of k-partition $\mathcal{V} = (V_1, \ldots, V_k)$ of $[m]$, in an attempt to reach a weight-capacitated k-partition of low total cost. In particular, the algorithm starts by searching for a feasible weight-capacitated k-partition, and if one is reached, it calls another subroutine, FSWAP (Algorithm 9), which continues swapping while maintaining feasibility and

Algorithm 8: swapping subroutine of BIN: SWAP

Input : input tuple $\mathcal{R} \coloneqq (m, k, \alpha, \beta, S', W', a)$ of weighted **capacitated** k-**partition**, and k-partition $\mathcal{V} = (V_1, \ldots, V_k)$ of $[m]$

Output: a k-partition of $\mathcal{V}'$ of $[m]$

Let $L \coloneqq \{i \in [k] \,|\, a(V_i) < \alpha\}$ be a "first-in first-out" queue;

while $L \neq \emptyset$ *and* $t < t_{\max}$ **do**

  let $Q \coloneqq [k] \backslash L$ be a "first-in first-out" queue;

  select ℓ from "top" of L, remove $L \coloneqq L \backslash \{\ell\}$, and increment $t \coloneqq t + 1$;

  while $Q \neq \emptyset$ *and* $a(V_\ell) < \alpha$ **do**

    select q from "top" of Q, remove $Q \coloneqq Q \backslash \{q\}$;

    let $F \coloneqq \arg\max_{H \subseteq V_q} \left\{ |H| \,\middle|\, \binom{V_\ell \cup H}{2} \cap S' = \emptyset \right\}$;

    find $F' \coloneqq \arg\max_{H \subseteq F} \{a(H) : a(V_q \backslash H) \geq \alpha\}$ by greedy search;

    update $V_q \coloneqq V_q \backslash F'$ and $V_\ell \coloneqq V_\ell \cup F'$;

  if $a(V_\ell) < \alpha$ **then**

    "push" ℓ back onto L;

if $L = \emptyset$ **then**

  call FSWAP on $(\mathcal{R}, \mathcal{V}, t)$, letting $\mathcal{V}'$ be the solution returned;

else

  define $\mathcal{V}' \coloneqq (V_1, \ldots, V_k)$;

never increasing the overall cost of the present solution. Note that the latter subroutine, FSWAP, can be called independently on any feasible weight-capacitated k-partition; for instance, when creating a parallel algorithm this can be assigned to one thread with the sole purpose of improving incumbent solutions.

For the given input $\mathcal{R}$ and k-partition $\mathcal{V}$ of $[m]$, the "closeness" of $\mathcal{V}$ to being feasible can be measured by how many parts V_h, $h \in [k]$, are *capacitated*, *i.e.* satisfying

$$a(V_h) \geq \alpha, \tag{3.25}$$

$$a(V_h) \leq \beta, \tag{3.26}$$

$$\binom{V_h}{2} \cap S' = \emptyset. \tag{3.27}$$

Indeed, if all parts are capacitated, then the k-partition is a weight-capacitated k-partition. The algorithm SWAP attempts to manipulate $\mathcal{V}$ into a feasible solution by increasing the number of capacitated parts. In particular, it first partitions $[k]$ into the set of capacitated parts of $\mathcal{V}$, $Q \coloneqq \{h \in [k] \,|\, a(V_h) \geq \alpha\}$, and the set of non-capacitated parts, $L \coloneqq [m] \setminus Q$. Then, for any $\ell \in L$, and some $q \in Q$, the algorithm removes a maximal set of vertices $F \subseteq V_q$ for which the three properties (3.25)–(3.27) hold for $V_q' \coloneqq V_q \setminus F$, and the latter two properties (3.26)–(3.27) hold for $V_\ell' \coloneqq V_\ell \cup F$. Clearly, if $F \neq \emptyset$, then $|V_\ell'| > |V_\ell|$, and if further $|V_\ell'| \geq \alpha$, then the resulting k-partition $\mathcal{V}'$ is closer to being feasible—the number of capacitated parts of $\mathcal{V}'$ is strictly more than $\mathcal{V}$. To see that finding a maximal such set F yields $a(V_\ell') = a(V_\ell \cup F) \leq \beta$, notice that

$$\begin{aligned} a(V_\ell)' &= a(V_\ell) + a(F) && (3.28)\\ &\leq a(V_\ell) + (a(V_q) - \alpha) && (3.29)\\ &< \alpha + a(V_q) - \alpha && (3.30)\\ &\leq \beta, && (3.31) \end{aligned}$$

where the second line is due to $\alpha \leq a(V_q \setminus F)$, and the third line is due to $a(V_\ell) < \alpha$. This procedure is performed for each $\ell \in L$, until either the secondary search over Q has been exhausted, or when $a(V_\ell') \geq \alpha$, in which case ℓ may be placed into Q.

SWAP is not guaranteed to stop with a weight-capacitated k-partition, hence we limit the total of iterations spent search for one by $t_{\max}$. To increase variability in the search, we push elements onto either queue (L or Q in Algorithm 8), in a randomized order. If, however a solution is found, we proceed to FSWAP, which attempts to decrease the overall cost of the input partition.

The FSWAP subroutine takes a weight-capacitated k-partition of $[m]$, *i.e.* a feasible solution, and attempts to improve the value of the solution. In particular, the algorithm runs until a number of iterations has reached a limit, where an iteration is as follows. First, a part $\ell \in [k]$ from which some vertex may be removed, *i.e.* $a(V_\ell \setminus \{v\}) \geq \alpha$ for some $v \in V_h$, is selected randomly. Then the vertices of V_ℓ are scanned in some order,

Algorithm 9: swapping feasible solutions: FSWAP

Input : $t \in \mathbb{Z}_+$, input tuple $\mathcal{R} \coloneqq (m, k, \alpha, \beta, S', W', a)$ of **weight capacitated k-partition**, weight-capacitated k-partition $\mathcal{V} = (V_1, \ldots, V_k)$ of $[m]$

Output: a feasible weight-capacitated k-partition $\mathcal{V}'$ of $[n]$

while $t < t_{\max}$ **do**

choose $\ell \in [k]$ randomly, and increment $t \coloneqq t + 1$;

let $Q \coloneqq \{v \in V_\ell \,|\, a(V_\ell \backslash \{v\}) \geq \alpha\}$ be a "first-in first-out" queue;

while $Q \neq \emptyset$ **do**

let v be "top" of Q, and remove $Q \coloneqq Q \backslash \{v\}$;

let $L \coloneqq \left\{ h \in [k] \backslash \{\ell\} \,\middle|\, a(V_h \cup \{v\}) \leq \beta, \binom{V_h \cup \{v\}}{2} \cap S' = \emptyset \right\}$;

for $i \in L$ **do**

define $\delta_i \coloneqq \sum_{u \in V_\ell \backslash \{v\}} W_{uv} - \sum_{u \in V_i} W_{uv}$;

let $i^* \coloneqq \arg\min_{i \in L} \{\delta_i\}$;

if $\delta_{i^*} < 0$ **then**

update $V_{i^*} \coloneqq V_{i^*} \cup \{v\}$ and $V_\ell \coloneqq V_\ell \backslash \{v\}$;

with the intent of removing some $v \in V_\ell$ and replacing v into some part $h \in [k] \setminus \{\ell\}$. This can only happen if $V_h \cup \{v\}$ would still be a capacitated part, *i.e.* if

$$a(V_h \cup \{v\}) \leq \beta, \binom{V_h \cup \{v\}}{2} \cap S' = \emptyset. \tag{3.32}$$

Once these conditions are satisfied, v is replaced only if the net difference in cost is non-positive, *i.e.* if

$$\sum_{u \in V_\ell \setminus \{v\}} W_{uv} - \sum_{u \in V_i} W_{uv} < 0.$$

Indeed, this implies that any resulting k-partition output by FSWAP is in fact a feasible solution of cost not larger than the input partition $\mathcal{V}$. Since the procedure, however, is not guaranteed to strictly improve this cost, we randomize the order vertices are placed in the queues, as to increase variability.

3.5 Branching strategy

For given input $\mathcal{I} = (n, k, \alpha, \beta, D, W)$ to SBC, and an arbitrary subproblem $\mathcal{J} = (m, S, \tau, z)$ for which the **weight capacitated k-partition** input becomes $\mathcal{R}(\mathcal{I}, \mathcal{J}) = (m, k, \alpha, \beta, S', W', a)$, the branching procedure is as follows. First we select a pair of vertices of $[m]$ which are not fixed to be together in every feasible solution of $\mathcal{J}$, nor are they separated by every solution of $\mathcal{J}$, *i.e.* a pair $e \coloneqq \{r, s\} \in \binom{[m]}{2} \setminus S'$. Then two new subproblems $\mathcal{J}_0$ and $\mathcal{J}_1$ are formed and placed into the pool of active subproblems $\mathfrak{P}$, where $\mathcal{J}_0 \coloneqq (m, S_0, \tau_0, z)$ represents the set of solutions of $\mathcal{J}$ with r and s separate, and $\mathcal{J}_1 \coloneqq (m-1, S_1, \tau_1, z)$ represents the set of solutions of $\mathcal{J}$ with r and s together in some part. Here, we must have $S_0 \coloneqq S \cup \{e\}$, and $\tau_0 \coloneqq \tau$, since separating the pair $\{r, s\}$ does not yield a new k-partition of $[m]$.

On the other hand, for $\mathcal{J}_1$, merging $\tau(r)$ and $\tau(s)$, while maintaining the property that $\tau_1(i)$ must be the ith smallest representative, yields the definition:

$$\tau_1(i) \coloneqq \begin{cases} \tau(i) & \text{if } i < s, i \neq r, \\ \tau(r) \cup \tau(s) & \text{if } i = s, \\ \tau(i+1) & \text{if } i \geq s. \end{cases}$$

Indeed, by our choice to represent any part P of an m-partition of $[n]$ by its minimum constituent in $[n]$, we see that $\mathrm{rep}(\tau(r)) < \mathrm{rep}(\tau(s))$ (see §3.1), and the claim holds. Since the pairs of S are indices in $[m]$, the auxiliary transformation $\sigma\colon [m] \to [m-1]$:

$$\sigma(i) \coloneqq \begin{cases} i & \text{if } i < s, \\ r & \text{if } i = s, \\ i-1 & \text{if } i > s, \end{cases}$$

allows us to define S^*: $\{\sigma(i), \sigma(j)\} \in S^*$ if and only if $\{i, j\} \in S$ and $\sigma(i) \neq \sigma(j)$. Since $\{r, s\} \notin S$, the expression $\sigma(i) \neq \sigma(j)$ holds for every $\{i, j\} \in S$. It is clear now that $\mathcal{J}_0$ and $\mathcal{J}_1$ represent the desired partition of $\mathcal{J}$.

Recall from Section 3.3, that each subproblem is equipped with a set of cuts $\mathcal{C} \subseteq \mathcal{C}(\mathcal{I}, \mathcal{J})$, or pairs $(A, b) \in \mathcal{C}$ which represent valid inequalities of the form

$$\sum_{\{i,j\} \in \binom{[m]}{2}} A_{ij} Y_{ij} \leq b, \qquad \text{for } Y \in \mathcal{Y}(m, k, \alpha, \beta, S', a).$$

After branching on subproblem $\mathcal{J}$, we scan $\mathcal{C}$ by type (see §3.3.2), and update indices to be included in $\mathcal{C}_0$ and $\mathcal{C}_1$—the cuts to be included in $\mathcal{J}_0$ or $\mathcal{J}_1$, respectively. In particular, since $\mathcal{J}_0$ shares the same index set $[m]$, we let $\mathcal{C}_0$ be $\mathcal{C}$ after possibly removing the lower bound inequality corresponding to $\{r, s\}$; we thus proceed as follows for $\mathcal{C}_1$:

1. *Lower bound inequalities.* Suppose $\mathcal{C}$ contains lower bound inequalities represented by a set $L \subseteq \binom{[m]}{2}$. Then include in $\mathcal{C}_1$ those inequalities represented by the set $L_1 \subseteq \binom{[m-1]}{2}$, defined to have the pair $\{\sigma(i), \sigma(j)\}$ whenever $\{i, j\} \in L$ and $\sigma(i) \neq \sigma(j)$. Note that $\sigma(i) = \sigma(j)$ only if $\{i, j\} = \{r, s\}$.

2. *Triangle inequalities.* Suppose $\mathcal{C}$ contains triangle inequalities encoded by an ordered set $T \subseteq [m]^3$, where for any $(i, j, h) \in T$, we have $i < j$ and $h \notin \{i, j\}$. Then include in $\mathcal{C}_1$ those inequalities represented by the ordered set $T_1 \subseteq [m-1]^3$, defined to have the triple $(\min\{\sigma(i), \sigma(j)\}, \max\{\sigma(i), \sigma(j)\}, \sigma(h)) \in T_1$ whenever $(i, j, h) \in T$, and the elements $\sigma(i)$, $\sigma(j)$, and $\sigma(h)$ are distinct. Note that the three elements are distinct whenever $\{r, s\} \not\subseteq \{i, j, h\}$.

3. *(Lower) clique inequalities.* Suppose $\mathcal{C}$ contains lower clique inequalities represented by a family of sets $\mathcal{K} \in 2^{[m]}$, where for any $K \in \mathcal{K}$, we have $|K| \geq k+1$ and

$|K| \mod k \neq 0$. Then include in $\mathcal{C}_1$ those inequalities represented by the family $\mathcal{K}_1 \in 2^{[m-1]}$, defined to have the set $\sigma(K) \coloneqq \{\sigma(i) : i \in K\}$ whenever $K \in \mathcal{K}$ and $|\sigma(K)| \mod k \neq 0$. Note that $|\sigma(K)| \mod k = 0$ only if $\{r, s\} \subseteq K$ and $|K| \mod k = 1$.

4. *2-partition.* Suppose $\mathcal{C}$ contains 2-partition inequalities represented by a family $\mathcal{T} \in [m] \times 2^{[m]}$, where for any $(q, T) \in \mathcal{T}$, we have $q \notin T$ and $3 \leq |T| \leq k$. Then include in $\mathcal{C}_1$ those inequalities represented by the family $\mathcal{T}_1 \in [m-1] \times 2^{[m-1]}$, defined to have the tuple $(\sigma(q), \sigma(T))$ whenever $\sigma(q) \notin \sigma(T)$ and $3 \leq |\sigma(T)| \leq k$. Note that $\sigma(q) \in \sigma(T)$ if $q \in \{r, s\}$ and $\{r, s\} \cap T \neq \emptyset$, and $|\sigma(T)| < 3$ when $\{r, s\} \subseteq T$.

5. *Star cover inequalities.* Suppose $\mathcal{C}$ contains star cover inequalities represented by a family of sets $\mathcal{Q} \in 2^{[m]}$, where for any $Q \in \mathcal{Q}$, the Q is a minimal cover of $[m]$ with respect to capacity $n - \alpha$. Then include in $\mathcal{C}_1$ those inequalities represented by the family $\mathcal{Q}_1 \in 2^{[m-1]}$, defined to have the set $\sigma(Q)$ whenever $Q \in \mathcal{Q}$ and $\sigma(Q)$ is a minimal cover of $[m]$ with respect to capacity $n - \alpha$. Note that $\sigma(Q)$ is minimal whenever $\{r, s\} \subseteq Q$ or $\{r, s\} \cap Q = \emptyset$.

6. *Flower inequalities.* Suppose $\mathcal{C}$ contains flower inequalities represented by a family of sets $\mathcal{F} \in 2^{[m]}$, where for any $F \in \mathcal{F}$, we have $|F| \leq \alpha k_\alpha$. Then include in $\mathcal{C}_1$ those inequalities represented by the family $\mathcal{F}_1 \in 2^{[m-1]}$, defined to have $\sigma(F)$ whenever $F \in \mathcal{F}$ and $|\sigma(F)| \leq \alpha k_\alpha$. Note that $|\sigma(F)| > \alpha k_\alpha$ cannot occur.

Ideally, our branching choice of $\{r, s\}$ should partition $\mathcal{J}$ into subproblems which enable improvement upon z—the current best bound on $\mathcal{J}$. To this end, we choose the *least-decided* pair *i.e.*

$$\{r, s\} \in \arg\min \left\{ \left| \tfrac{1}{2} \left(1 + \tfrac{-1}{k-1} \right) - Y_{ij} \right| \,\middle|\, \{i, j\} \in \binom{[m]}{2} \backslash S \right\}, \tag{3.33}$$

where ties are broken by selecting, among the least-decided pairs, $\{r, s\}$ with smallest W_{rs} value. This choice was also made in Ghaddar *et al.* [41] regarding a similar SDP-based branch-and-cut algorithm which was applied to the **minimum k-partition** problem (a subclass of **capacitated k-partition**) based on empirical evidence presented by

Rendl *et al.* [86] in the **maximum cut** setting. They both observed that branching on the least-decided variable did in fact yield the largest margin of improvement in relaxation bounds when compared to other branching methods. This is in contrast with the findings for LP-relaxations (see [3] for a detailed review on branching), however, SDP interior-point methods differ in that "warm-starting" is difficult (see [8] for success in this area, applied to **minimum graph bisection** using spectral bundle methods).

Chapter 4

Test results and discussion

In this chapter we present an analysis of tests performed using our branch-and-cut algorithm SBC (§3). A description of implementation aspects is given first, followed by results from random tests on instances of the **capacitated k-partition** problem (§2.2.2). We end with a report on the performance of SBC when applied to the data set of a previous conference; namely, the 13th Annual INFORMS Computing Society (ICS-2013) [36].

4.1 Implementation

Our algorithm, SBC, was implemented in the C++ programming language environment, compiled with GCC version 4.6.3, on an x86-64 machine with an Intel Core i7 950 chip—clocked at 3.06GHz—with 9Gb of available RAM, and Linux Ubuntu 12.04 LTS.

4.1.1 Branch-and-bound framework

Our optimization algorithm is embedded in a branch-and-bound (B&B) framework, with subproblems bounded by semidefinite programming (SDP) solutions. We choose the general B&B shell found in Eckstein *et al.* [30], known as the Parallel Enumeration and Branch-and-Bound Library (PEBBL; v1.4). PEBBL is an open source C++ library, which allows for the customization of the bounding procedure, the enumeration of near-optimal solution, and the immediate parallelization of an embedded algorithm—our testing, however, remains in a serial setting. This library is supported by the ACRO optimization project [66] in that it allows PEBBL to be built with other auxiliary libraries—one which we particularly find useful is the general utilities library UTILIB

[67]. We use PEBBL build 1.4, compiled with optimization, along with many data structures of UTILIB.

4.1.2 Semidefinite programming solver

The SBC algorithm attempts to solve numerous subproblems, each of which employs an SDP relaxation of a weighted **capacitated** k-**partition** instance (*cf.* §3.1.2). The SDP solver chosen here is the open source SemiDefinite Programming Algorithm (SDPA) of Fujisawa *et al.* [39, 104], which encodes a Mehrotra predictor-corrector—or primal-dual—interior point method. We choose this solver for its callability as a C++ package, and its ability to be parallelized (both in multi-threading, and in MPI setting, though the latter requires an upgrade to SDPA: SDPARA [105]). Moreover, our preliminary tests compared SDPA favorably to the SemiDefinite and Dual Minimization (SeDuMi) software of Sturm [93] on small **maximum cut** instances, where as much as 10% speed-ups were observed. We compiled SDPA version 7.3.8, which solves matrix operations using the BLAS/LAPACK implementation OpenBLAS, version 0.2.8 [103, 99]. The standard form in which a semidefinite programming problem is represented is specified, in the SDPA manual, as the following pair of *primal* and *dual* problems, respectively:

$$\begin{aligned} \text{(P) minimize } & \sum_{i\in[M]} c_i x_i \\ \text{subject to } & X = \sum_{i\in[M]} F_i x_i - F_0 \\ & X \succeq 0, \end{aligned} \tag{4.1}$$

and

$$\begin{aligned} \text{(D) maximize } & F_0 \bullet Y \\ \text{subject to } & F_i \bullet Y = c_i \text{ for } i \in [M] \\ & Y \succeq 0. \end{aligned} \tag{4.2}$$

Here, the input consists of positive integers M and N, scalars c_i, for $i \in [M]$, and $N \times N$ matrices F_i, for $i \in \{0, 1, \ldots, M\}$, $x \in \mathbb{R}^M$. Furthermore, the vector $x = (x_i : i \in [M])$ and $N \times N$ matrix X are the primal variables, whereas $N \times N$ matrix Y is the dual variable.

Notice that each subproblem $\mathcal{J}$ yields a **weight capacitated k-partition** instance $\mathcal{R} \coloneqq \mathcal{R}(\mathcal{I}, \mathcal{J}) = (m, k, \alpha, \beta, S', W', a)$ for which the formulation of optimization model SWCKP (*cf.* §3.1.2) does not conform to either the primal or the dual format above. This is remedied by first replacing the objective

$$\text{minimize } \tfrac{k-1}{2k} W' \bullet \left(\tfrac{1}{k-1} J + Y \right),$$

with

$$\tfrac{1}{2k} W' \bullet J - \text{maximize } \left(\tfrac{-(k-1)}{2k} W' \right) \bullet Y.$$

Thus we now have an objective function which resembles that of a dual problem, only after an optimum to the problem is found, the answer must be negated and further translated by $\frac{1}{2k} W' \bullet J = \sum_{\{i,j\} \in \binom{[m]}{2}} W'_{ij}$. In other words, we are finding a maximum-weight (capacitated) k-cut of $\mathcal{J}$, and then we take the complementary minimum-weight k-partition solution and send it back to the main SBC algorithm. All that remains is transforming the set of constraints to appear similar to those in (4.2).

Any system of constraints containing a single linear inequality, $A \bullet Y \leq b$, and semidefinite requirement, $Y \in \mathcal{S}^m_+$, can be equivalently represented as the following:

$$\begin{bmatrix} A & 0 \\ 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} Y & 0 \\ 0 & \xi \end{bmatrix} = b. \tag{4.3}$$

$$\begin{bmatrix} Y & 0 \\ 0 & \xi \end{bmatrix} \succeq 0. \tag{4.4}$$

Note that the instance $\mathcal{R}$ formulated as SWCKP (*cf.* (3.12)–(3.16)) has m-dimensional matrix variables, where $m = O(n)$ with n denoting the initial size of the input $\mathcal{I}$ to SBC. The problem also contain $c = |\mathcal{C}|$ inequality constraints—$\mathcal{C}$ contains cuts generated as in section §3.3.2—and $m + |S'|$ equalities, where $s \coloneqq |S'| = O(m^2)$. Transforming this model, as above, into the dual form of SDPA results in an $(m+c)$-dimensional problem with $m+c+s$ constraints, all of which are now in equality form. The primal-dual interior point method used by SDPA (*cf.* Yamashita *et al.* [104, 105]), runs in $O(NM^2)$, where our case has $N = m + c$ and $M = m + c + s$. If S' is sparse, or $s = O(m)$, then the run time becomes $O((m + c)^3)$. Further, if a large number of constraints are included

within $\mathcal{C}$ at any iteration—*e.g.* all $\Theta(m^3)$ triangle inequalities—then the run-time is estimated to be $O(m^9)$. Clearly, care must be taken when generating $\mathcal{C}$, but regardless, only small problems instances can be solved to optimality, since SDPA will be called numerous times per subproblem.

Fortunately, growing a variable matrix increases overall sparsity of the problem, as the parameter matrices F_i are block matrices, with the first block coinciding with (possibly dense) matrices of dimension $m \times m$, whereas the second block is a $c \times c$ identity matrix corresponding to the slack variables of all original inequalities (*i.e.* 'ξ' in the above transformation).

As a final note, recall that calculating the Lovász-theta function $\vartheta(G)$ for $G = ([m], S')$ at subproblem $\mathcal{J}$, is part of the proprocessing subroutine of SBC (*cf.* §3.2), and it involves solving the SDP (3.8)–(3.11). Since the problem only needs to be solved at most once, and consists only of equality constraints, the problem size in SDPA format here—and thus the run-time per calculation—is much smaller relative to the multiple SDPs solved when bounding the subproblem.

4.2 Random data

In this section, we present the results from running our algorithm on randomly generated weighted matrices $W \in \mathbb{R}_+^{n \times n}$. To do so, we call the *Rudy graph generator*, compiled in C (available in [2]). In particular, we generate four different classes of graphs we refer to as "2-grid", "random", "clique", and "random grid". The 2-grid graphs are planar, bidimensional grids over n^2 vertices, where n is specified as part of the input to Rudy, and edge weights are either 0 or 1. For example our file `2grid_33`, represents the 2-grid on $3^2 = 9$ vertices. The random graphs are completely dense graphs on n vertices, with edge weights uniformly at random, picked from the set $\{1, \ldots, 10\}$; `random_20_109` is the random graph generated by Rudy, with $n = 20$ and a seed of 109 for the random number generator. Similarly, random grids are planar, with either 3 or 4 dimensions, and edges randomly generated over the interval $\{1, \ldots, 100\}$; an example of a 3-dimensional random grid graph appears in instance `444rand_100_398`, denoting

that $4^3 = 64$ vertices are present, and edge weights are selected using a random number generator with seed 398. Finally, the clique graphs are completely dense graphs with edges weights $W_{ij} = |j - i|$, for each $\{i, j\} \in \binom{[n]}{2}$; `clique_30` is one such graph we use, where here $n = 30$.

Many similar graphs have been generated and optimized over in Ghaddar *et al.* [41] as well as in the follow-up paper by Anjos *et al.* [5], both of which studied a branch-and-cut semidefinite programming algorithm similar to ours, and applied to **maximum k-cut** instances. The graphs they generated using Rudy included negative edge weights, since they were interested in physics applications, such as the Potts spin glass model. Unfortunately, our algorithm was not designed for such problem instances, as we focus on the conference scheduling application discussed in the first chapter.

Our algorithm SBC is tested on select instances $(n, k, \alpha, \beta, S, W)$ of **capacitated k-partition**, although for presentation purposes, we display resulting solutions in terms of the value of the complementary k-cut (*i.e.* $\sum_{i<j} W_{ij}$, less the k-partition cost). In particular, for each $n \times n$ matrix W arising from one of the four aforementioned types, we choose values of $k \in \{3, 5, 7\}$; two capacity scenarios $(\alpha, \beta) = (1, n)$ or $\left(\left\lfloor \frac{n}{k} \right\rfloor + 1, \left\lceil \frac{n}{k} \right\rceil - 1\right)$, referred to as *trivial* and *tight* capacities, respectively; and the set of separate pairs S (*cf.* §2.1) is one of three types, S_0, S_1, or S_2, explained next. For the set of vertices $[n]$, we define $S_0 \coloneqq \emptyset$, S_1 is a disjoint union of $\left\lfloor \frac{n}{3} \right\rfloor$ edges (*i.e.* $S_1 \coloneqq \{(3i - 2, 3i - 1) : i = 1, \ldots, \left\lfloor \frac{n}{3} \right\rfloor\}$), and S_2 is a maximal set of $\left\lfloor \frac{n}{3} \right\rfloor$ disjoint triangles (*i.e.* $S_2 \coloneqq \left\{(3i - 2, 3i - 1), (3i - 2, 3i), (3i - 1, 3i) : i = 1, \ldots, \left\lfloor \frac{n}{3} \right\rfloor\right\}$. We choose these settings since S_1 or S_2, together with tight capacities α and β, since the resulting problem instance is somewhat restricted and symmetric, yet intuitively hard to solve; in other words, a large number of k-colorings of the graph S remain.

Tests are reported in separate tables, each corresponding to a particular choice of parameters k, and α and β. The rows in each table are organized by choice of W, and the columns are subdivided into three sections corresponding to the particular choice of S (specified in the column header). For each instance, we present problem statistics of the average of three distinctly seeded runs. In particular, the columns contain, in order from left to right: the number of nodes appearing in the branch-and-bound search, the

total amount of time (CPU seconds) spent computing, the final relative gap in solution quality (between worst bound and best solution upon exiting the procedure), the initial relative gap found at the root, and the time spent solving all SDP instances arising within the tree nodes.

Some further parameters are specified for our experiments as follows. In each test, we only allowed 1000 seconds of total run-time, regardless of problem size. Since, however, the increase in problem size correlates with the increase in computational work, many of the instances could not be solved to optimality in such a short period of time. Recall that the representation of our problems in the SDP solver (SDPA) notation, grows proportionally with the number of cuts included, which in turn grow at least cubicly in n; this is especially prohibitive if all $\Theta(n^3)$ triangle cuts are included (*cf.* §3.3.2). But since the number of cuts must be adequate for improvement to be made at each subproblem, and we have observed in preliminary tests that the SDP problems do not branch too deep, we thus impose a fairly modest cap on the total number of cuts allowed at any given time, *i.e.* at 2000. We further stop adding cuts whenever more than 300 cuts have been added, allowing each subproblem—especially those for large n—to generate many of the polynomial time separable cuts prior to spending time on those which have $\mathcal{NP}$-hard separation problems.

After an SDP is solved we remove all non-binding inequalities and store them in a pool for at most 5 consecutive iterations—in preliminary testing, more than 5 iterations were unnecessary, and less than 3 showed a decrease in solution quality per SDP solved. The number of rounds the SDP relaxation is updated with new cuts and solved is at most 20, or until the increase in objective value is tailing off with relative tolerance 10^{-5}. When solving the SDP, we allow the solver (SDPA) at most 25 iterations to converge upon an approximate feasible solution. Since SDPA allows multi-threading—and our algorithm used PEBBL only in serial mode—we found that the optimal number of threads to use was 2 times the number of cores, thus we set this parameter to 8 for our given architecture (see §4.1 for specs). Otherwise, the default parameter settings of SDPA suffice for our tests.

When generating cutting planes, we completely enumerate the lower bound and

triangle inequalities, but otherwise the number of attempts at separating the harder inequalities is limited (see §3.3.2 for details). The clique inequalities turned out to be very helpful, and a large number appear to be binding at the tailing off of SDP solutions. On the other hand, many 2-partition inequalities, although violated in the initial phases of the cutting plane procedure, are ultimately only binding at optimality when in the form of triangle inequalities. Similarly, larger cliques ($c > k + 1$) appear frequently at the outset of the cutting plane procedure, but a majority of the clique inequalities which remain binding are those with $c = k + 1$. The last type of cut we generate, flower inequalities, rarely appeared to be violated, and only so at the root. Furthermore, we did not include the star covering inequalities in our current tests, as the clique and triangle inequalities appeared strong enough on their own, and their number at each iteration remained steadily large (*e.g.* for example, each dense problem with more than 60 nodes typically included all 2000 inequalities, which is roughly the same size as $\binom{60}{2}$, or the maximum number of lower bound inequalities possible).

In our heuristic procedure, we allowed at most 5 consecutive rounds of a k-packing to be "loaded" into PACK for packing into a k-partition, or to continue in SWAP towards a feasible weight-capacitated k-partition (see §3.3). Within either PACK or SWAP, we allow kn rounds of potential packing or swapping of vertices, respectively. Intuitively, this means that each of the n vertices has an average of k chances of being packed into or swapped out of a bin, k of which exist. Indeed, if a particular subproblem does not contain a feasible solution close to the global optimum, our heuristic cannot turn a local k-packing into a solution of best value; hence we limit the number of attempts at improving its value. We also note that the Lovasz-theta function computed in the preprocessing step is only computed if the size of the graph is large enough, *i.e.* if $|E(G)|$ is large and there are at least $k + 1$ vertices with degrees $k + 1$.

Tables 4.1 through 4.6 display our tests on instances with the four types of graphs generated as W, with n ranging from 9 to 90. The first table (4.1) displays statistics for each instance with $k = 3$, α and β trivial; here, the columns are broken up into three blocks according to choice of S, with columns respectively displaying the total number of nodes explored, total CPU time spent in seconds, relative final gap, relative

Table 4.1: $k = 3$, α and β trivial

	S_0					S_1					S_2				
W	# Nodes	CPU (s)	F. Gap (%)	R. Gap (%)	SDP (s)	# Nodes	CPU (s)	F. Gap (%)	R. Gap (%)	SDP (s)	# Nodes	CPU (s)	F. Gap (%)	R. Gap (%)	SDP (s)
2grid-33	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.0	0.00	8.33	0.0
2grid-44	1	0.0	0.00	0.00	0.0	1	0.1	0.00	8.33	0.1	1	0.2	0.00	12.50	0.2
2grid-55	1	0.0	0.00	0.00	0.0	1	0.2	0.00	2.50	0.2	25	4.0	0.00	7.50	3.9
2grid-66	1	1.1	0.00	1.67	1.0	77	4.9	0.00	1.67	4.6	271	51.3	0.00	1.67	49.7
2grid-77	19	8.5	0.00	1.19	8.2	13	13.3	0.00	5.95	13.0	289	875.1	0.00	4.76	858.7
2grid-88	7,393	999.8	5.56	5.56	951.0	8,837	999.8	0.69	0.69	959.4	135	56.6	0.00	12.50	55.0
3333rand-100-400	2,297	1,006.1	2.12	2.12	975.0	8,287	999.8	1.57	1.57	949.9	895	375.6	0.00	6.27	364.8
333rand-100-091	29	3.4	0.00	2.27	3.2	1	0.5	0.00	3.52	0.5	37	8.4	0.00	4.19	8.2
3grid-20-091	1	0.5	0.00	2.04	0.5	1	0.0	0.00	0.00	0.0	49	11.1	0.00	2.38	10.8
444rand-100-398	71	20.0	0.00	1.17	19.1	63	73.9	0.00	2.57	72.1	199	421.4	0.00	5.81	413.5
clique-20	1	0.7	0.00	0.00	0.7	1	0.3	0.00	0.00	0.3	1	0.3	0.00	0.00	0.3
clique-30	1	1.6	0.00	0.00	1.6	7	11.3	0.00	0.65	11.1	3	10.1	0.00	4.79	9.9
clique-40	3	23.5	0.00	0.02	23.1	1	8.4	0.00	0.00	8.2	3	28.1	0.00	5.69	27.7
clique-50	5	31.2	0.00	0.01	30.7	1	25.7	0.00	0.00	25.3	5	40.3	0.00	3.07	39.7
clique-60	1	16.1	0.00	0.00	15.8	15	63.8	0.00	0.56	62.7	19	65.9	0.00	7.99	64.7
clique-70	19	56.0	0.00	0.13	55.1	1,173	1,003.5	1.47	1.91	980.5	27	33.9	0.00	5.92	33.2
clique-80	21	113.1	0.00	0.13	111.4	1,307	1,000.1	1.74	2.01	977.9	279	268.9	0.00	5.78	262.8
clique-90	27	258.2	0.00	0.14	254.8	921	1,000.3	1.63	1.89	978.9	209	308.8	0.00	6.12	302.5
random-20-109	3	3.9	0.00	3.84	3.8	3	4.2	0.00	1.51	4.1	3	5.6	0.00	3.62	5.4
random-30-018	23	94.4	0.00	1.13	92.7	17	60.4	0.00	1.04	59.3	27	104.1	0.00	4.01	102.1
random-40-047	145	1,000.5	0.23	1.05	985.6	137	1,001.9	0.16	2.65	987.1	129	1,004.9	1.18	2.60	990.0
random-50-229	99	1,001.3	0.52	2.37	988.9	99	1,005.4	0.84	2.58	992.8	91	1,008.1	1.53	2.11	995.7
random-60-935	357	1,000.7	0.67	1.71	982.1	285	1,000.0	1.08	2.19	983.2	229	1,003.6	1.25	2.47	987.6
random-70-030	1,955	999.9	1.04	2.41	976.3	1,433	999.7	1.13	2.54	976.8	1,475	1,003.8	1.60	3.40	980.6
random-80-702	1,787	999.9	1.99	2.31	975.8	1,825	1,000.1	1.96	2.30	975.6	1,727	1,000.0	3.04	3.39	976.2
random-90-099	1,525	1,000.2	2.27	2.53	976.8	1,507	1,000.1	1.94	2.21	976.8	1,453	1,000.0	2.80	3.10	977.0

gap found at root, and total time spent solely on solving SDPs. When $S = S_0$, almost all of the 2-grid and clique graphs which form W yield problems which are solvable in under 260 seconds, and where less than 30 tree nodes are searched. In fact, a majority of them are solved within 5 tree nodes, which implies that the SDP relaxation at the root is a very strong approximation of the optimum solution, and possibly produces a matrix which is close to "integral". The small number of search nodes, along with the small initial integrality gap found at the root of larger problems such as `clique-80`, implies that our heuristic produces feasible solutions which are close to the optimum. Note, however, that the total time taken increases quickly with the number of vertices: the time roughly doubles when successively comparing `clique-60` through `clique-90`. The amount of time dedicated to solving SDPs appears to be at least 90% of the total time spent on any of the above problems, and in the larger clique graphs, the percentage is at least 98%.

The other grid graphs yield peculiar behavior. While the 3-grid and random 3-dimensional grids are solvable in under 20 seconds each, the instances `2grid-88` and `3333rand-100-400` show anomalous behavior. The gaps found at both root nodes persist until the maximum time limit of 1000 seconds is reached. In each case, many thousands of nodes are searched, averaging less than 1 second per subproblem in which multiple SDPs are solved. Upon further inspection, we found that the initial bounds at each root are equal to the corresponding optimal solution value. However, the resulting matrices were highly fractional, and our heuristic could not improve upon the sub-optimal incumbent found at the root. This behavior is also found when $S = S_1$, but is mitigated when restricted to $S = S_2$.

The completely dense random graphs `random-20-109` through `random-90-099` show us that as n grows, the problems become successively harder to solve to optimality. In particular, while `random-20-109` is solvable in 3.9 seconds over only 3 search nodes, the instance `random-30-018` takes an order of magnitude longer to solve, and already the problem `random-40-047` is not solvable to optimality in under 1000 seconds. On the other hand, each problem with $n \geq 30$ is tightly bounded by the SDP relaxations solved at the root, as the relative gaps are all roughly under 2.5% to start. The improvement

Table 4.2: $k = 3$, α and β tight

	S_0					S_1					S_2				
W	# Nodes	CPU (s)	F. Gap (%)	R. Gap (%)	SDP (s)	# Nodes	CPU (s)	F. Gap (%)	R. Gap (%)	SDP (s)	# Nodes	CPU (s)	F. Gap (%)	R. Gap (%)	SDP (s)
2grid-33	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	7	0.1	0.00	8.33	0.1
2grid-44	1	0.0	0.00	0.00	0.0	5	0.1	0.00	8.33	0.1	5	0.5	0.00	8.33	0.5
2grid-55	1	0.0	0.00	0.00	0.0	5	0.6	0.00	7.50	0.5	5	1.4	0.00	7.50	1.4
2grid-66	13	1.6	0.00	1.67	1.5	97	11.9	0.00	3.33	11.2	29	4.7	0.00	1.67	4.5
2grid-77	45	8.6	0.00	1.19	7.9	323	40.7	0.00	5.95	37.2	63	144.8	0.00	14.29	142.1
2grid-88	7,319	999.8	5.56	5.56	951.7	2,669	999.9	25.00	25.00	967.7	135	56.0	0.00	51.39	54.4
3333rand-100-400	2,289	1,003.3	5.46	5.46	973.0	8,309	1,000.0	28.03	28.03	951.4	895	374.4	0.00	57.74	363.9
333rand-100-091	3	0.6	0.00	2.51	0.6	23	2.7	0.00	5.95	2.6	25	6.6	0.00	11.00	6.4
3grid-20-091	9	1.1	0.00	3.74	1.0	5	0.7	0.00	3.74	0.7	11	3.5	0.00	13.27	3.4
444rand-100-398	255	152.9	0.00	3.44	146.9	25	11.2	0.00	47.75	10.5	191	656.8	0.00	30.89	642.8
clique-20	1	0.6	0.00	0.00	0.6	1	0.3	0.00	0.00	0.3	1	0.4	0.00	0.00	0.4
clique-30	1	1.6	0.00	0.00	1.5	7	11.2	0.00	0.37	10.9	5	9.6	0.00	38.66	9.4
clique-40	3	23.9	0.00	0.02	23.5	1	8.5	0.00	0.00	8.3	3	28.5	0.00	5.69	28.0
clique-50	5	30.3	0.00	0.01	29.8	1	26.2	0.00	0.00	25.8	5	40.1	0.00	2.75	39.5
clique-60	1	16.7	0.00	0.00	16.4	15	65.7	0.00	0.53	64.5	19	64.2	0.00	3.94	63.1
clique-70	19	56.8	0.00	12.42	55.9	1,173	1,004.5	1.48	69.00	981.6	27	34.3	0.00	36.54	33.7
clique-80	21	113.0	0.00	0.13	111.3	1,305	999.9	31.29	31.48	977.7	279	269.2	0.00	69.71	263.2
clique-90	27	260.1	0.00	8.12	256.7	921	999.8	2.28	2.54	979.1	209	308.7	0.00	100.00	302.5
random-20-109	3	3.4	0.00	4.58	3.2	3	4.0	0.00	0.01	3.8	3	4.1	0.00	3.64	3.9
random-30-018	33	129.3	0.00	0.95	126.4	15	60.6	0.00	0.62	59.2	31	120.7	0.00	37.58	117.8
random-40-047	143	1,001.0	0.21	1.40	981.9	155	928.7	0.00	2.62	910.9	121	1,004.7	0.73	2.17	985.4
random-50-229	95	1,007.8	0.57	2.91	991.1	97	1,000.1	1.22	1.97	982.9	89	1,000.2	1.74	2.46	983.1
random-60-935	357	1,002.5	0.82	2.13	980.1	293	1,003.2	1.09	2.52	981.9	239	1,007.9	1.38	2.50	987.1
random-70-030	2,017	999.7	1.04	2.26	976.3	1,453	1,007.0	1.18	70.63	983.9	1,475	1,001.8	1.48	36.58	978.6
random-80-702	1,789	1,000.1	2.18	2.70	975.6	1,821	1,000.4	34.42	34.64	976.0	1,731	999.7	65.73	65.86	976.2
random-90-099	1,523	1,000.0	1.88	2.92	976.7	1,505	999.6	2.16	3.01	977.1	1,453	1,000.0	100.00	100.00	976.3

upon the starting integrality gap, unfortunately, is very slow; for instance our algorithm lowers the initial gap of 2.53% for `random-90-099` to 2.27% after searching 1,525 tree nodes in 1000 seconds—only a 10.3% drop in percentage points. Furthermore, as n grows, the final integrality gap is increasing, whereas the number of nodes searched is decreasing; this is expected, since the time limit is fixed and larger SDPs take longer to solve.

Comparing instances with only S varying—keeping $k = 3$, and α and β trivial—we see that many intial integrality gaps become larger as S becomes more restrictive, as S changes from S_0 to S_1, and again to S_2. For example, the root gap percentages increase in this order from 0.14% to 1.89% to 6.12% in the instance `clique-90`. Similarly, the total CPU time, the number of B&B nodes, and the average time spent per node, are all increasing for `444rand-100-398`. On the other hand, many problems remain solvable, or unsolvable, within the 1000 seconds time limit, and those which are solvable have running times which are roughly of the same order of magnitude (*e.g.* the smaller random graph instances, or the clique graphs for $n \leq 60$). However, the clique graphs with $n \geq 70$ become unsolvable by our algorithm when comparing S_0 to S_1, but then become solvable again when considering S_2.

Now, comparing $k = 3$ with the pair (α, β) as either trivial or tight (*i.e.* Tables 4.1 and 4.2), we first notice that almost every problem remains solvable, or unsolvable, and the running times are nearly identical. Furthermore, the anomalous behavior found for the few instances with α and β trivial, is persisent. However, the larger clique and random graph instances (*i.e.* $n \geq 70$), especially when $S = S_1$ or S_2, exhibit very large initial gaps at the corresponding root nodes. Since in most cases, any positive final gaps between differing (α, β) instances are comparable, and those solvable for trivial (α, β) remain solvable for tight capacities, we conclude that the initial SDP bounds are just as strong, but the heuristic is not generating good enough feasible solutions. This is expected, as the problems are becoming more restrictive.

If we let $k = 5$ and (α, β) be trivial, Table 4.3 shows us that all the grid graphs—2-grid, 3-grid, and random grids—are solvable at the root, barring `3333rand-100-400`

Table 4.3: $k = 5$, α and β trivial

	S_0					S_1					S_2				
W	# Nodes	CPU (s)	F. Gap (%)	R. Gap (%)	SDP (s)	# Nodes	CPU (s)	F. Gap (%)	R. Gap (%)	SDP (s)	# Nodes	CPU (s)	F. Gap (%)	R. Gap (%)	SDP (s)
2grid-44	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0
2grid-55	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0
2grid-66	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0
2grid-77	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0
2grid-88	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0
3333rand-100-400	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	15,031	999.8	0.01	0.01	939.9
333rand-100-091	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0
3grid-20-091	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0
444rand-100-398	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0
clique-20	1	0.6	0.00	0.00	0.6	5	4.8	0.00	0.32	4.7	3	2.4	0.00	0.65	2.3
clique-30	1	2.4	0.00	0.00	2.4	1	2.5	0.00	0.00	2.4	1	2.8	0.00	0.00	2.8
clique-40	1	7.4	0.00	0.00	7.3	1	12.4	0.00	0.00	12.2	3	19.1	0.00	1.31	18.8
clique-50	1	32.1	0.00	0.00	31.7	19	128.8	0.00	0.22	126.8	7	95.0	0.00	1.00	93.7
clique-60	1	38.7	0.00	0.00	38.2	5	67.9	0.00	0.30	66.9	7	78.1	0.00	1.23	77.0
clique-70	117	748.4	0.00	0.16	736.8	217	1,002.6	0.63	0.70	986.3	213	1,001.7	1.67	1.89	985.5
clique-80	73	1,007.6	0.12	0.36	994.0	131	1,003.5	0.51	0.60	989.1	103	1,002.4	0.71	0.80	988.8
clique-90	35	1,002.2	0.07	0.27	988.6	65	999.9	0.59	0.62	987.7	53	1,017.1	0.90	0.97	1,003.7
random-20-109	3	5.9	0.00	2.12	5.7	1	1.3	0.00	0.00	1.2	3	5.9	0.00	1.29	5.7
random-30-018	53	381.3	0.00	1.46	374.6	47	307.1	0.00	1.66	301.7	37	248.1	0.00	1.39	243.8
random-40-047	53	1,011.9	0.79	1.72	999.5	57	1,007.3	0.75	2.04	995.3	55	1,006.3	0.90	1.86	994.4
random-50-229	49	1,016.8	1.53	2.19	1,005.9	45	1,012.0	1.62	2.10	1,001.2	51	1,020.3	1.80	2.05	1,009.2
random-60-935	85	1,009.2	1.75	1.98	997.1	83	1,011.8	1.14	1.95	999.7	77	1,027.1	1.73	2.17	1,015.2
random-70-030	871	1,000.4	1.78	2.02	980.2	843	1,000.0	1.72	2.21	979.8	831	1,000.4	2.37	3.16	980.4
random-80-702	635	1,000.9	1.80	2.18	980.9	623	1,000.1	1.93	2.25	979.9	609	999.9	2.27	2.87	980.1
random-90-099	453	1,001.4	1.79	2.28	982.0	439	1,001.3	1.89	2.24	982.1	433	1,001.2	2.04	2.57	982.2

Table 4.4: $k = 5$, α and β tight

	S_0					S_1					S_2				
W	# Nodes	CPU (s)	F. Gap (%)	R. Gap (%)	SDP (s)	# Nodes	CPU (s)	F. Gap (%)	R. Gap (%)	SDP (s)	# Nodes	CPU (s)	F. Gap (%)	R. Gap (%)	SDP (s)
2grid˙44.79	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0
2grid˙55.79	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.2	0.00	2.50	0.2
2grid˙66.79	1	2.7	0.00	100.00	2.6	1	2.4	0.00	100.00	2.3	7	1.5	0.00	1.67	1.4
2grid˙77.79	11,949	999.8	1.19	1.19	896.7	1	0.0	0.00	0.00	0.0	8,225	999.8	2.38	2.38	903.1
2grid˙88.79	20,603	999.8	0.69	0.69	937.7	18,221	999.9	2.08	2.08	945.8	12,309	999.9	1.39	1.39	948.7
3333rand˙100˙400.79	4,355	1,000.6	0.85	0.85	889.1	16,765	999.8	0.48	0.48	942.9	1	0.0	0.00	0.00	0.0
333rand˙100˙091.79	1	0.0	0.00	0.00	0.0	1	0.4	0.00	0.74	0.4	1	0.0	0.00	0.00	0.0
3grid˙20˙091.79	1	0.0	0.00	0.00	0.0	7	0.5	0.00	3.06	0.4	1	0.4	0.00	0.68	0.4
444rand˙100˙398.79	4,565	1,000.0	1.82	1.82	871.3	3,759	1,000.0	2.08	2.08	871.6	1	0.0	0.00	0.00	0.0
clique˙20.79	1	0.6	0.00	0.00	0.6	5	4.6	0.00	0.16	4.4	1	1.8	0.00	0.00	1.7
clique˙30.79	1	2.4	0.00	0.00	2.4	1	2.5	0.00	0.00	2.5	1	2.6	0.00	0.00	2.5
clique˙40.79	1	7.4	0.00	0.00	7.3	3	14.4	0.00	3.40	14.2	3	19.1	0.00	3.83	18.7
clique˙50.79	1	30.2	0.00	0.00	29.8	17	125.6	0.00	1.24	123.6	7	94.7	0.00	1.51	93.3
clique˙60.79	1	38.2	0.00	0.00	37.6	5	66.3	0.00	3.48	65.3	7	78.3	0.00	2.39	77.2
clique˙70.79	117	612.4	0.00	4.03	602.9	217	1,002.4	1.63	2.47	986.1	213	1,001.6	1.63	3.24	985.4
clique˙80.79	73	1,007.1	6.37	7.15	993.5	131	1,002.6	3.95	4.67	988.2	103	1,000.8	4.35	4.43	987.2
clique˙90.79	35	1,004.1	3.51	5.27	990.5	65	1,001.4	2.80	3.36	989.3	53	1,010.9	0.82	3.57	997.5
random˙20˙109.79	5	5.3	0.00	2.12	5.2	1	1.6	0.00	0.00	1.6	3	7.2	0.00	1.28	6.9
random˙30˙018.79	61	417.2	0.00	1.03	408.3	53	254.0	0.00	1.67	248.8	37	261.2	0.00	1.38	255.7
random˙40˙047.79	57	1,009.7	0.76	1.40	994.5	55	1,005.3	0.96	1.60	990.1	55	1,012.2	0.78	1.62	996.8
random˙50˙229.79	47	1,012.8	1.14	1.81	998.9	47	1,018.5	1.92	2.13	1,004.1	47	1,007.3	1.52	1.76	993.1
random˙60˙935.79	87	1,006.5	1.38	2.19	990.8	81	1,002.9	2.46	2.70	987.5	71	1,000.0	2.00	2.39	984.9
random˙70˙030.79	871	1,001.8	1.68	2.26	981.6	843	1,001.2	1.45	2.69	981.0	837	1,000.1	2.10	3.15	979.9
random˙80˙702.79	633	1,000.8	1.88	2.97	980.8	623	1,000.2	2.06	2.86	980.3	611	1,000.9	2.13	3.06	980.9
random˙90˙099.79	451	999.7	1.98	3.09	980.6	439	1,000.3	1.65	2.97	981.1	435	1,001.0	2.44	3.01	981.8

for S_2. In fact, in each case, the heuristic found the optimum solution during the preprocessing phase, and thus no time was spent solving any SDP relaxations. Comparing $k = 5$ with $k = 3$ (Table 4.1), we see that the clique graphs with $n \leq 60$ are solvable in similar amounts of time, and in about half of the cases no branching is required, *i.e.* the solution is found at the root node. On the other hand, for $k = 5$, the remaining clique instances take much more time to solve, and many are left unsolved within the 1000 second limit. Furthermore, when restricted to the random graph instances for $k = 5$, the number of search nodes is cut by a factor of 2–4, and the improvement from initial gap to final gap percentages is degraded. For example, when W is random-60-030 and $S = S_1$, the problem with $k = 5$ searched 83 nodes in 1011.8 seconds, bringing the gap down from 1.95% to 1.14% for a decrease of 41.6%, while for $k = 3$, number of nodes searched was 285, and the gap was improved from 2.19% to 1.08% for a decrease of 50.7%. Of further note, is the increase in time spent solving SDPs; for instance, when $k = 3$ and W is `clique-70`, the 1000 second limit is reached and 980.5 seconds of that time were spent solving SDPs, whereas for $k = 5$, the amount of time spent solving SDPs was 986.3.

As with the comparison of trivial and tight capacities for the case $k = 3$, the only significant differences in statistics between Tables 4.3 and 4.4 for $k = 5$ are the changes in initial integrality gaps. Again, this may be due to our heuristic not finding good solutions at the outset of the SBC algorithm. Further, the anomalous behavior found in larger grid graphs has returned, which is explained by the fact that our feasible solution heuristic is weak in particular when the instance is highly restrictive, *i.e.* for tight (α, β) and non-trivial S. Otherwise, the running times and the number of search nodes of each instance are comparable between the two cases of capacities α and β.

For the case $k = 7$ (*cf.* Table 4.5), we notice that as in the previous case with $k = 5$, all grid graphs are easy to solve, *i.e.* the heuristic finds the optimum at the root during a preprocessing phase, and bypasses any need to solve the more cumbersome SDP relaxations. Also similar is the fact that clique graphs for $n \geq 70$ cannot be solved to optimality in under 1000 seconds. We notice that for instance `clique-70`, and for $S = S_0$ or S_1, the total number of search nodes for $k = 7$ is about half of the number

Table 4.5: $k = 7$, α and β trivial

	S_0					S_1					S_2				
W	# Nodes	CPU (s)	F. Gap (%)	R. Gap (%)	SDP (s)	# Nodes	CPU (s)	F. Gap (%)	R. Gap (%)	SDP (s)	# Nodes	CPU (s)	F. Gap (%)	R. Gap (%)	SDP (s)
2grid-44	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0
2grid-55	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0
2grid-66	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0
2grid-77	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0
2grid-88	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0
3333rand-100-400	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0
333rand-100-091	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0
3grid-20-091	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0
444rand-100-398	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0
clique-20	5	4.8	0.00	0.23	4.7	11	6.8	0.00	0.77	6.6	15	8.1	0.00	0.85	7.8
clique-30	1	19.8	0.00	0.20	19.6	9	45.2	0.00	0.53	44.9	5	32.9	0.00	0.96	32.7
clique-40	11	100.3	0.00	0.54	99.6	1	22.8	0.00	0.00	22.7	1	16.1	0.00	0.00	16.0
clique-50	7	157.3	0.00	0.63	156.3	13	135.6	0.00	0.40	134.6	7	92.6	0.00	0.79	92.0
clique-60	61	1,092.4	0.00	0.39	1,086.0	9	285.7	0.00	0.30	283.8	27	502.2	0.00	1.51	498.3
clique-70	95	1,003.8	0.14	1.13	992.9	117	1,003.3	0.46	0.81	992.4	109	1,003.0	0.49	0.81	992.6
clique-80	45	1,017.7	0.08	1.22	1,007.9	57	1,008.7	0.36	0.63	999.6	53	1,002.4	0.50	1.08	993.5
clique-90	21	1,048.7	0.44	1.64	1,039.2	45	1,013.4	0.37	1.03	1,003.9	31	1,014.8	0.49	1.29	1,005.4
random-20-109	1	1.3	0.00	0.00	1.3	1	1.0	0.00	0.00	0.9	7	5.6	0.00	0.59	5.4
random-30-018	11	154.5	0.00	1.15	153.2	15	153.3	0.00	0.86	152.0	75	555.3	0.00	1.35	549.5
random-40-047	27	1,004.1	1.12	1.28	997.3	27	1,024.0	0.69	1.57	1,017.3	27	1,020.4	1.56	1.80	1,013.6
random-50-229	9	1,115.5	1.49	2.05	1,112.0	9	1,156.2	1.29	2.11	1,152.7	5	1,003.7	1.68	1.95	1,000.7
random-60-935	15	1,098.1	1.36	1.77	1,095.5	13	1,030.3	1.94	2.15	1,027.8	11	1,080.9	1.80	2.11	1,078.4
random-70-030	377	1,001.2	1.33	2.48	987.5	363	1,001.4	1.67	2.13	987.8	367	1,000.4	1.50	2.21	986.7
random-80-702	229	1,003.6	1.33	2.31	991.2	223	1,001.5	1.80	2.51	989.3	217	1,002.0	1.73	2.43	989.7
random-90-099	143	1,005.7	1.77	2.46	994.5	143	1,003.4	1.68	2.38	992.2	139	1,003.8	1.69	2.40	992.6

Table 4.6: $k = 7$, α and β tight

	S_0					S_1					S_2				
W	# Nodes	CPU (s)	F. Gap (%)	R. Gap (%)	SDP (s)	# Nodes	CPU (s)	F. Gap (%)	R. Gap (%)	SDP (s)	# Nodes	CPU (s)	F. Gap (%)	R. Gap (%)	SDP (s)
2grid-44	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0
2grid-55	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0
2grid-66	22,465	994.3	0.00	100.00	843.9	20,805	999.8	100.00	100.00	776.8	19,931	999.7	100.00	100.00	818.6
2grid-77	2,933	1,000.2	3.57	3.57	934.2	12,373	999.7	2.38	2.38	849.6	2,939	999.8	1.19	1.19	910.7
2grid-88	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	2,653	999.9	0.69	0.69	927.6
3333rand-100-400	13,007	999.9	0.57	0.57	947.8	13,145	999.8	0.77	0.77	946.8	1,805	1,002.7	0.46	0.46	972.4
333rand-100-091	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0
3grid-20-091	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0	1	0.0	0.00	0.00	0.0
444rand-100-398	1,939	1,000.1	0.60	0.60	932.9	1,409	1,000.4	2.37	2.37	954.4	1,269	1,000.2	0.72	0.72	957.8
clique-20	5	2.6	0.00	0.38	2.5	11	6.8	0.00	0.77	6.6	11	6.5	0.00	0.85	6.3
clique-30	13	61.2	0.00	0.39	60.7	7	53.1	0.00	0.41	52.7	7	36.7	0.00	1.56	36.5
clique-40	5	50.5	0.00	1.40	50.2	1	22.2	0.00	0.00	22.1	1	17.3	0.00	0.00	17.2
clique-50	7	179.3	0.00	1.34	178.2	15	142.5	0.00	2.60	141.5	5	86.2	0.00	2.06	85.5
clique-60	69	898.9	0.00	2.13	892.5	9	305.7	0.00	2.00	303.7	27	502.5	0.00	2.44	498.6
clique-70	95	1,003.1	2.10	4.15	992.2	117	1,003.9	0.32	2.93	993.3	109	1,005.0	2.77	3.93	994.6
clique-80	45	1,017.4	2.61	4.82	1,007.7	57	1,008.3	1.89	3.73	999.3	53	1,003.1	0.89	4.04	994.2
clique-90	21	1,048.8	1.25	3.90	1,039.3	45	1,012.9	1.92	3.72	1,003.4	31	1,016.3	0.63	2.76	1,006.9
random-20-109	1	1.5	0.00	0.00	1.4	1	1.0	0.00	0.00	0.9	7	5.5	0.00	0.59	5.3
random-30-018	15	166.3	0.00	1.20	164.5	9	117.1	0.00	1.19	116.0	77	542.6	0.00	1.53	536.0
random-40-047	27	1,009.2	1.07	1.82	1,001.4	27	1,033.5	0.92	1.43	1,025.7	27	1,034.0	1.31	1.96	1,025.9
random-50-229	7	1,109.8	0.86	1.66	1,105.8	7	1,031.3	1.32	2.38	1,027.6	7	1,015.2	1.46	1.95	1,011.6
random-60-935	15	1,114.2	1.30	1.46	1,111.1	13	1,009.5	1.97	2.46	1,006.7	11	1,080.3	1.58	1.93	1,077.3
random-70-030	377	999.7	1.53	2.47	985.8	363	1,000.1	1.62	2.22	986.4	367	1,000.3	1.69	2.53	986.6
random-80-702	227	999.9	1.70	2.78	987.5	223	1,003.0	1.73	2.41	990.6	217	1,001.8	1.90	2.59	989.5
random-90-099	143	999.8	1.82	2.67	988.7	143	1,004.2	1.98	2.72	993.1	139	1,005.9	1.73	2.69	994.8

found in Table 4.3, where $k = 5$. Furthermore, the total time spent solving SDPs here has increased by about 6 seconds, and the gap improvement from root to finish is actually better. This latter claim is supported by the fact that as k grows, more weighted edges of W will appear in the k-cut (see also Turán's Theorem 2.3.1).

If we look at the last table (Table 4.6), where $k = 7$ and the capacities α and β are tight, we see the same phenomenon as in the previous comparisons: many grid graphs yield a small gap initially at the root, but then this is not reduced even after solving thousands of subproblems. In fact, when W is chosen as `2grid-66` and $S = S_1$ or S_2, no feasible solution is ever found, hence the final gap of 100%. This further underscores the need to improve upon our heuristic procedure when generating feasible solutions for tightly constrained instances of **capacitated k-partition**. Otherwise, the solvable problems remain solvable, and the unsolved problems yield comparable gaps and number of search nodes when comparing trivial *vs.* tight capacities (*i.e.* Tables 4.5 and 4.6).

To recap, we noticed that as k grows, the problems become harder, in that both the running times are longer, and the unsolvable instances yield fewer search nodes. The relative gaps shown may not be directly comparable, as the number of weighted edges in a k'-cut for $k' > k$ tends to be larger in total than that of a k-cut, hence the objective values are larger and relative gap percentages are smaller. However, in order to reduce the initial gap found at the root node, a robust heuristic is needed. In particular, whenever α and β are non-trivial, we saw some cases, over all k and S, where no improvement was made, yet the SDP relaxation bound had not necessarily degraded. In order to further strengthen the SDP bounds, it may be necessary to explore the strength of the flower inequalities, since they were rarely generated in our experiments, and the strength of the star cover inequalities, as they were omitted from our tests. Of particular note, is the fact that the structure of W highly affects the performance of our algorithm, since random graphs are extremely difficult to solve to optimality, yet the grid graphs—barring the anomalies discussed—are solvable by our heuristic procedure independent of the choices of k, (α, β), and S.

We end this section by noting that our results above are somewhat comparable

to the experiments found in Anjos *et al.* [5], where **maximum k-cut** instances were solved to optimality using a semidefinite branch-and-cut algorithm similar to ours, only the SDP solver was a bundle method. Here, we restrict attention in our tables to the first subdivision of columns ($S = S_0$), for (α, β) trivial. They solved many problems where negative edge weights were allowed, but they also solved instances with the clique graphs from $n = 20$ up to $n = 70$ and random graphs from $n = 20$ up to $n = 50$, all for $k = 3$, 5, and 7—note, the latter were generated with different seeds and are not directly comparable. For smaller k and n values, their algorithm was faster when solving the clique and random graphs than ours. For instance, we solved `random-30-018` in 94.4 seconds over 23 nodes, while Anjos *et al.* reportedly solved a similar problem random-30-k=3 in 26 seconds over 173 nodes. We solve clique-40 in 23.5 seconds over 3 nodes, while they solve it in 7 seconds over 15 nodes.

On the contrary, our algorithm is faster when solving larger clique instances for any k, and small random graphs for $k = 5$ and $k = 7$. In particular, for $k = 3$, `clique-70` was solved by Anjos *et al.* in 830 seconds over 439 nodes, whereas we solved it in 56.0 seconds over 19 nodes. Further, for $k = 7$, `clique-50` was solved by Anjos *et al.* in 2,179 seconds over 1,853 nodes, whereas we solved the problem in 157.3 seconds over 7 nodes. Although not entirely comparable, the statistics for our instance `random-30-047` on $n = 30$ vertices with $k = 7$ *vs.* their instance `random-30-k=3` on 30 vertices for $k = 7$, demonstrate the biggest differences: we solve our instance in roughly 2.5 minutes over 11 nodes, while Anjos *et al.* solve theirs in about 2.5 hours over almost 29,000 nodes. It should be noted that the tests in Anjos *et al.* [5] were performed in a 32-bit environment with a 2.3 GHz Intel Xeon processor, while we performed ours in a 64-bit environment on a 3.06 GHz Intel Core i7 processor.

4.3 Case study: ICS-2013

In this section, we discuss an application of our **capacitated k-partition** algorithm to the 13th Annual INFORMS Computing Society Conference scheduling problem, which we denote ICS-2013.

4.3.1 Data set and parameter choices

The organizers of ICS-2013 were kind enough to provide the data set to their conference. Their problem was to schedule $n = 61$ sessions, each consisting of a handful of presentations which were given in an equal-length sequential order, into $k = 9$ time slots, so as to not cause any time conflicts and to further structure serial talks in a logical fashion—*e.g.* into multiple tracks. The number of time slots devoted to these technical talks—here we assume that the plenary and intermissions are handled manually outside of our algorithm—forced bounds on the size of each to be very close, *i.e.* $\alpha = 5$ and $\beta = 7$ (see Proposition 2.1.3 for a brief note on tightening of such parameters). Other information was also given, which included the session chairpersons, and the titles, abstracts, and up to five keywords associated with individual talks.

To form the cost function, we first aggregated all appropriate text extracted from the title, abstract, and keyword set associated with each talk, into a single document for each session. Then these files were preprocessed and subsequently formed into a cost matrix $W \in [0,1]^{n\times n}$ by feeding them into the *Text Mining Generator* or TMG [29] for Matlab (see §A).

The remaining parameter S of input tuple $(n, k, \alpha, \beta, S, W)$ for **capacitated k-partition** is chosen to encode all pairs of participants known to be in conflict if appearing in parallel time slots. In other words, we first found all persons (not necessarily speakers) associated with a session $s \in [n]$, A_s, and let S be the intersection graph of set system $(\cup_s\{A_s : s \in [n]\})$. This graph turns out to have 132 of the possible 1830, which is 7.2% dense. With k being larger than that of our random tests, α and β tight, and S somewhat restrictive, the previous section's experiments tell us that finding an exact solution may be difficult, but the SDP relaxation may be strong at the root node.

4.3.2 ConSP schedule *vs.* actual schedule

Solving this problem with the given parameters $(n, k, \alpha, \beta, S, W)$ as mentioned previously, involves first solving the **capacitated k-partition** problem. Since this problem was tested on random data in the previous section, we adhere to the same parameter

choices, *e.g.* we allow at most 2000 cuts to be added per SDP solve. The only difference being that we let the problem run for 4 hours, as opposed to 1000 seconds. For reference, the total sum $\frac{1}{2}W \bullet J$ is 19,189.01—a trivial upper bound on the cost of any k-cut—where W is chosen to have entries W_{ij} encoding the similarity between all talks crossing the pair of sessions i and j (see Appendix A for other suggested similarity measures). Furthermore, the actual schedule of ICS-2013 was of cost 17,400.28, although with our definition of separate pairs S, this schedule is not feasible—there were 10 conflicting pairs scheduled in parallel, although the conflicts were only between author-coauthor pairs (so no presenter was assigned to be in two places at once).

After running the problem on our Intel processor—mentioned at the beginning of the chapter—on three separate occasions, each with separate seeds an allowed 4 hours CPU time, the best integrality gap observed was 0.851%, *i.e.* the relative gap between the best known lower bound (feasible solution value) 17,797.63, and the worst known upper bound (maximum of relaxed solution bounds) 17,950.32. At this instance's root node, our heuristic produced a solution of 17,776.10, and the relaxation was bounded by 17,972.50 upon branching, for an initial gap of 1.09%. Thus, our algorithm starts with a strong bound, and improves the gap by 22.2%. The best known solution is found within the first few minutes, and so the improvement rate is very slow, *i.e.* tailing off is observed early on. The progress over the 4-hour time interval in displayed in Figure 4.1.

When comparing our results with the schedulers' implementation of ICS-2013, we first note that our best feasible schedule does not permit any conflicting pairs, whereas the actual schedule did not directly mitigate author-coauthor conflicting pairs. Secondly, our schedule has a larger overall weighted k-cut, and thus lower total similarity between parallel pairs of talks. In particular, our proposed schedule amounts to roughly, a 2.3% increase in the weighted k-cut. If we convert the costs in terms of similarity—our primary interest in ConSP—the similarity of our schedule is 1,391.38, whereas the schedulers found a schedule of total similarity 1,788.93. Thus, our optimization approach to this problem yields a decrease of 22.2% in parallel similarity across all presentations. Although our methods have not provably found the optimum, we can

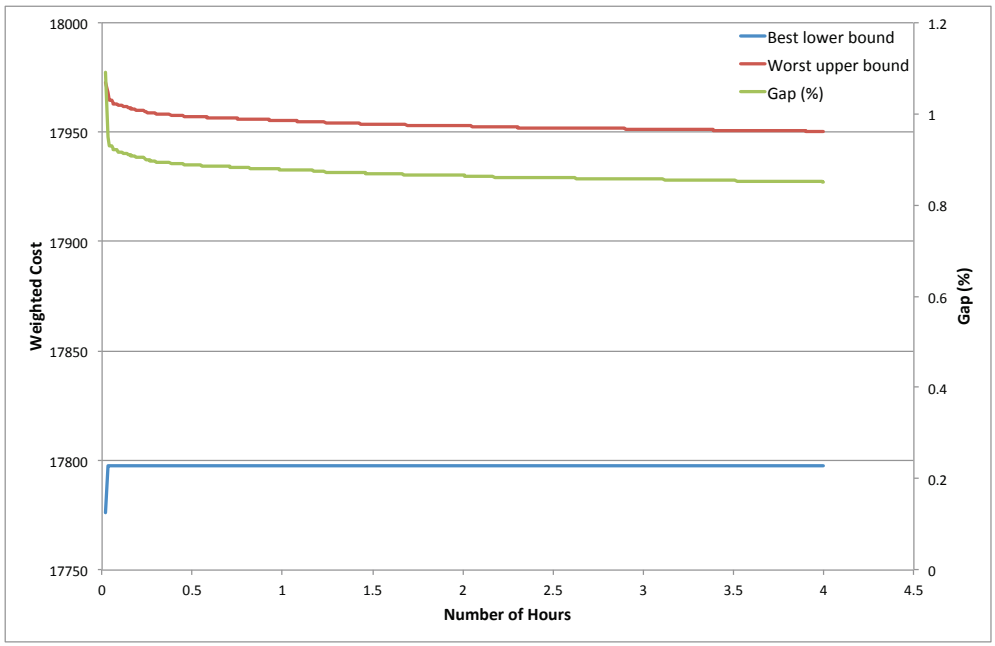

Figure 4.1: Progress of our algorithm applied to the ICS-2013 data set

indeed provide improvements, or at least suggested starting points for schedulers of a conference.

4.4 Concluding remarks

In this thesis, we proposed an algorithmic framework for scheduling conferences. We started with a symmetric objective which is well-suited for virtually all participants of a conference: minimizing similarity between parallel presentations. This novel, yet simple measure allowed for a reformulation of the problem into a more familiar setting which calls for finding particularly structured partitions of a set of sessions. The known properties of related combinatorial optimization problems aided in our formulation of the scheduling problem as a constrained variant of **maximum k-cut**—or **minimum k-partition**—which we denoted **capacitated k-partition**. We proved that this new, more generic problem is not only $\mathcal{NP}$-hard to optimize, but testing for feasibility of the problem is $\mathcal{NP}$-complete. We also demonstrated the strength of a particular semidefinite programming relaxation of the problem, and proved that particular valid inequalities of the corresponding polyhedron were automatically satisfied by such a relaxation.

The algorithm we designed was embedded in a branch-and-bound shell, consisting of a semidefinite programming relaxation equipped with a cutting plane subroutine as the bounding procedure, and a novel two-part heuristic method which aims to first construct k-partitions out of k-packings, and then attempts to modify parts of the partition to obtain feasible solutions; once the latter is obtained, better feasible solutions are sought after. The combinatorial nature of this greedy local search heuristic, along with the consecutive searches favoring "good" solutions, make this a fast and productive algorithm for improving incumbent values. This also couples well with the bottleneck procedure of solving muliple SDPs with interior point methods. Indeed, our tests have shown that the time spent in the SDP solver is at least 90% of the total amount.

After describing our algorithm in detail, we then performed experiments on a variety of randomly generated graphs used to form the cost matrices for input to the **capacitated k-partition** problem. The results showed that both the SDP bounds were strong and the heuristic procedure yielded near-optimum feasible solutions for many instances. Since **maximum k-cut** was shown to be a subclass of the problem we study, we also compared our output statistics with another branch-and-cut heuristic algorithm which also bounds instances with SDP relaxations. Although, the tailored algorithm outperformed our generic procedure for small k and n, we observe that our algorithm is competitive for cost matrices arising from dense random graphs and the more structured clique graphs.

After the initial tests were performed to fine tune parameters, we let our algorithm run for only 4 hours on a medium-sized conference known as ICS-2013. The conference was tightly constrained with little room for time slot size differences, and a modest density of separated pairs of sessions. Our algorithm resulted in the best feasible solution being found within the first few minutes, and otherwise reduced the relative gap between the worst upper bound and this incumbent to less than 1%. While optimality was not achieved, this seems a reasonably good solution for the allotted amount of time. This result tells us that the algorithm can be used as a tool for schedulers to automatically start with a provably good solution to the problem, and then implement the schedule with possible expert modifications. This combination of mathematical and

computational tools with the human interactive side is useful for constructing better schedules for those choosing to attend such events.

In the future, we plan to re-implement the algorithm so that it becomes faster in practice. In particular, we plan to exploit the message-passing parallism offered by PEBBL's branch-and-bound framework, which would allow many clusters of processors to solve many SDPs relaxations in parallel. On the other hand, we would like to explore more efficient ways of solving each individual SDP, for instance, via a bundle method such as that developed by Anjos *et al.* [5] for the **maximum k-cut** problem. It would also be interesting to explore different methods of constructing the similarity measures by incorporating more natural language processing tools. Finally, we would like to expand on the algorithmic framework designed for scheduling conference; particularly, it would be interesting if the **conference scheduling problem** incorporated other constraints, especially ones which adhere to the symmetric structure exploited in the reformulation of the problem as a clustering problem.

Part II

A United States Coast Guard resource allocation problem

Chapter 5

USCG Boat Allocation Module (BAM): A software tool

5.1 Introduction

In this chapter, we describe the current results of an ongoing project known as the Boat Allocation Module (BAM), which is a joint effort between the CG-771 team of the United States Coast Guard (USCG) and a dynamic team at the Command, Control, and Interoperability Center for Advanced Data Analysis (CCICADA).[1] This project—one of many making up the USCG's Coastal Operation Analytical Suite of Tools (COAST)—was initiated in early Spring 2012 as part of the "Engage to Excel" initiative (E2E), with the goal of designing a maritime resource allocation model along with a software tool for USCG planning analysts.[2] Upon completion in late Winter 2012, a thoroughly tested and documented software package was delivered, and a second, currently ongoing, phase of the project was begun to explore a notion of sharing allocated resources. The completed first phase is discussed below, while preliminary results of the second are deferred to the appendix.

---

[1] At CCICADA, the project was headed by Dr. Fred Roberts, CCICADA Director. Dr. Endre Boros, Director of the RUTCOR Center at Rutgers University served as Principal Scientist for the project, guiding the work of the model builders. Dr. Paul Kantor, CCICADA Director of Research, was heavily involved in all phases of the work. The model was built and documented by CCICADA Graduate Research Assistants Christie Nelson and Matthew Oster. They worked in close coordination with Coast Guard personnel under the direction of CDR Kevin Hanson, USCG, LT Patrick Ball, USCG, and LT Chad Conrad, USCG. Other researchers and experts who contributed to the projects success include: James Wojtowicz, Managing Director of CCICADA, Dr. William M. Pottenger, CCICADA Director of Transition, Dr. William Wallace, CCICADA PI at RPI, Dr. Tom Sharkey, Senior Researcher at RPI, Michael Lehocky, USCG, Kim Babcock, USCG, and Todd Aikins, USCG.

[2] Under the E2E initiative, university Centers of Excellence work closely with agencies of the Department of Homeland Security to accelerate the development of cutting edge solutions to real operational problems by collaborating from the beginning of problem formulation to transition of complete pieces.

5.1.1 Motivation and background

The U.S. Coast Guard is the primary Federal agency responsible for "maritime safety, security, and stewardship" [78]. The task force is comprised of approximately 43,000 active duty members, 7,800 reservists, 8,300 civilians, and 33,000 auxiliary personnel. In 2012, the USCG responded to almost 20,000 nationwide Search and Rescue (SAR) cases, saving more than 3,500 lives as well as $77 million in property damages [47, 49]. USCG missions such as SAR are performed primarily by *cutters* (*i.e.* marine vessels above 65 feet in length also serving as quarters for its assigned crews), *boats* (marine vessels ranging from 12 to 65 feet in length), and *aircraft* (fixed- and rotary-wing, *i.e.* airplanes and helicopters, respectively). In the sequel, we focus attention on the USCG boat fleet.

The USCG-patrolled waters of the United States are covered by 178 coastal stations. Each station is managed by a local sector, which covers at most two dozen stations. In turn, every sector belongs to a proximal command center known as a district, of which there are 9 in total. The Office of Boat Forces (OBF) annually distributes among districts the USCG inventory of small boats—currently 11 types are in use (see Figure 5.1 for a list of types)—which are further assigned to individual *home* or docking stations. This fairly large allocation problem is highly influenced by the expected demand at the station level for each of the 11 mission types. For example, at the New York City station, USCG members may spend thousands of hours ensuring the safety of associated waters by performing a Homeland Security-based mission known as Ports, Waterways, and Coastal Security (PWCS). Other mission include Search and Rescue (SAR), Aids to Navigation (AtoN), and Drug Interdiction.

Each boat type has an amount of hours the OBF prefers users not to exceed each year, after which the boat must undergo routine scheduled maintenance, *e.g.* the response boat-medium (RB-M) has a cap of 600 hours per year, whereas the response boat-small (RB-S) is allowed up to 1000 hours of use per year. A boat type also has a set of generic attributes or capabilities desired by stations. For instance, a station

with a history of Drug Interdiction operations requires one of the response or law enforcement boats, but if it is also designated as requiring a boat with a Big or Heavy Weather capability, then only the response boat-medium (RB-M) meets both criteria. Other capabilities include the ability to operate in icy conditions (LH-ICE/SH-ICE) or tactical scenarios (Tactical).

After the events of September 11, 2001, the stations saw an increase in operational requirements [79], which has subsequently led to emergency boat acquisitions. For instance, motor lifeboats (MLB) are expected to jump in number from 106 (2009 data; *cf.* Figure 5.1) to 170 by the year 2015 [48]. Although such acquisitions are necessary to meet increased requirements and replace outdated vessels, they are expensive and do not necessarily provide a complete solution to the overall allocation problem. We CCICADA and the USCG team CG-771 of analysts and operational experts worked together to develop an optimization approach to the annual boat allocation problem by building a software tool, or decision support system, for use by Coast Guard analysts and planning managers. We present a model for the allocation problem known as BAM, its software implementation and adoption within the USCG, and a brief overview of an extension of the problem to "sharing" resources. First we provide a brief review of relevant literature.

5.1.2 Related works

The discipline known as Operations Research was born just prior to World War II [55], and received its first applications in the military during that war. A particularly well-known optimization topic which arises in military applications is scheduling. For example, Brown et al. [17], studied the problem of planning routine maintenance of U.S. Navy combat ships. The problem solved was that of covering so-called primary and secondary events as well as meeting maintenance and capability requirements. The schedules were also requested to conform to manually constructed schedules deemed ideal by Navy analysts, which was approximately modeled by incorporating *elastic* constraints—a constraint which may violated, but at a large cost or penalty. This type of constraint also appears in another maritime scheduling paper, Brown et al. [14], where

a decision support tool was developed for assigning U.S. Coast Guard cutters to weekly patrols in which performing scheduled maintenance and maintaining fleet readiness—a measure of expected ability to embark on a task—were primary factors in the optimization problem. A follow-up paper [16] articulates the general notion of modeling *persistence*, the desire to keep re-optimized solutions close in appearance when data inputs are slightly modified, and argues that this property is crucial to maintaining the trust of those in management positions. Other scheduling-related applications within military departments (not necessarily those in the United States) include deployment and maintenance scheduling of helicopters [24, 51, 54], procurement planning [17, 87], capital planning [15], berth assignments [56], and general boat allocation [98].

Many of the aforementioned works focus only on a small part of the general problem at hand. For instance, in [14], the scheduling of sixteen cutters to a single USCG district was addressed. The reasons for lack of generality is typically a combination of problem difficulty, limited computational resources, and agency compartmentalization. In this thesis, we approach the boat allocation problem of the USCG with generality in mind, while still including application-specific constraining factors.

The problem we address was also studied independently by Wagner and Radovilsky [98]; especially, they studied the annual allocation of the U.S. Coast fleet of boats, or marine vessels of length at most 65, subject to USCG governing policy known as the Business Rules. Their work included both a deterministic and stochastic modeling approach, as well as similar resource sharing constraint. They developed a decision support system in Microsoft Excel, and tested it on data from USCG Fiscal Year 2009 (FY09). While the problem description was identical, our modeling approaches were not. For instance, they focused on aggregate demands for mission hours at each station, while we modeled individual mission types. They extended their model to handle variability by linearizing value-at-risk constraints—typically found in portfolio optimization—whereas we handle variability by adding a "pseudo" mission meant to encode discretionary hours reserved for commanding officers. Both efforts, however, show that an academic-industry collaboration is helpful in addressing both the technical and operational aspects of military applications.

5.1.3 Our contributions

We worked with personnel from CG-771 and designed the first optimization model (aside from the indpendent paper [98]) which aids in planning yearly boat allocation. This model allocates to each station a variety of boats which meet capability requirements and other USCG Business Rules while minimizing a weighted sum of unmet mission hours. Our model is also able to provide alternative solutions in which a single boat type is shared within particular subsets of stations—potentially reducing the total amount spent on maintenance-based costs for the year. A key feature here is that the primary variables are boat hours by type, rather than by individual vessels, ultimately reducing the problem size as well as giving the OBF and commanding officers flexibity in terms of the final implementation.

After agreeing upon a suitable mixed integer programming problem (MIP) model for BAM, we implemented it in FICO's Xpress Optimization Suite. The implementation is able to read and write MS Excel files—in formats determined by CG-771 analysts—which enables the user to dynamically view real-time changes in output after each run without manually refreshing the spreadsheets. The user interface was designed to be straightforward and very flexible in terms of which parameters are tunable, including for example, linear programming (LP) relaxation solution methods. We performed tests on the software in accordance to USCG verification and validation (V&V) standards, many of which were designed and executed with a member of CG-771 present, leading to the model's accredidation within the agency. The model was also demonstrated in realtime to Rear Admiral Mark E. Butt, the Assistant Commandant for Capabilities (CG-7), among other OBF officials and USCG members. The software package is currently functioning on the CG-771 analysts' laptops, and is being used to aid in future boat allocation and planning decisions.

Aside from the model's practical contributions, we have also shown that implementing sharing (via our tool's recommendations or not) is a theoretically hard computational problem. More accurately, we prove that solving the problem description for BAM is $\mathcal{NP}$-hard. Moreover, we prove that implementing a shared solution of BAM

by "seasonally" assigning the shared boats to individual stations, *i.e.* boats can be transferred for use at a new station only at the start of a new season, is an $\mathcal{NP}$-hard optimization problem when minimizing the number of shared boats in use.

The remainder of this chapter is organized as follows. First we give the formal model of the boat allocation problem, along with a discussion of each constraint in section §5.2. The next section (§5.3) is dedicated to the software package development and testing, as well as the overall implementation within the USCG. This discussion is followed by computational tests and interpretations in section §5.4. We conclude with a few remarks and ideas for future projects in section §5.5. The accompanying Appendix B includes preliminary details of an ongoing project related to the implementation of sharing resources, including a proof of hardness.

5.2 BAM model

Here we are given a fleet of boats varying in type, and must distribute them among the set of U.S. coastal stations, so as to meet requirements dictated by the USCG Business Rules. Rather than minimizing a cost associated with boat allocations, we fix the total available funds and minimize a measure of overall demand shortfall. In particular, we optimize a weighted sum of unmet mission hours requirements, over all stations. Relaxing the demand constraints in this way allows us to find boat allocations to scenarios where limited funds would otherwise render the problem infeasible. Next we introduce the defining constraints of BAM. Note that the allocation problem spans a period of time of one fiscal year.

Let S be the set of stations, M the set of mission types, T the set of boat types, and C the set of boat type capabilities. We define a variable b_{ts} as the total number of boats of type $t \in T$ assigned to station $s \in S$, and let y_{ts} be a binary variable encoding whether or not any boat of type t is allocated to station s. Thus, if b_{ts} is to be a positive integer, then we must have:

$$y_{ts} \leq b_{ts} \qquad \text{for } t \in T, s \in S, \tag{5.1}$$

$$b_{ts} \in \mathbb{Z}_+ \qquad \text{for } t \in T, s \in S, \tag{5.2}$$

$$y_{ts} \in \{0,1\} \quad \text{for } t \in T, s \in S, \tag{5.3}$$

since having a boat of type t ($y_{ts} = 1$) implies b_{ts} is non-zero ($b_{ts} \geq 1$). If the amount of inventory of boat type $t \in T$ is denoted B_t, and each stations $s \in S$ cannot receive more than B'_t boats of type t—with $B'_t \leq B_t$—then we also have

$$b_{ts} \leq B'_t y_{ts} \quad \text{for } t \in T, s \in S, \tag{5.4}$$

$$\sum_{s \in S} b_{ts} \leq B_t \quad \text{for } t \in T. \tag{5.5}$$

Letting P_s denote pier space available at station $s \in S$, we also include the constraint

$$\sum_{t \in T} b_{ts} \leq P_s \quad \text{for } s \in S. \tag{5.6}$$

The next set of constraints encode the USCG Business Rules, policies governing a feasible allocation.

Every station $s \in S$ requires a subset of capabilities $C_s \subseteq C$, where each $c \in C_s$ must be covered by a positive number of capable boats, R_{sc}. In other words, letting $T_c \subseteq T$ denote the set of boats of type $t \in T$ with capability $c \in C$, we have the constraint:

$$\sum_{t \in T_c} b_{ts} \geq R_{sc} \quad \text{for } c \in C_s, s \in S. \tag{5.7}$$

We define a similar constraint imposing an upper bound on the number of types allowed to cover a station's designated attributes: letting R'_{sc} denote such a bound, we have

$$\sum_{t \in T_c} y_{ts} \leq R'_{sc} \quad \text{for } c \in C_s, s \in S. \tag{5.8}$$

Another variable we introduce is the total amount of time, in hours, assigned to boats. Letting $M_t \subseteq M$ be the subset of missions types any boat of type $t \in T$ is able to perform, we define h_{tsm} to be the amount of hours assigned to missions of type $m \in M$ among the b_{ts} boats of type $t \in T$ at stations $s \in S$. Now we impose bounds on the average number of hours assigned to each boat of type $t \in T$ at station $s \in S$, that is on $\sum_{m \in M_t} h_{tsm}/b_{ts}$. In particular, we have a preferred capacity of A_t per boat of type $t \in T$, but allow E_t extra hours at the station level, and a slightly smaller global surplus of $L_t E_t$, where $L_t \in [0,1]$ is a fractional parameter. This translates into the

following set of constraints:

$$\sum_{m \in M_t} h_{tsm} \leq (A_t + E_t) b_{ts} \qquad \text{for } t \in T, s \in S, \tag{5.9}$$

$$\sum_{s \in S} \sum_{m \in M_t} h_{tsm} \leq (A_t + L_t E_t) \sum_{s \in S} b_{ts} \qquad \text{for } t \in T, \tag{5.10}$$

$$h_{tsm} \geq 0 \qquad \text{for } m \in M_t, t \in T, s \in S. \tag{5.11}$$

We also impose a related lower bound at the station level, since otherwise our preliminary tests revealed that the most expensive boats assigned to stations almost always performed no missions. Letting E'_t denote a maximum deficit below A_t boats of type $t \in T$ are allowed, the constraint is as follows:

$$\sum_{m \in M_t} h_{tsm} \geq (A_t - E'_t) b_{ts} \qquad \text{for } t \in T, s \in S. \tag{5.12}$$

Every boat type $t \in T$ requires the training of capable personnel, and the annual amount of hours associated with each type's minimum is U_t. Since also any logged training hours count towards a boat's total usage, we let some $m^* \in M$ denote a "pseudo" mission called Training, and thus the following constraint arises:

$$h_{t\,s\,m^*} \geq U_t y_{ts} \qquad \text{for } t \in T, s \in S, m = m^*. \tag{5.13}$$

This encodes that fact that a station only needs training hours aboard a type if that type is allocated to the station. Note also that the previous constraints hold for $m = m^*$.

With any allocation comes a cost. Here we assume that the inventory amounts, B_t, pertain to boats that have been previously acquired and are ready for use. Thus we do not include a purchasing cost, but rather a maintenance-based cost for each boat. In particular, the model has a calendar-based maintenance cost, or *fixed cost* F_t, and an engine-based maintenance cost, or *variable cost* V_t. The latter parameter V_t is incurred per hour a boat of type $t \in T$ is underway or in operation, whereas the former F_t is incurred per allocated boat. Notice that V_t attributes to the overall cost of an allocation indepedently of F_t, *i.e.* the fixed cost is incurred per boat allocated, whether or not such boats are utilized otherwise. Furthermore, there is an increasing cost or penalty associated with assigning more than two boat types, since with each

new boat type, an increase in training per USCG member at a station is observed.[3] Thus the cost was determined to be incremental in the number of extra types, *i.e.* we define J_i to be a non-negative cost amount, for $i \geq 3$, where $\sum_{3 \leq i \leq k} J_i$ denotes the total cost of assigning exactly k distinct boat types to a station. Now by introducing a variable $z_{si} \in \{0,1\}$ which, for any $s \in S$ and solution $y \in \{0,1\}^{T \times S}$, is to be of value 1 whenever $i \leq \sum_{t \in T} y_{ts}$, and 0 otherwise, the cost constraint is as follows for any given budget D in U.S. dollars:

$$\sum_{s \in S} \left(\left(\sum_{t \in T} F_t b_{ts} + V_t \sum_{m \in M_t} h_{tsm} \right) + \sum_{i=3}^{|T|} J_i z_{si} \right) \leq D. \tag{5.14}$$

The variables z_{si} are encoded properly by enforcing

$$\sum_{i=3}^{|T|} z_{si} \geq \sum_{t \in T} y_{ts} - 2 \quad \text{for } s \in S, \tag{5.15}$$

$$z_{s\,i-1} \geq z_{si} \quad \text{for } s \in S, i = 4, \ldots, |T|, \tag{5.16}$$

$$z_{si} \in \{0,1\} \quad \text{for } s \in S, i = 3, \ldots, |T|. \tag{5.17}$$

Indeed, whenever $k \coloneqq \sum_{t \in T} y_{ts} \geq 3$, and constraint (5.15) is satisfied by $(z_{si} : s \in S, i = 3, \ldots, |T|)$, the other constraints (5.16) and (5.17) imply that, for any $s \in S$, we must have $z_{si} \geq 1$ for $3 \leq i \leq k$. Furthermore, (5.15) is satisfied with equality if and only if $z_{si} = 1$ for $3 \leq i \leq k$ and $z_{si} = 0$ for $i > k$, and so $\sum_{i=3}^{k} J_i$ is added to the left-hand side of the budget constraint (5.14). Note that the $|S|$ binary constraints (5.17) are necessary if $J_{i-1} \leq J_i$, for $4 \leq i \leq |T|$.

Now given the above constraints, we would also like to choose an allocation in which every station $s \in S$ has its hourly requirements H_{sm}, for each mission type $m \in M$ met by any boats able to perform that type of mission, *i.e.*

$$\sum_{t \in T : m \in M_t} h_{tsm} \geq H_{sm}.$$

Unfortunately, this is not always possible, especially for small enough budget D. Thus we choose to make the hourly requirements an elastic constraint, where we penalize

[3]The USCG Business Rules dictate that at least two boats are to be assigned at each station, and the capability requirements of the boats typically forces a station to have at least two distinct types; hence we do not penalize for allocating two or less types per station.

only those missions for which there is a shortage of hours. We introduce a nonnegative weight or penalty multiplier W_{sm} for each $m \in M$ and $s \in S$, and minimize

$$\sum_{s \in S} \sum_{m \in M} W_{sm} \cdot \max \left\{ 0, H_{sm} - \sum_{t \in T : m \in M_t} h_{tsm} \right\}.$$

The value of the objective can be seen as a convex combination of all shortage hours x_{sm}, over $m \in M$, $s \in S$, if we scale the multipliers W_{sm} so that $\sum_{s \in S} \sum_{m \in M} W_{sm} = 1$. Thus, when considering some $s \in S$, the values W_{sm} encode mission preference, *i.e.* $W_{s\,m_0} > W_{s\,m_1}$ implies that meeting mission hour requirements towards m_0 one more important than those towards m_1.

To make the objective function linear, we introduce non-negative variables x_{sm} for each $s \in S$ and $m \in M$, and impose the following constraints:

$$x_{sm} \geq H_{sm} - \sum_{t \in T : m \in M_t} h_{tsm} \quad \text{for } m \in M, s \in S, \tag{5.18}$$

$$x_{sm} \geq 0 \quad \text{for } m \in M, s \in S. \tag{5.19}$$

and aim to minimize

$$\sum_{s \in S} \sum_{m \in M} W_{sm} x_{sm}. \tag{5.20}$$

Let any feasible solution to the above set of constraints (5.1)–(5.19) be called a *feasible allocation*, and any feasible allocation which minimizes (5.20) is called an *optimal allocation*.

Since the hourly requirements are not rigid constraints, the system (5.1)–(5.19) is always feasible for large enough budget D, global and local inventories B_t and B'_t, $t \in T$, and pier spaces P_s, $s \in S$. Thus we can include auxiliary variables within the corresponding constraints which are meant to find where any such parameter might need an increase in order to achieve feasibility. In our case, to keep the number of new variables small, we only choose to introduce two sets of variables v_t and u_s, which denote an increase in B_t, for $t \in T$, or P_s, for $s \in S$, respectively. Now we replace the corresponding constraints (5.5) and (5.6) with:

$$\sum_{s \in S} b_{ts} \leq B_t + v_t \quad \text{for } t \in T, \tag{5.21}$$

$$v_t \in \mathbb{Z}_+ \qquad \text{for } t \in T. \tag{5.22}$$

$$\sum_{t \in T} b_{ts} \leq P_s + u_s \qquad \text{for } s \in S, \tag{5.23}$$

$$u_s \in \mathbb{Z}_+ \qquad \text{for } s \in S, \tag{5.24}$$

For the system defined by (5.1)–(5.4), (5.7)–(5.19), and (5.23)–(5.22), we call any feasible solution an *augmented allocation*. Now it is easy to see that a feasible allocation exists if and only if an augmented allocation exists with $\sum_{s \in S} u_s + \sum_{t \in T} v_t = 0$. Thus, if the objective is altered to:

$$\sum_{s \in S} \sum_{m \in M} W_{sm} x_{sm} + \alpha \sum_{t \in T} v_t + \beta \sum_{s \in S} u_s, \tag{5.25}$$

where α and β are penalty multipliers chosen large enough, *e.g.* $\sum_{s \in S} \sum_{m \in M} H_{sm} + 1$, so to meet any augmented allocation of value less than $\min\{\alpha, \beta\}$ must be a feasible allocation. Conversely, any lower bound (*e.g.* from a relaxed solution) on the objective value (5.25) of any augmented allocation which is at least $\min\{\alpha, \beta\}$, proves that no feasible allocation exists. This dichotomy allows the USCG analyst to conduct preliminary tests to determine whether or not the input parameters define a feasible allocation.

In our model, we give the option of allowing boats of a single type $t^* \in T$ to be *shared*, or assigned among subsets of stations rather than assigned only to individual stations.[4] More specifically, given κ disjoint subsets $S_k \subseteq S$, for $k = 1, \ldots, \kappa$, which partition S, we introduce a variable q_k denoting how many boats of type t^* are assigned to be utilized by any station $s \in S_k$, for $k = 1, \ldots, \kappa$, and enforce:

$$q_k = \sum_{s \in S_k} b_{t^* s} \qquad \text{for } k = 1, \ldots, \kappa, \tag{5.26}$$

$$q_k \in \mathbb{Z}_+ \qquad \text{for } k = 1, \ldots, \kappa. \tag{5.27}$$

We then relax the integrality of $b_{t^* s}$, eliminating the redundancy that constraint (5.27) would otherwise have, by replacing (5.2) with:

$$b_{ts} \in \mathbb{Z}_+ \qquad \text{for } t \in T \setminus \{t^*\}, s \in S, \tag{5.28}$$

$$b_{t^* s} \geq 0 \qquad \text{for } s \in S. \tag{5.29}$$

[4]Though extension to multiple types is easy to model, sharing boats is not typical in the USCG, hence we restrict the presentation to a single shared boat type.

The constraints (5.1), (5.3)–(5.19), and (5.26)–(5.29), provide a relaxation of the original system which intuitively provides feasible solutions, or *shared allocations*, in which the fractional value $b_{t^* s}$, for some k and $s \in S_k$, is now only an estimate on the number of boats of type t^* which are needed to meet the hours assigned to s, $\sum_{m \in M_{t^*}} h_{s\, t^* m}$. In other words, this shared allocation problem does not provide a clear "home" station for each of the boats of type t^* assigned among sets S_k, and thus we interpret the solution as a sort of lower bound on the quantitative benefit of sharing compared to not sharing resources.

When $b_{t^* s}$ is allowed to be fractional as above, the constraint (5.1) implies that $b_{t^* s}$ is either 0 or a real number greater than or equal to 1. We relax this constraint to a slightly weaker semi-continuous type of constraint:

$$\gamma \cdot y_{t^* s} \leq b_{t^* s} \qquad \text{for } s \in S, \tag{5.30}$$

$$y_{ts} \leq b_{ts} \qquad \text{for } t \in T \setminus \{t^*\}, s \in S \tag{5.31}$$

for some $\gamma \in (0, 1]$. Intuitively, this means that whenever a boat of type t^* is assigned to a subset of stations S_k, at least a fraction of γ of any such boat's hours A_t should be utilized. Thus, when implementation of the shared allocation is considered, there are no stations requiring a small fraction of hours ϵ with $0 < \epsilon < \gamma$.

The entire model appears together as (5.32)—(5.55).

$$\text{(BAM) minimize} \quad \sum_{s \in S} \sum_{m \in M} W_{sm} x_{sm} + \alpha \sum_{t \in T} v_t + \beta \sum_{s \in S} u_s \tag{5.32}$$

$$\text{subject to} \quad \gamma \cdot y_{t^* s} \leq b_{t^* s} \qquad s \in S \tag{5.33}$$

$$y_{ts} \leq b_{ts} \qquad t \in T \setminus \{t^*\}, s \in S \tag{5.34}$$

$$b_{ts} \leq B'_t y_{ts} \qquad t \in T, s \in S \tag{5.35}$$

$$\sum_{s \in S} b_{ts} \leq B_t + v_t \qquad t \in T \tag{5.36}$$

$$\sum_{t \in T} b_{ts} \leq P_s + u_s \qquad s \in S \tag{5.37}$$

$$\sum_{t \in T_c} b_{ts} \geq R_{sc} \qquad c \in C_s, s \in S \tag{5.38}$$

$$\sum_{t \in T_c} y_{ts} \leq R'_{sc} \qquad c \in C_s, s \in S \tag{5.39}$$

$$\sum_{m\in M_t} h_{tsm} \le (A_t + E_t) b_{ts} \qquad t \in T, s \in S \tag{5.40}$$

$$\sum_{s\in S}\sum_{m\in M_t} h_{tsm} \le (A_t + L_t E_t) \sum_{s\in S} b_{ts} \qquad t \in T \tag{5.41}$$

$$\sum_{m\in M_t} h_{tsm} \ge (A_t - E'_t) b_{ts} \qquad t \in T, s \in S \tag{5.42}$$

$$h_{t\,s\,m^*} \ge U_t y_{ts} \qquad t \in T, s \in S \tag{5.43}$$

$$\sum_{s\in S}\left(F_t\left(b_{ts}+V_t\sum_{m\in M_t} h_{tsm}\right)+\sum_{i=3}^{|T|} J_i \cdot z_{si}\right) \le D \tag{5.44}$$

$$\sum_{i=3}^{|T|} z_{si} \ge \sum_{t\in T} y_{ts} - 2 s \in S, \tag{5.45}$$

$$z_{s\,i-1} \ge z_{si} \qquad s \in S, i = 4, \ldots, |T|, \tag{5.46}$$

$$z_{s3} \in \{0,1\} \qquad s \in S, i = 3, \ldots, |T|. \tag{5.47}$$

$$x_{sm} \ge H_{sm} - \sum_{t\in T: m\in M_t} h_{tsm} \; m \in M, s \in S \tag{5.48}$$

$$x_{sm} \ge 0 \qquad m \in M, s \in S \tag{5.49}$$

$$q_k = \sum_{s\in S_k} b_{t^*\,s} \qquad k = 1, \ldots, \kappa \tag{5.50}$$

$$q_k \in \mathbb{Z}_+ \qquad k = 1, \ldots, \kappa \tag{5.51}$$

$$b_{ts} \in \mathbb{Z}_+ \qquad t \in T \backslash \{t^*\}, s \in S \tag{5.52}$$

$$u_s \in \mathbb{Z}_+ \qquad s \in S \tag{5.53}$$

$$y_{ts} \in \{0,1\} \qquad t \in T, s \in S \tag{5.54}$$

$$h_{tsm} \ge 0 \qquad m \in M_t, t \in T, s \in S \tag{5.55}$$

Notice that the constraints above with t^* singled out may still exist if no sharing is desired, since the trivial partition S_k, with $|S_k| = 1$, of S may be chosen. Furthermore, notice that $b_{t^*\,s} \ge 0$ is implied by (5.30), and also that constraint $v_t \in \mathbb{Z}_+$ is superfluous for every $t \in T$, even when $t = t^*$, since $\sum_{k=1}^{\kappa}\sum_{s\in S_k} b_{t^*\,s} = \sum_{s\in S} b_{t^*\,s}$; thus we omit them from the overall formulation.

Given any solution to the BAM model (5.32)–(5.55), unless $\sum_{s\in S} u_s + \sum_{t\in T} v_t > 0$, we post-process the aggregate variables b_{ts}, for each $t \in T \setminus \{t^*\}, s \in S$, into individual boat allocations by assigning to each boat $\sum_{m\in M_t} h_{tsm}/b_{ts}$ hours, which is possible since:

$$A_t - E'_t \le \frac{\sum_{m\in M_t} h_{tsm}}{b_{ts}} \le A_t + E_t.$$

Thus, in the sequel, we refer to a boat's hours rather than the "average" boat's hours.

If $\kappa < |S|$, then some subset $S_k \subseteq S$ must be non-trivial, *i.e.* $|S_k| > 1$, in which case the q_k boats allocated to S_k are given to the corresponding managing sectors or districts to develop an implemenation strategy; for this reason, we restrict attention to the partition of S into sectors or districts. Hence, when a shared solution is found, our model does not explicitly assign a "home" station to the boats shared among subsets $S_k \subseteq S$, for each k.

5.3 Implementation

At the onset of the software design of BAM, the USCG requested a tool which could incorporate Microsoft's Excel spreadsheets. It was also necessary to provide an intuitive interface to the optimization solver for USCG analysts. FICO's Xpress Optimization Suite (XOS) was a viable solution—we implemented in Xpress IVE v1.23.00, with Mosel v3.4.0, and Optimizer v23.01.03—as it is able to both read and write Excel files. Moreover, XOS has a Windows-based interface, IVE, equipped with an intuitive algebraic modeling language (which doubles as a programming language), and is powerful enough to solve large-scale mixed-integers optimization problems. Many of the parameters internal to XOS are tunable, some of which we include in our implementations. For example, to allow for flexibility when solving linear programming (LP) relaxations, we include a parameter specifying whether the Primal Simplex, Dual Simplex, or Newton Barrier Method should be used (defaulting to the Dual Simplex, as the number of constraints tends to be large). Once a program has been run, the IVE interface also allows decision-makers to explore solutions to various input data, and to quickly run other scenarios without having reload the output spreadsheet.

The interface is a text file which can be displayed in the IVE, and has all modifiable parameters listed in sequence, where each parameter is explained with in-line comments including suggested default values. An accompanying User Guide drafted by the development team includes a description of the boat allocation problem, the formats of corresponding auxiliary files, and a complete step-by-step tutorial. In-depth

descriptions of the key parameters are also included in the manual. The user guide was successfully tested on a member of CG-771 external to the development of the software.

One important auxiliary file is the "core" text file containing the encoding of the model (5.32)–(5.55). This file is typically not directly needed by an analyst when running experiments, though it is written in the XOS Mosel language and is thus easily modifiable. For example, new constraints can be added to the core file with a few extra lines of code. Other files include input and output Excel spreadsheets, both of which have been formatted by CG-771 members.

The software tool was designed with flexibility and expansion in mind. In particular, one can specify scenarios where only a subset of either the inventory of boat types or coastal stations are available, or the number of shared boat types can be expanded beyond a single type without having to revisit or modify the Excel input file. On the other hand, the tool enables the user to introduce, for example, new boat types or stations by a simple change in the input file.

The USCG analysts of CG-771 were partners throughout the optimization model development process, and were also major contributors towards testing the credibility of the tool. In accordance with the Validation and Verification (V&V) standards of the USCG, we worked together in designing and running tests which would attempt to expose any flaws of the model or its implementation. This process was a key part in achieving accredidation status of the BAM software package.

While the tool is still fairly new to those in managerial positions at the USCG, it is integrated into the CG-771 analysts' computer systems. It is primarily used to aid in decision-making for various hypothetical scenarios, many of which are presented in the following section.

5.4 Results

5.4.1 Data set

The data set used for our experiments was provided by the CG-771 team. Many parameters (*e.g.* preferred hourly use A_t for each boat type) came from USCG databases,

Boat name	Abbreviation	Number of boats FY09	Number of boats B_t	Max number of hours per boat A_t	Fixed cost ($) per boat f_t	Var cost ($) per hour v_t
Motor lifeboat	MLB	106	109	600	36,951	120
Response Boat						
—Medium	RB-M	167	162	600	36,000	120
—Small	RB-S	335	231	1,000	5,657	47
—Small Auxiliary	RBS-AUX	13	0	500	5,657	47
Special Purpose Craft						
—Near-shore lifeboat	SPC-NLB	2	3	350	15,000	60
—Heavy weather	SPC-HWX	4	4	350	15,000	120
—Law enforcement	SPC-LE	41	26	1,000	9,217	87
—Shallow water	SPC-SW	47	46	500	4,390	63
—Air	SPC-AIR	8	12	100	2,000	45
—Ice	SPC-ICE	24	46	50	1,000	15
—Skiff	SPC-SKF	56	0	100	500	15

Figure 5.1: USCG boats by type and associated BAM model parameters; FY09 data from 2009 [98]; remaining data acquired for tests in 2012.

whereas values such as E_t—the maximum amount a boat may exceed the preferred yearly use A_t—were estimated by observing past allocations. In the subsection that follows, we present tests performed to determine the effects of changes to some such parameters. A majority of the tests were performed on an x86-64 machine with an Intel Core i7 950 chip—clocked at 3.06GHz—with 9Gb of available RAM and the Windows 7 operating system, and were allowed to run up to 300 seconds each.

Figure 5.1 provides the values of many parameters used by the BAM model which are indexed by boat type. The third column displays the amounts of each boat type reportedly available in the USCG's fiscal year of 2009 (FY09), whereas the fourth lists those available to our model, B_t. Although about half the quantities between columns FY09 and B_t are relatively similar, it is worth noting that the RBS-AUX and SPC-SKF boat types were not used in our tests. The former type, RBS-AUX, is absent due to the ongoing effort by the USCG to update them with the newer model of the same basic boat, the RB-S. The SPC-SKF type was chosen to be left out of the BAM tests due to its relatively low hourly availability and maintenance costs. Furthermore, the SPC-SKF are trailerable, meaning they are much easier than other boat types to transport

between stations. Thus both types were deemed to be manageable outside the model.

Another quantity which differs from the FY09 values is the value of the RB-S preferable hours $A_{\text{RB-S}}$. As recorded in the similar work in [98], the USCG determined that each RB-S can be utilized up to 500 hours each year before calendar-based maintenance becomes mandatory. The CG-771 team along with confirmation from the OBF informed us, however, that engineers are allowing this number to be extended to as much as 1000 hour per RB-S. Note that fixing $E_{\text{RB-S}} = 500$ in not equivalent to changing $A_{\text{RB-S}}$, as the global bound (5.10) in this case can be too restrictive depending on the value of $L_{\text{RB-S}}$—consider $L_{\text{RB-S}} = 0$. Thus we let $A_{\text{RB-S}} = 1000$ and $E_{\text{RB-S}} = 0$; see also the fifth column of Figure 5.1. Due to the versatility of RB-S boats as well as the large quantity available compared to the remainder of the fleet, this increase in A_t applies only to the RB-S boat type.

For many boat types, we have $B'_t = B_t$, which denotes that there is no restrictive limit on the number of boats of type $t \in T$ at the station-level. On the other hand, many of the special-purpose craft have B'_t set to 1.

Each P_s was set to the constant value of 14, which was determined to be the minimum such constant allowing for a feasible allocation, *i.e.* the minimum $\delta \in \mathbb{Z}_+$, such that $P_s := \delta$ for all $s \in S$, yielded a solution with $\sum_{s\in S} u_s + \sum_{t\in T} v_t = 0$.

Preliminary testing of BAM showed that choosing the parameter $\gamma \in [0.1, 0.9]$ did not cause a significant change in objective value, while values below 0.1 were deemed unreasonable by CG-771, so we chose the least restrictive value $\gamma := 0.1$. Of course, any choice of γ is superfluous if sharing is not desired, *i.e.* if the partition S_k, $k = 1, \ldots, \kappa$, is such that $|S_k| = 1$ for each k.

The maximum hourly increase of a station's boat, E_t, was set to be 10% of the preferred amount A_t, with the exception of RB-S where we recall that $E_{\text{RB-S}} := 0$. Similarly, the value of the maximum hourly deviation E'_t below A_t was set to 70%, since many boat types (*e.g.* SPC-ICE) had not been operational more than an average of 30% of available hours in the 2012 data provided. Further, the fraction of hours L_t an average boat of type $t \in T$ can be utilized (nationwide) was set to zero. This implies that although an individual boat at any station is able to perform up to 110% of the

preferred hours A_t, the type, on average, must not exceed this preference. This is due to the fact that boats are scheduled for annual maintenance, and adhering on average to the preference hours ensures that such a schedule is followed.

At any station $s \in S$, and for any capability $c \in C_s$, the minimum number of boats of capability types $t \in T_c$ required at s, *i.e.* the value of R_{sc}, ranges from 1 to 4 where, for example, $R_{s,\text{Tactical}} = 4$ means that s requires at least 4 boats with Tactical capabilities. On the other hand, the maximum number of types allowed $t \in T_c$ to satisfy $c \in C_s$ at s, *i.e.* the value R'_{sc}, was either 1 or $|T_c|$, where the latter choice encodes the fact that station s does not impose such a restriction—in the implementation, this constraint can be removed. If $R'_{sc} = 1$, then only one type of boat $t \in T_c$ is allowed at station s, of which there must be at least R_{sc} in quantity.

The value of each J_i is defined as the incremental cost of assigning one more boat type, increasing from $i-1$ types to i, for each $i = 3, \ldots, |T|$. Recall that the USCG acknowledges that at least two types are needed at each station to properly carry out mission requirements, and that increasing the number of distinct types also involves more operational training for stationed personnel. Furthermore, it is highly undesirable for a station to have too many types, which is defined as 5 or more. For this reason, we have chosen to $J_3 = J_4 = \$1,500$, and $J_i = \$10,000$ for $i \geq 5$. In our tests, 5 different types were rarely needed, and 6 or more were never assigned to a station. On the other hand, every test resulted in at least one station being assigned boats of 3 or more distinct types.

Our model allows at most one type of boat, *i.e.* $t^* =$ "RB-S", to be shared among stations at the sector or district level. In other words, we only allow the partition S_k, $k = 1, \ldots, \kappa$, to be defined with all subsets S_k being either an individual stations (no sharing), a sector, or a district; in this order, the problems become successively less restrictive since each district is made up of sectors.

Recall that the set of mission types M contains the "pseudo" mission $m^* =$ "Training". We also include a second "pseudo" mission known as "Command Discretion", denoted m^{**}. Here, any hours $H_{s\,m^{**}}$ required by a station $s \in S$ are actually hours

reserved for those in command to use towards any unexpected variability in the remaining mission hours H_{sm}, $m \in M \setminus \{m^{**}\}$. For our main tests, unless otherwise noted, we fixed $H_{s\,m^{**}} = 0$.

The budget value D, the hourly mission requirements at a station H_{sm} and corresponding precedence factor or penalty W_{sm}, for $m \in M, s \in S$, and hourly training minimum of a type U_t, for $t \in T$, were provided by the CG-771, and have been requested to remain undisclosed.

Recall that α and β must be large enough to detect whether a feasible solution to BAM exists. We choose $\alpha = \beta = \sum_{s \in S} \sum_{m \in M} H_{sm} + 1$, since $0 \leq W_{sm} \leq 1$ for each $m \in M$, $s \in S$.

Preliminary tests showed that with the given data set for BAM, there was no feasible allocation, as at least 8 SPC-HWX vessels were needed, *i.e.* an optimum solution to (5.32)–(5.55) yielded $u_t = 4$ for $t =$ "SPC-HWX". After further investigation, we found 4 stations with two capabilities having values of corresponding R_{sc} and R'_{sc} *cf.* (5.7)–(5.8) which were restrictive enough to require at least 2 SPC-HWX vessels each. A small alteration of replacing R'_{sc} with $R'_{sc} + 1$ fixed the problem, *i.e.* allowed feasible allocations, but it should be pointed out that this step is a relaxation of the corresponding Business Rules. We feel that this is the least "invasive" possible change, at least for the purpose of testing.

5.4.2 Tests

We first tested the BAM model with the data discussed in the previous section (§5.4.1) by finding the smallest budget D in which the value of the objective function is 0—for each of the three sharing choices—*i.e.* the first D for which a feasible allocation exists with no shortage hours; denote the value of D as the corresponding problem's *optimum budget*. We find each optimum budget by a simple binary search starting with an initial D large enough to allow some feasible allocation, *e.g.* \$100 million, and stopped the search within a tolerance of \$10,000. Here, each individual MIP was solved with a 30 minute limit, and stopped earlier than 30 minutes if either the objective value was bounded away from 0, or had a worst upper bound less than 100. This amounted to

Sharing plan	Opt. budget (%)	Low bound	Abs. gap	CPU (s)	# Nodes
station	92.88	0.00	73.01	300	16,867
sector	92.37	0.00	98.52	300	22,189
district	91.86	0.00	10.76	300	28,224

Figure 5.2: Xpress B&B statistics for solving each sharing plan with optimum budgets

stopping after solving 10 BAM instances in less than 5 hours. Note that all budget values are confidential, and so we report those found by our model in percentages of a value D^*, which represents the actual USCG budget corresponding to the data set we were provided—denote this data set of FY10.

Each test run presented hereafter was allowed only a 300 second limit, including the re-running of the data sets with optimum budget values—Figure 5.2 displays these results. This small amount of time allows the solver enough time to find fairly good solutions, and is approximately when *tailing off* is typically observed, *i.e.* when the relative improvement between the worst upper bound and best known solution value slows to within 0.1%, per 100 seconds. In the case the upper bound remains at 0, tailing off is said to occur when the relative improvement of the feasible solutions drops below %0.1 per 100 seconds when the value of the solution is 100 or above, and otherwise when no absolute improvement has occurred in 100 seconds. The size of the matrix for BAM in Xpress' standard form for solving the corresponding MIP, and how the XOS preprocessing reduces the problem size further, is found in Figure 5.3.

To give a sense of the savings our model provides, consider the hypothetical case $D^* = \$100,000,000$. Here, even without sharing, we find that the optimum budget is $\$92,880,000$, for a savings of over 7 million dollars. When sharing is allowed at

	Matrix	Preprocessing
rows (constraints)	45,686	9,672
columns (variables)	33,995	17,291
non-zero elements	214,831	83,427

Figure 5.3: Matrix statistics for BAM input to Xpress IVE, sharing disallowed

	BAT [98]			BAM		
Boat type	**Original USCG allocation FY09**	**BAT allocation**	**Implemented allocation**	**No sharing**	**Sector-wide sharing**	**District-wide sharing**
MLB	106	102	102	58	63	64
RB-M	166	166	158	162	162	158
RB-S	360	208	318	231	231	231
RBS-AUX	10	10	10	0	0	0
SPC-NLB	3	2	3	3	3	3
SPC-HWX	4	0	4	4	4	4
SPC-LE	33	26	20	26	25	26
SPC-SW	47	47	47	46	46	46
SPC-AIR	8	8	12	12	10	12
SPC-ICE	0	24	0	46	46	46
SPC-SKF	67	29	42	0	0	0
total	804	622	716	588	590	590

Figure 5.4: Difference between USCG, BAT [98], and BAM allocations

the sector level, over $\$500,000$ more could be saved. When sharing is brought up to the district level, we see that another approximate savings of $\$500,000$, for a total (hypothetical) savings of $\$8,140,000$. It should noted, however, that with the cost savings comes an allocation with potentially vast differences. In a future work, it would be interesting to include a modeling property known as persistence, where consecutive runs of our model with small data parameter changes result in similar allocations—or as similar as possible (see for example [16]).

Figure 5.4 records the boat allocations provided by both our tool (BAM)—with D fixed to the corresponding optimal budgets—and the Boat Allocation Tool (BAT) along with the comparable original USCG fiscal year 2009 allocation, which are found in [98]. The two models solve the USCG boat allocation problem, but are not identical in either the constraint set (*e.g.* only BAM includes capability constraints (5.7)–(5.8)) or the data set (*e.g.* BAT utilized FY09 data, whereas ours was modified from FY10). Unfortunately, the latter data set was not completely disclosed, and thus we cannot directly compare cost savings between them. However, with the help of a number of

unifying measures of success described in [98], we make other comparisons of the impact of each model. The measures are listed next, with only P8 redefined to take into account our costs taking percentage form.

- P1: Total number of boats assigned, or $\sum_{s\in S}\sum_{t\in T} b_{ts}$.
- P2: Percentage of stations with excess hours, or

$$|\{s \in S|\sum_{m,t} h_{tsm} - \sum_{m} H_{sm} > 0\}|/|S|.$$

- P3: Percentage of station with shortage hours, or

$$|\{s \in S|\sum_{m} H_{sm} - \sum_{m,t} h_{tsm} > 0\}|/|S|.$$

- P4: Average excess hours per station with an excess, or

$$\sum_{s\in S} \max\{0, \sum_{m,t} h_{tsm} - \sum_{m} H_{sm}\}/|\{s \in S|\sum_{m,t} h_{tsm} - \sum_{m} H_{sm} > 0\}|.$$

- P5: Average shortage hours per station with a shortage, or

$$\sum_{s\in S} \max\{0, \sum_{m} H_{sm} - \sum_{m,t} h_{tsm}\}/|\{s \in S|\sum_{m} H_{sm} - \sum_{m,t} h_{tsm}\}|.$$

- P6: Percentage of stations with more than 2 boat types, $|\{s \in S|\sum_t y_{ts} - 2 > 0\}|$.
- P7: Average number of boat types per station, or $\sum_{s\in S}\sum_{t\in T} y_{ts}/|S|$.
- P8: Total cost as percentage of USCG budget estimate D^*, D/D^*; BAM only.
- P9: Capacity utilization, or
  - USCG allocation: $1 - \sum_{s\in S} \max\{0, \sum_t A_t b_{ts} - \sum_m H_{sm}\}/\sum_{t\in T} A_t B_t$;
  - BAT/BAM allocations: $1 - \sum_{s\in S} \max\{0, \sum_{m,t} h_{tsm} - \sum_m H_{sm}\}/\sum_{t\in T} A_t B_t$.
- P10: Demand shortfall, or
  - USCG allocation: $\sum_{s\in S} \max\{0, \sum_m H_{sm} - \sum_t A_t b_{ts}\}/\sum_{s\in S}\sum_{m\in M} H_{sm}$;
  - BAT/BAM allocation: $\sum_{s\in S} \max\{0, \sum_m x_{sm}\}/\sum_{s\in S}\sum_{m\in M} H_{sm}$.

Performance metric	BAT [98]			BAM		
	Original alloc.	BAT alloc.	Impl. alloc.	No sharing	Sector sharing	District sharing
P1: Total size of utilized fleet	804	622	716	588	590	590
P2: Percentage of stations with excess hours (%)	61.2	1.7	41.6	28.2	35.6	33.3
P3: Percentage of stations with a shortage of hours (%)	38.8	0.0	1.1	1.1	3.4	1.1
P4: Average excess hours per station with an excess	556.3	102.0	209.8	90.7	96.2	100.2
P5: Average shortage hours per station with a shortage	563.1	0.0	70.0	2.5	109.4	6.0
P6: Percentage of stations with more than two boat types (%)	37.6	30.9	30.9	35.1	34.5	34.5
P7: Average number of boat types per station	3.1	2.3	2.2	2.3	2.4	2.3
P8: Fleet operating cost relative to est. USCG FY10 (%)	*	*	*	92.88	92.37	91.86
P9: Capacity utilization (%)	85.3	99.0	96.2	99.0	98.7	98.7
P10: Demand shortfall rate (%)	9.90	0.00	0.04	0.00	0.25	0.00

Figure 5.5: Difference between USCG original, BAT [98], and BAM measures of success

Looking at the results of applying the above measures to our solutions with optimum budgets (*cf.* Figure 5.5), we see that the total number of vessels allocated slightly increases when sharing is allowed in BAM. This can be partially explained by the accompanying Figure 5.6, which shows that a majority of boat types saw an increase in excess hours, but there was a decrease in excess RB-S hours. In other words, sharing

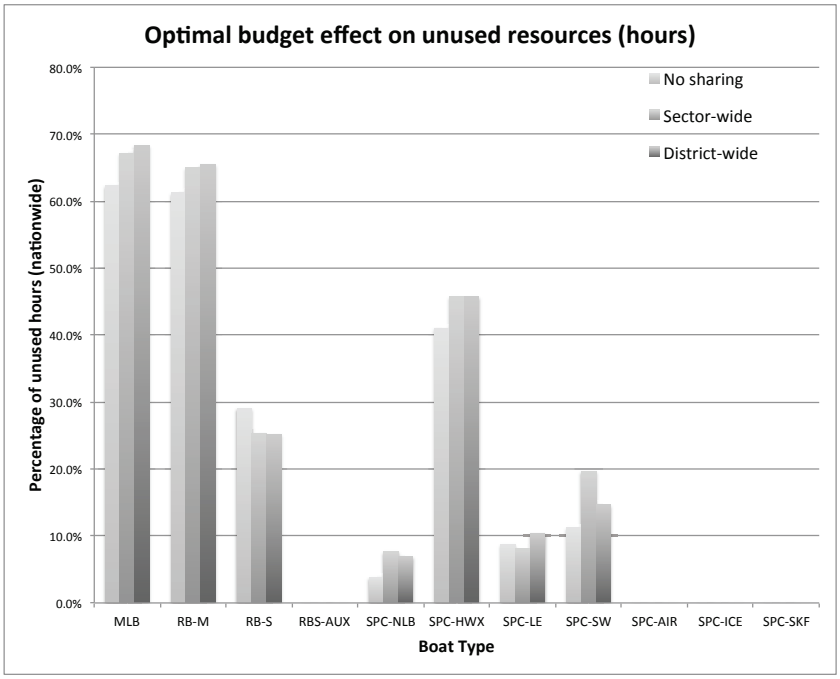

Figure 5.6: Optimal budget effect on unused resources (hours)

boats of type RB-S allows for their more efficient hourly use, which in turn eliminates some hours spent aboard vessels with higher variable costs (*e.g.* MLB and RB-M; *cf.* Figure 5.1). This effort results in a savings, since roughly the same number of total hours are spent aboard all vessels, and implies that vessels with small minimum hourly requirements (*i.e.* small $A_t - E'_t$ in (5.12)) are desirable.

Measures P2–P5 demonstrate that our BAM model does not reduce total excess hours, which is to be expected, as the objective function (5.25) of our model does not penalize excess hours assigned. However, excess assignments are not considered wasteful, so long as vessels adhere to scheduled maintenance, since we are told that these hours can be transfered to the "Training" or "Command Discretion" missions. In fact, in our tests, we have observed that any surplus occuring at a station, *i.e.* $H_{sm} < \sum_{t \in T} h_{tsm}$, is primarily found for $m = m^*$, or "Training". Since our tests were run with $W_{s\,m^*} > \min_{m \in M}\{W_{sm}\}$, we find that the cause is the minimum training amount U_t required per boat type allocated to the station.

When comparing our performance with BAT, the average excess (P4) is lower in BAM, but the number of stations having excess hours in BAM is much higher. Furthermore, our optimal budget solutions, whether sharing or not, are improvements—with respect to P2–P5—upon the allocations made by the USCG with and without the help of BAT.

The next two measures, P6 and P7, show that BAM's extra charges J_i for incrementally adding new boat types to a station keep the average number of types per station relatively low, at approximately 2.3. Furthermore, these values dominate those of the initial USCG allocation. However, the BAT model and data set yields better P6 values than BAM, *i.e.* BAT achieves a smaller number of allocated types across the stations, and almost identical to the BAM P7 values.

Our measure P8 is incompatible with the BAT, since it represents the ratio of a problem's budget D to the USCG FY10 estimate D^* (see Figure 5.2). Nevertheless, without sharing we can save over 7.1%, or several million dollars, whereas with sharing, it may be possible to save up to 8.1%, or an extra several hundred thousand dollars.

Finally, P9 and P10 respectively provide measures of how successfully an allocation

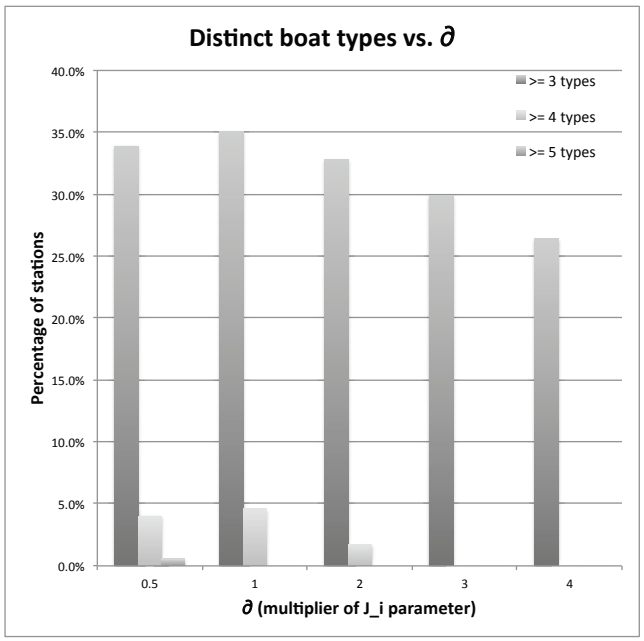

Figure 5.7: Dinstinct boat types *vs.* δ

utilizes the available resources and, respectively, the overall shortfall in hourly demands. Our statistics are comparable to those of BAT, which can be seen as large improvements over the USCG initial allocation.

Our next set of tests include varying particular input parameters and measuring their effect on measures P1–P10. In particular, Figure 5.7 displays the percentage of stations with strictly more than 2, 3, or 4 distinct boat types, when the J_i values are scaled by a constant factor δ. It is clear that virtually no station ever receives 5 or more distinct types, and for large enough δ, no station will be given a fourth distinct type. On the other hand, as δ grows, the percentage of stations with more than two boat types tends to decrease as expected, but may never fall below 2 due to constraining factors such as station capability requirements and a fixed inventory.

Figure 5.8 displays a graph of two related plots. The first plot (squares) shows an increasing trend in average shortage hours per station with observed shortages (P5) with respect to γ—the minimum percentage of hours a shared RB-S vessel is operated at a station. On the other hand, the second plot (diamonds) tells us that as γ increase, the number of stations with a single vessel which is being shared decreases; these two

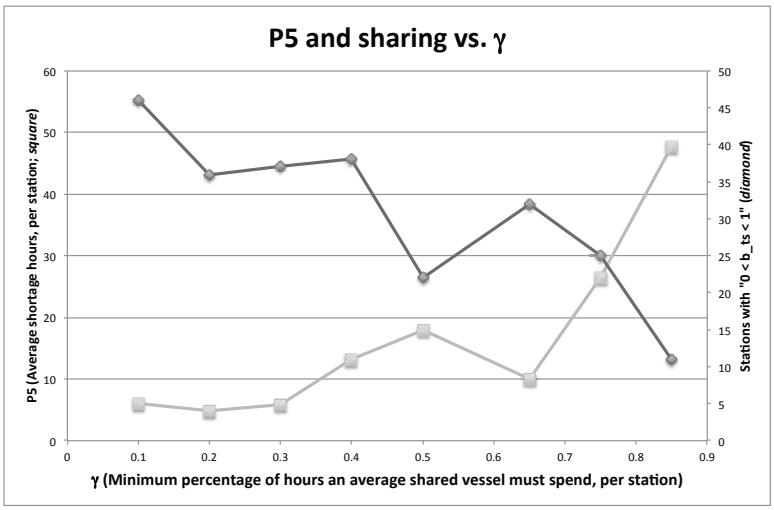

Figure 5.8: P5 and sharing *vs.* γ

measures also appear to have an inverse relationship. Indeed, since the total number of (weighted) unmet hours cannot decrease as γ approaches value 1, *i.e.* as it becomes more restrictive, these observations tell us that sharing is most utilized when γ is small. However, a caveat arises here in the actual implementation of shared allocations. In particular, for small values of γ, a large number of stations can do without harboring an RB-S full-time, but still require a (possibly small) amount of the vessel's utility.

Figure 5.9 depicts two trends as the variability in a boat's scheduled hours is allowed

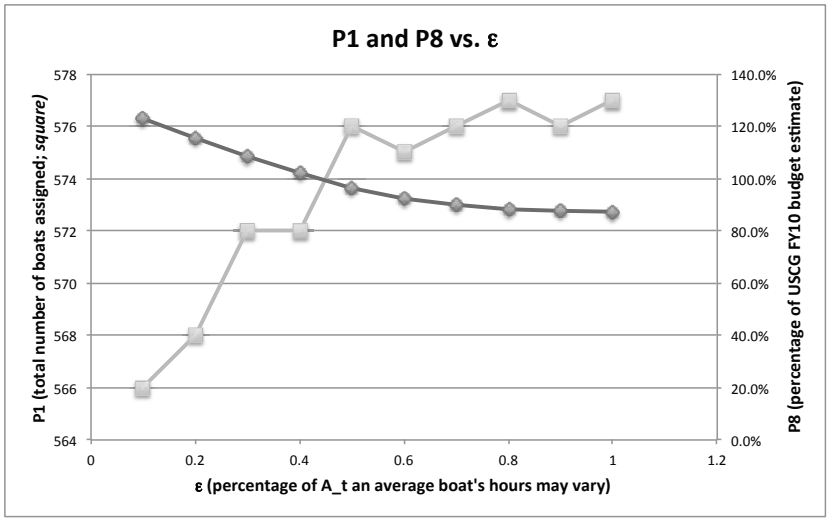

Figure 5.9: P1 and P8 *vs.* ϵ

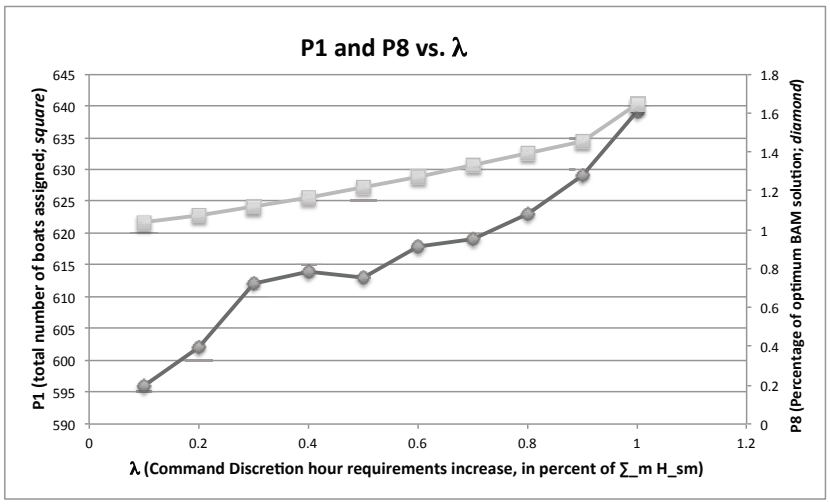

Figure 5.10: P1 and P8 *vs.* λ

to increase. The horizontal axis displays ϵ, where we set $E_t = E'_t = \epsilon$, for all $t \in T$, and so

$$A_t - \epsilon \leq \frac{\sum_{m \in M_t} h_{tsm}}{b_{ts}} \leq A_t + \epsilon.$$

The first plot (squares) records the value of P1, or the total number of boats assigned for each given ϵ. Here we notice that as the allowed range of assignable hours expands, more vessels are allocated. One explanation for this behavior is that the capability constraints force a number of boats to be allocated at the stations. So if at some station the capable boats are expensive to maintain, *i.e.* V_t is large, and ϵ is small enough, it becomes worthwhile to utilize such boats minimally and to utilize a new boat which has a cheaper maintenance rate. This is further confirmed by the second plot (diamonds) of Figure 5.9, which shows that the relative optimal budget decreases—at a decreasing rate—as ϵ increases.

Figure 5.10 displays two plots with a horizontal axis consisting of a parameter λ, defined to be the percentage of the total mission hours $\sum_{m \in M \setminus \{\text{Command Discretion}\}} H_{sm}$ that is set aside for the mission m^{**} = "Command Discretion", *i.e.* to account for variability. In other words, larger values of λ encode a requirement that stations allow for larger surpluses in mission hour demand. The first plot (squares) shows that as the number of hours $H_{s\,m^{**}}$ grows, the total number of boats (*i.e.* P1) increases at a roughly linear rate. The second plot (diamonds) shows that the optimum budget (*i.e.* P8) is

Sharing plan	# RB-S	Opt. budget (%)	Low bound	Abs. gap	CPU (s)	# Nodes
station	310	88.47	0.00	34.00	300	18,938
sector	295	88.14	0.00	44.13	300	25,526
district	285	87.90	0.00	30.79	300	37,091

Figure 5.11: Xpress branch-and-bound statistics for solving each sharing plan with respective optimum budgets and maximum needed RB-S

also an increasing function with respect to λ. However, the seemingly sporadic jumps in the latter plot are possibly due to the increase in boats of varying fixed costs, as well as the penalty for introducing new types.

Figures 5.11 and 5.12 provide statistics of what happens to solutions when the number of RB-S is allowed to be essentially unlimited, *i.e.* $B_{\text{RB-S}} \to \infty$. The first table records the resulting optimal budgets relative to the USCG FY10 estimate. Here we see that when compared to the optimal budgets with a constrained number of RB-S (*cf.* 5.11), the extra savings is around 4%, or a few million dollars more than already saved, regardless of sharing region. The tradeoff here is an increase in the number of RB-S used over all stations (*cf.* Figure 5.12). Here we see that as many as 80 new RB-S vessels are needed in order to save on maintenance-based costs. This is due to the versatility in terms of missions it can perform (*i.e.* $M_{\text{RB-S}} \subseteq M$) and both its

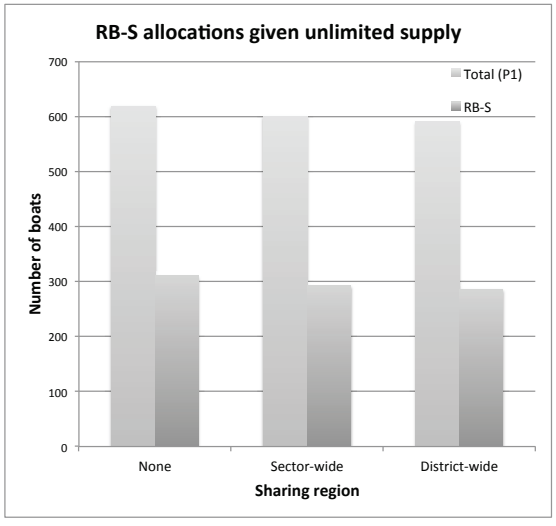

Figure 5.12: RB-S allocations given unlimited supply

maintanence-based costs being relatively inexpensive. However, this solution does not take into account the cost of purchasing the extra vessels, which is worthwhile only if it amounts to less than the savings gained in maintenance-based costs over the entire RB-S lifecycle.

5.5 Concluding remarks

In this thesis, we discussed one approach to solving a resource allocation problem of the USCG. Our use of elastic constraints allowed for an objective of meeting all mission hour requirements, but if shortage is necessary, then any unmet hours were penalized. The model was reconstructed many times and tested thoroughly after each change, with the USCG working beside us closely. This close-knit collaboration was a critical factor to the successful outcome of a working decision support system, which is currently installed on the personal computers of USCG analysts.

The boat allocation problem along with provided input data from the USCG showed that certain resources, *e.g.* SPC-NLB, were in short supply if the Business Rules were to be followed precisely. This prompted the investigation of potentially sharing available resources. While the USCG Business Rules do not currently allow such strategies, the model nevertheless allows the analyst to explore a multitude of such scenarios. On the other hand, we do not provide details on a station-level implementation of a shared allocation provided by BAM, as the model only provides assignments of the shared boats at the sector or district level—this investigation is part of an ongoing second phase of BAM. We provide preliminary evidence that an implementation of "seasonal" sharing—*i.e.* allocating a shared solution by assigning homes to each shared boat from quarter to quareter of an annual allocation—is computationally difficult (see forthcoming Appendix §B).

Our tool provides reasonable quality solutions in short amounts of time, allowing for quick re-optimization of hypothetical scenarios. It enables cost savings, as well as utilization of fewer boats than those available in inventory. Practically speaking, this implies that more savings can be made by selling unused resources, which in turn

means that less replacement costs will be incurred in the future. Such a tool is crucial for planning purposes, whether unforeseen budget cuts arise, new boat types become available, or the Business Rules evolve.

Appendix A

Constructing input parameters to ConSP

Here we present a possible approach to constructing the input parameters (n, k, m, α, β, S, W) of **conference scheduling problem** (*cf.* §2.1). The number of sessions n, time slots k, and rooms m, we assume are given, and α and β are typically 1 and n, respectively.

The set S consists of conflicting pairs, which we determine as follows. Suppose each session $i \in [n]$ has a set of *participants* A_i associated with it, or those persons essential to at least one presentation within the session. We can create S to be all pairs $\{i, j\} \in \binom{[n]}{2}$ such that $A_i \cap A_j \neq \emptyset$, *i.e.* a conflicting pair is one which shares a common participant. In this case, the set S resembles the edge set of an *intersection graph* $G_{\mathcal{H}} = (V, E)$ of set system $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, *i.e.* $\mathcal{E} \subseteq 2^{\mathcal{V}}$, and $V \coloneqq \mathcal{E}$ and $E \coloneqq \{(\mathcal{S}, \mathcal{T}) \in \binom{\mathcal{E}}{2} | \mathcal{S} \cap \mathcal{T} \neq \emptyset\}$.[1] With this choice in forming S, the organizers of a conference have an "easier" task of determining n sets A_i, rather than choosing a subset $S \subseteq \binom{[n]}{2}$ directly. It also gives flexibility in that A_i can contain anyone from a designated speaker or chairperson of the corresponding session, to a co-author or graduate advisor or advisee involved in some constituent talk.

The main focus of this section is now on how to form the non-negative, symmetric matrix $W \in \mathbb{R}^{n \times n}$, which is to encode a measure on relative pairwise similarity. Intuitively, we wish to identify the topics associated with each session or presentation therein, and quantify how much overlap lies between them. We assume that every conference is given some text document associated with each individual presentation, *e.g.* a title, abstract, or a set of keywords. Let D be the set of all such documents,

[1] It is easy to prove that the family of intersection graph coincides with all possible graph: consider any G and consider the set system over $\mathcal{V} = 2^{E(G)}$ with $\mathcal{E} = \{\delta_G(v) | v \in V(G)\}$, and note that $G = G_{(\mathcal{V}, \mathcal{E})}$, since for any $i, j \in V(G)$, $\{i, j\} \in G$ if and only if $\{e\} = \delta_G(i) \cap \delta_G(j)$.

i.e. the *corpus*. The text documents of D can be preprocessed so that unimportant word, *e.g.* articles or prepositions, are removed, and that others are strengthened by techniques such as collocation. After doing so, let T be the set of words appearing in at least one document, or *terms*.

For a given set of text documents D and terms T, a matrix M in the classical *vector space model* of Salton, Wong, and Yang [88] can be constructed, *i.e.* $M \in \mathbb{R}^{T \times D}$, where for any $t \in T$, $d \in D$, M_{td} represents some function of term t with respect to document d and corpus D. For example, a standard non-negative function for computing each entry is the decomposable *term-frequency inverse-document-frequency* function, denoted $f(t, d, D) \coloneqq g(t, d) \cdot h(t, D)$, where $g(t, d)$ is some measure on the frequency of term t within document d, and $h(t, D)$ is a measure of the term's relevance across corpus D. Intuitively, f produces large values whenever a term is both significant to a document and rarely found across the corpus as a whole, whereas otherwise the value is small, *i.e.* close to zero. The following choices of g and h are suitable for a given corpus D:

$$\begin{aligned} g(t,d) &\coloneqq \log(r(t,d)+1) && \text{for } t \in T, d \in D, \\ h(t,D) &= \log\left(\tfrac{|D|}{|\{d' \in D : r(t,d') > 0\}|}\right) && \text{for } t \in T, \end{aligned} \tag{A.1}$$

where $r(t, d)$ is the *raw frequency*, or the number of times term t appears in document d.

If we further scale the columns of matrix $M \coloneqq (v_d : d \in D)$ so that $\|v_d\| = 1$ for each $d \in D$, then the inner product of any such pair $v_d^\top v_{d'}$ yields a fractional value encoding the relative similarity between documents d and d'. In other words, $\widehat{W} \in \mathbb{R}^{D \times D}$ defined as $\widehat{W}_{d,d'} \coloneqq v_d^\top v_{d'}$ is now a similarity matrix encoding topic overlap between pairs of presentations. However, we ultimately wish to construct the similarity matrix $W \in \mathbb{R}^{n \times n}$ representing pairwise session similarity.

One obvious way of determining the similarity between sessions is to simply concatenate the text documents withing each session, so that D is a corpus in one-to-one corresponding with sessions, rather than individual talks. In such a case, W can be chosen as the above matrix $\widehat{W}$.

Another natural choice for similarity matrix W is as follows. Let $D_s \subseteq D$ denote the set of documents affiliated with constituent presentations of session $s \in [n]$, and let

W^{avg} denote the *average similarity matrix*, measuring the average similarity across all sessions between two sessions, *i.e.*

$$W^{\text{avg}}_{s,s'} := \frac{1}{|D_s| \cdot |D_{s'}|} \sum_{p \in D_s} \sum_{p' \in D_{s'}} \widehat{W}_{p,p'} \quad \text{for } s, s' \in [n]. \tag{A.2}$$

A perhaps more realistic similarity measure arises if we consider only those pairs of presentations $E_{s,s'} \subseteq D_s \times D_{s'}$ across sessions $s, s' \in [n]$ which may overlap in time if s and s' appeared in parallel (*e.g.* the INFORMS Annual Meeting [35] typically has 3–4 talks per technical session, each given identically intervals in which to present). Thus we define the *time-adjacent similarity matrix* W^{adj} as:

$$W^{\text{adj}}_{s,s'} := \frac{1}{|E_{s,s'}|} \sum_{\{p,p'\} \in E_{s,s'}} W_{p,p'} \quad \text{for } s, s' \in [n]. \tag{A.3}$$

We have thus demonstrated a few natural choices when forming the input for the **conference scheduling problem**. We end by mentioning that there are a number of software packages used for the computation of term-document matrices—among other text mining and natural language processing tasks—*e.g.* GenSim for Python [97], and TMG for Matlab [29].

Appendix B

Computational complexity of BAM

Here we give a proof of the $\mathcal{NP}$-hardness of BAM, as well as proof that a "seasonal" implementation strategy of a shared boat type is $\mathcal{NP}$-hard.

The **partition problem** is an $\mathcal{NP}$-complete decision problem [40] which takes as input a multi-set $\mathcal{A} \coloneqq \{a_i : i \in [n]\}$ of n positive integers, and asks whether or not a partition of $\mathcal{A}$ exists so that the sum over each set is identical. In other words, we must find two disjoint subsets N_0 and N_1 of $[n]$ where $N_0 \cup N_1 = [n]$, and so that

$$\sum_{i \in N} a_i = \sum_{i \in [n] \setminus N} a_i,$$

or equivalently

$$\sum_{i \in N} a_i = \tfrac{1}{2} \sum_{i \in [n]} a_i.$$

We reduce an arbitrary instance of **partition problem** to our **boat allocation problem**, the decision problem which asks whether or not an allocation of boats to stations exists in which all hourly demands across the stations is met for a fixed budget—BAM is the optimization version (see §5.2 for details and parameter descriptions). We will show that if exactly two boat types are available—each with identical capabilities, hourly capacities, and inventory sizes—and if any station can be allocated only one such boat type, then whenever the total number of hourly station demands is a particular factor of $\sum_{i \in [n]} a_i$, we can meet all mission hour requirements if and only if a partition of $\mathcal{A}$ exists. Since the former problem is known to be $\mathcal{NP}$-complete, this reduction implies that the latter problem is also an $\mathcal{NP}$-complete decision problem.

Suppose $(n, \mathcal{A})$ is a **partition problem** instance with $\mathcal{A} \coloneqq \{a_i : i \in [n]\}$. Now we define an instance of the **boat allocation problem** as follows. Note first that the weights W_{sm} found in the description of BAM do not apply, and the penalties α and β

in the model are not part of the input here.

- Let $S \coloneqq [n]$.
- Let T and C be arbitrary set of size two, *i.e.* $|T| = |C| = 2$.
- Let M be a singleton, *i.e.* $|M| = 1$, and $M_t = M$, for $t \in T$—we drop the 'm' subscript hereafter.
- Define $A_t \coloneqq 1$, for each $t \in T$.
- Define $B_t \coloneqq B'_t \coloneqq \frac{1}{2}\sum_{i\in[n]} a_i$, for each $t \in T$.
- Let $R_{sc} \coloneqq R'_{sc} \coloneqq 1$, for each $c \in C$, $s \in S$.
- Define $H_s \coloneqq a_s$ for each $s \in S = [n]$.
- Define all remaining parameters to be 0, *i.e.* $D = 0$, and $E_t \coloneqq E'_t \coloneqq L_t \coloneqq U_t \coloneqq 0$, for each $t \in T$.

From the above parameter definitions, it is easy to see that since $\sum_{t\in T} B_t = \sum_{i\in[n]} a_i$, and $A_t = 1$, $E_t = E'_t$ for each $t \in T$, then all mission hour requirements can only be met if all boats are allocated among S and all hours available to each vessel are utilized. This immediately implies that any partition of $\mathcal{A}$ into two sets N_0 and N_1 with identical sums can be turned into a feasible allocation of the above instance: designate $t_0 \in T$ to be the type assigned among stations $s \in N_0$ and let $h_{t_0\,s} = b_{t_0\,s} = a_s$, and similarly for $t_1 \in T$, let $h_{t_1\,s} = b_{t_1\,s} = a_s$. Further, since $H_s = a_s$, we have $H_s = \sum_{t\in T} h_{ts}$ for each $s \in S$, *i.e.* all mission hour requirements are met.

The other direction is trivial, since if no partition exists for $\mathcal{A}$, then for every subset $N \subseteq [n]$, we must have, without loss of generality,

$$\sum_{i\in N} a_i > \tfrac{1}{2}\sum_{i\in[n]} a_i.$$

This implies that whichever set of stations type $t_0 \in T$ is assigned to, at least one station will have a shortage of mission hours, *i.e.* $\sum_{s\in N} b_{t_0\,s} \leq B_t < \sum_{i\in N} a_i$. Thus we have proved the claim, which we record in the following theorem.

Theorem B.0.1. *The* **boat allocation problem** *is $\mathcal{NP}$-complete, even if $|T|= |C|= 2$ and $|M|= 1$.* □

Here we define an optimization problem associated with BAM which we call the **boat sharing problem**, or BSP. Intuitively, the problem statement is: given a shared allocation among a partition S_k, $k = 1, \ldots, \kappa$, of S, and a number λ of seasons which partition the hours requirements—$\lambda \geq 3$—find the minimum number of boats of the shared type (*e.g.* RB-S) to be assigned "home" stations, where boats can be reassigned to stations at the end of each season, and all hours constraints must be exactly satisfied. Now we precisely formulate the problem by first defining the following input parameters.

- S: set of stations
- B: number of boats available
- A: preferred number of hours a boat can be used
- E: maximum extra hours an individual boat can work past A
- E': maximum hours under A an individual boat can work
- λ: number of seasons
- μ: maximum number of times an individual boat can be reassigned between seasons
- $H_{s\ell}$: number of hours station $s \in S$ requires in season $\ell = 1, \ldots, \lambda$

The goal of the problem is to minimize the number of boats assigned among stations in S, so that each station's seasonal hours ($H_{s\ell}$) are met exactly. Further constraints include the necessity for a boat to work up between $A - E$ and $A + E$ hours total over all seasons. $\sum_\ell H_{s\ell}$ must be positive for each $s \in S$, otherwise we remove such a station from the problem. This formulation turns out to be an $\mathcal{NP}$-Hard optimization problem, even when λ is fixed at 3. The *decision version* of optimization problem BTS contain the above parameters, along with k, and asks whether or not it is possible to

meet all station hour demands with k or less boats in total. We prove this problem to be $\mathcal{NP}$-complete, which implies the $\mathcal{NP}$-hardness of BTS.

We reduce from the **numerical 3-dimensional matching** decision problem, or NTDM, where three equal-sized multi-sets—*i.e.* duplicates are allowed—of integers X, Y, and Z, are given, as well as an integer β. This problem asks whether there exists a subset M of $X \times Y \times Z$, where every element of $X \cup Y \cup Z$ belongs to exactly one triple $(x, y, z) \in M$. Furthermore, M must also satisfy $x + y + z = \beta$ for each $(x, y, z) \in M$. This problem is $\mathcal{NP}$-Complete, as shown in Garey and Johnson [40] by reduction from the **3-partition** problem.

We reduce an arbitrary instance (X, Y, Z, β) of NTDM to the decision problem BSP, by defining input so that $|S| \coloneqq k \coloneqq |X|$, $A \coloneqq \beta$, $E \coloneqq E' \coloneqq 0$, $\mu \coloneqq 3$, and let $H_{s\,1}$ be the sth element of the arbitrarily ordered set X, and similarly, let $H_{s\,2}$ be the sth element of Y, and $H_{s\,3}$ the sth element of Z; otherwise $H_{s\,\ell} = 0$ for $s \in S$, $\ell = 4, \ldots, \lambda$. Now it is easy to see that if there exists a solution M to NTDM for our instance at hand, then for each $(x, y, z) \in M$, we have $x + y + z = H_{s_x\,1} + H_{s_y\,2} + H_{s_z\,3} = \beta$, for a distinct triple of corresponding stations $(s_x, s_y, s_z) \in S^3$. Furthermore, since $A = \beta$, exactly one boat can be assigned to station s_x, move to s_y after season 1 is over, then onto s_z at the end of season 2, and come back to s_x—its home station—at the end of the 3rd period for the remainder of the λ seasons, yielding a total of 3 $(= \mu)$ relocations; let us call such a sequence a "path". Thus, there exists a solution to decision version of BSP, since all hours are covered feasibly by $|S| \leq k$ paths—each path corresponding to a distinct boat. But since $|S| \geq k$ by the fact that there exist kA hours to cover, the converse holds, *i.e.* $|S| = k$ implies that the hours must be covered by k distinct and entry-wise disjoint paths, yielding a feasible numerical 3-dimensional matching M of $X \times Y \times Z$, thus proving our claim.

Theorem B.0.2. *The* **boat sharing problem** *with input* $(S, B, A, E, E', \lambda, \mu, H_{s\ell})$ *is* $\mathcal{NP}$*-Hard, even when* $\mu \coloneqq 3$. □

References

[1] Biqmac solver. `http://biqmac.uni-klu.ac.at`. → *p. 5.*

[2] Rudy: a rudimentary graph generator. `http://www-user.tu-chemnitz.de/~helmberg/sdp_software.html`. → *p. 83.*

[3] T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, 2005. → *p. 79.*

[4] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5(1):13–51, 1995. → *pp. 4 and 58.*

[5] Miguel F Anjos, Bissan Ghaddar, Lena Hupp, Frauke Liers, and Angelika Wiegele. Solving k-way graph partitioning problems to optimality: The impact of semidefinite relaxations and the bundle method. 2012. → *pp. 5, 6, 20, 30, 52, 84, 97, 97, and 102.*

[6] Miguel F Anjos, Frauke Liers, Gregor Pardella, and Andreas Schmutzer. Engineering branch-and-cut algorithms for the equicut problem. In *Discrete Geometry and Optimization*, pages 17–32. Springer, 2013. → *p. 6.*

[7] David Applegate and William Cook. A computational study of the job-shop scheduling problem. *ORSA Journal on computing*, 3(2):149–156, 1991. → *p. 3.*

[8] Michael Armbruster, Marzena Fügenschuh, Christoph Helmberg, and Alexander Martin. Lp and sdp branch-and-cut algorithms for the minimum graph bisection problem: a computational comparison. *Mathematical Programming Computation*, 4(3):275–306, 2012. → *pp. 6 and 79.*

[9] Michael Armbruster, Christoph Helmberg, Marzena Fügenschuh, and Alexander Martin. On the graph bisection cut polytope. *SIAM Journal on Discrete Mathematics*, 22(3):1073–1098, 2008. → *pp. 6 and 58.*

[10] Egon Balas. Facets of the knapsack polytope. *Mathematical Programming*, 8(1):146–164, 1975. → *p. 43 and 43.*

[11] Francisco Barahona, Martin Grötschel, Michael Jünger, and Gerhard Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 36(3):493–513, 1988. → *p. 5.*

[12] Francisco Barahona and Ali Ridha Mahjoub. On the cut polytope. *Mathematical programming*, 36(2):157–173, 1986. → *p. 4.*

[13] E. Boros and P.L. Hammer. The max-cut problem and quadratic 0–1 optimization; polyhedral aspects, relaxations and bounds. *Annals of Operations Research*, 33(3):151–180, 1991. → *p. 4.*

[14] Gerald G Brown, Robert F Dell, and Robert A Farmer. Scheduling coast guard district cutters. *Interfaces*, 26(2):59–72, 1996. → *pp. 106 and 107.*

[15] Gerald G Brown, Robert F Dell, and Alexandra M Newman. Optimizing military capital planning. *Interfaces*, 34(6):415–425, 2004. → *p. 107.*

[16] Gerald G Brown, Robert F Dell, and R Kevin Wood. Optimization and persistence. *Interfaces*, 27(5):15–37, 1997. → *pp. 107 and 124.*

[17] Gerald G Brown, Clark E Goodman, and R Kevin Wood. Annual scheduling of atlantic fleet naval combatants. *Operations Research*, 38(2):249–259, 1990. → *pp. 106 and 107.*

[18] Lorenzo Brunetta, Michele Conforti, and Giovanni Rinaldi. A branch-and-cut algorithm for the equicut problem. *Mathematical Programming*, 78(2):243–263, 1997. → *p. 6.*

[19] Samuel Burer, Renato DC Monteiro, and Yin Zhang. Rank-two relaxation heuristics for max-cut and other binary quadratic programs. *SIAM Journal on Optimization*, 12(2):503–521, 2002. → *p. 6.*

[20] S. Chopra and MR Rao. The partition problem. *Mathematical Programming*, 59(1):87–115, 1993. → *pp. 6, 28, 29, and 37.*

[21] S. Chopra and MR Rao. Facets of the k-partition polytope. *Discrete Applied Mathematics*, 61(1):27–48, 1995. → *p. 6.*

[22] Michele Conforti, MR Rao, and Antonio Sassano. The equipartition polytope. i: Formulations, dimension and basic facets. *Mathematical Programming*, 49(1):49–70, 1990. → *pp. 7 and 32.*

[23] Michele Conforti, MR Rao, and Antonio Sassano. The equipartition polytope. ii: valid inequalities and facets. *Mathematical Programming*, 49(1-3):71–90, 1990. → *pp. 7 and 32.*

[24] Cory L Culver. Optimally scheduling distribution of the MH-60s helicopter and pilots to combat support (HC) squadrons. Technical report, DTIC Document, 2002. → *p. 107.*

[25] George B Dantzig. *Linear programming and extensions*. Princeton university press, 1998. → *p. 58.*

[26] Etienne de Klerk, Dmitrii V Pasechnik, and Joost P Warners. On approximate graph colouring and max-k-cut algorithms based on the θ-function. *Journal of Combinatorial Optimization*, 8(3):267–294, 2004. → *p. 5.*

[27] Michel Deza, Martin Grötschel, and Monique Laurent. Clique-web facets for multicut polytopes. *Mathematics of Operations Research*, 17(4):981–1000, 1992. → *p. 28.*

[28] Michel Marie Deza and Monique Laurent. *Geometry of cuts and metrics*, volume 15. Springer, 1997. → *pp. 4 and 6.*

[29] Eugenia Maria Kontopoulou Dimitrios Zeimpekis and Efstratios Gallopoulos. Text to matrix generator (tmg). `http://scgroup20.ceid.upatras.gr:8000/tmg/`. → *pp. 98 and 136.*

[30] Jonathan Eckstein, Cynthia A Phillips, and William E Hart. Pebbl 1.0 user guide. Technical report, RUTCOR Research Report RRR 19-2006. http://rutcor. rutgers. edu/pub/rrr/reports2006/19_2006. ps. Last access: 13 May, 2009. → *p. 80.*

[31] Andreas Eisenblätter. The semidefinite relaxation of the k-partition polytope is strong. In *Integer Programming and Combinatorial Optimization*, pages 273–290. Springer, 2006. → *pp. 6, 18, 19, 20, 29, 31, 31, 35, 36, 36, 38, and 58.*

[32] Carlos E Ferreira, Alexander Martin, C Carvalho de Souza, Robert Weismantel, and Laurence A Wolsey. Formulations and valid inequalities for the node capacitated graph partitioning problem. *Mathematical Programming*, 74(3):247–266, 1996. → *pp. 7, 40, 40, 40, 41, 42, and 42.*

[33] Carlos E Ferreira, Alexander Martin, C Carvalho de Souza, Robert Weismantel, and Laurence A Wolsey. The node capacitated graph partitioning problem: a computational study. *Mathematical Programming*, 81(2):229–256, 1998. → *pp. 7 and 40.*

[34] Paola Festa, Panos M Pardalos, Mauricio GC Resende, and Celso C Ribeiro. Randomized heuristics for the max-cut problem. *Optimization Methods and Software*, 17(6):1033–1058, 2002. → *p. 6.*

[35] Institute for Operations Research and the Management Sciences. 2012 informs annual meeting. `http://meetings2.informs.org/phoenix2012/index.php`. → *pp. 2 and 136.*

[36] Institute for Operations Research and the Management Sciences. 2013 informs computing society. `https://www.informs.org/Community/Conferences/ICS2013`. → *pp. 2, 3, and 80.*

[37] Oliver Friedmann, Thomas Dueholm Hansen, and Uri Zwick. Subexponential lower bounds for randomized pivoting rules for the simplex algorithm. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, pages 283–292. ACM, 2011. → *p. 58.*

[38] A. Frieze and M. Jerrum. Improved approximation algorithms for maxk-cut and max bisection. *Algorithmica*, 18(1):67–81, 1997. → *pp. 4, 5, 6, 18, 18, 18, 20, and 68.*

[39] Katsuki Fujisawa, Masakazu Kojima, Kazuhide Nakata, and Makoto Yamashita. Sdpa (semidefinite programming algorithm) users manualversion 6.2. 0. *Dept. Math. & Comp. Sciences, Tokyo Institute of Technology, Research Report B-308*, 1995. → *p. 81.*

[40] Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. Freeman New York, 1979. → *pp. 15, 29, 54, 137, and 140.*

[41] B. Ghaddar, M.F. Anjos, and F. Liers. A branch-and-cut algorithm based on semidefinite programming for the minimum k-partition problem. *Annals of Operations Research*, pages 1–20, 2009. → *pp. 5, 5, 6, 29, 30, 78, and 84.*

[42] M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995. → *pp. 4, 4, and 6.*

[43] M Grotschel and L Lovász. Combinatorial optimization. *Handbook of combinatorics*, 2:1541–1597, 1995. → *p. 4.*

[44] Martin Grötschel, László Lovász, and Alexander Schrijver. Geometric algorithms and combinatorial optimization. 1988. → *p. 4.*

[45] Martin Grötschel and Yoshiko Wakabayashi. A cutting plane algorithm for a clustering problem. *Mathematical Programming*, 45(1-3):59–96, 1989. → *pp. 7, 32, and 34.*

[46] Martin Grötschel and Yoshiko Wakabayashi. Facets of the clique partitioning polytope. *Mathematical Programming*, 47(1-3):367–387, 1990. → *pp. 7, 32, 33, 33, and 33.*

[47] United States Coast Guard. `http://www.uscg.mil/top/about`; Retrieved 2013-12-11. → *p. 105.*

[48] United States Coast Guard. `http://www.uscg.mil/acquisition/programs/acquisitionprograms.asp`; Retrieved 2013-12-11. → *p. 106.*

[49] United States Coast Guard. Coast guard snapshot 2012. `http://www.uscg.mil/top/about/doc/uscg_snapshot.pdf`; Retrieved 2013-12-11. → *p. 105.*

[50] Venkatesan Guruswami and Ali Kemal Sinop. Improved inapproximability results for maximum k-colorable subgraph. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 163–176. Springer, 2009. → *p. 5.*

[51] RA Hahn and Alexandra M Newman. Scheduling united states coast guard helicopter deployment and maintenance at clearwater air station, florida. *Computers & Operations Research*, 35(6):1829–1843, 2008. → *p. 107.*

[52] J. Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001. → *p. 4.*

[53] Christoph Helmberg, Franz Rendl, Robert J Vanderbei, and Henry Wolkowicz. An interior-point method for semidefinite programming. *SIAM Journal on Optimization*, 6(2):342–361, 1996. → *p. 58.*

[54] Ryusuke Hohzaki, Tatzuya Morimoto, and Sinsuke Omi. Flight scheduling for the sh-60j military helicopter. *Military Operations Research*, 16(2):5–17, 2011. → *p. 107.*

[55] HSOR.org. What is or. `http://www.hsor.org/what_is_or.cfm`; retrieved 2013-12-24. → *p. 106.*

[56] Akio Imai, Etsuko Nishimura, and Stratos Papadimitriou. The dynamic berth allocation problem for a container port. *Transportation Research Part B: Methodological*, 35(4):401–417, 2001. → *p. 107.*

[57] Xiaoyun Ji and John E Mitchell. Branch-and-price-and-cut on the clique partitioning problem with minimum clique size requirement. *Discrete Optimization*, 4(1):87–102, 2007. → *pp. 7, 32, and 45.*

[58] Volker Kaibel, Matthias Peinhardt, and Marc E Pfetsch. Orbitopal fixing. *Discrete Optimization*, 8(4):595–610, 2011. → *p. 5.*

[59] V. Kann, S. Khanna, J. Lagergren, and A. Panconesi. On the Hardness of Approximating Max k-Cut and Its Dual. *Chicago Journal of Theoretical Computer Science*, page 2, 1997. → *pp. 4 and 20.*

[60] Howard Karloff. How good is the goemans–williamson max cut algorithm? *SIAM Journal on Computing*, 29(1):336–350, 1999. → *pp. 4 and 20.*

[61] Richard M Karp. *Reducibility among combinatorial problems*. Springer, 1972. → *p. 16 and 16.*

[62] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for max-cut and other 2-variable csps? *SIAM Journal on Computing*, 37(1):319–357, 2007. → *pp. 4 and 20.*

[63] Donald E Knuth. *The sandwich theorem*. Stanford University, Department of Computer Science, 1993. → *pp. 5 and 56.*

[64] Nathan Krislock, Jérôme Malick, and Frédéric Roupin. Improved semidefinite bounding procedure for solving max-cut problems to optimality. *Mathematical Programming*, 143(1-2):61–86, 2014. → *p. 5.*

[65] Martine Labbé and F Aykut Özsoy. Size-constrained graph partitioning polytopes. *Discrete Mathematics*, 310(24):3473–3493, 2010. → *pp. 7, 31, 32, 32, 32, 33, 33, 36, and 39.*

[66] Sandia National Laboratories. Acro: A common repository for optimizers. `https://software.sandia.gov/trac/acro`. → *p. 80.*

[67] Sandia National Laboratories. Utilib library of c++ utilities. `https://software.sandia.gov/trac/utilib`. → *p. 81.*

[68] Jean B Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001. → *p. 7.*

[69] Monique Laurent. A comparison of the sherali-adams, lovász-schrijver, and lasserre relaxations for 0–1 programming. *Mathematics of Operations Research*, 28(3):470–496, 2003. → *p. 7.*

[70] Charles E Leiserson, Ronald L Rivest, Clifford Stein, and Thomas H Cormen. *Introduction to algorithms*. The MIT press, 2001. → *pp. 9, 12, and 16.*

[71] Frauke Liers, Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi. Computing exact ground states of hard ising spin glass problems by branch-and-cut. *New Optimization Algorithms in Physics*, pages 47–68, 2004. → *p. 5.*

[72] László Lovász. On the shannon capacity of a graph. *Information Theory, IEEE Transactions on*, 25(1):1–7, 1979. → *pp. 5, 15, and 56.*

[73] László Lovász and Alexander Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization*, 1(2):166–190, 1991. → *p. 7.*

[74] J.E. Mitchell. Realignment in the National Football League: Did they do it right? *Naval Research Logistics (NRL)*, 50(7):683–701, 2003. → *p. 7.*

[75] John E Mitchell. Branch-and-cut for the k-way equipartition problem. Technical report, Technical report, Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180, 2001. → *pp. 7, 7, 32, and 37.*

[76] George L Nemhauser and Laurence A Wolsey. *Integer and combinatorial optimization*, volume 18. Wiley New York, 1988. → *p. 9.*

[77] Yurii Nesterov, Arkadii Semenovich Nemirovskii, and Yinyu Ye. *Interior-point polynomial algorithms in convex programming*, volume 13. SIAM, 1994. → *p. 58.*

[78] United States Department of Homeland Security. Homeland security budget-in-brief fiscal year 2009. `http://www.dhs.gov/xlibrary/assets/budget_bib-fy2009.pdf#page=60`, p. 53, 2009. → *p. 105.*

[79] Goverment Printing Office. Homeland security act of 2002. `http://www.gpo.gov/fdsys/pkg/PLAW-107publ296/pdf/PLAW-107publ296.pdf`, 2002. → *p. 106.*

[80] Maarten Oosten, Jeroen HGC Rutten, and Frits CR Spieksma. The clique partitioning problem: facets and patching facets. *Networks*, 38(4):209–226, 2001. → *p. 33 and 33.*

[81] Christos H Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of computer and system sciences*, 43(3):425–440, 1991. → *p. 9.*

[82] Gabor Pataki and Levent Tunçel. On the generic properties of convex optimization problems in conic form. *Mathematical Programming*, 89(3):449–457, 2001. → *p. 58.*

[83] Svatopluk Poljak and Zsolt Tuza. Maximum cuts and large bipartite subgraphs. *DIMACS Series*, 20:181–244, 1995. → *p. 4.*

[84] Rong Qu, Edmund K Burke, Barry McCollum, Liam TG Merlot, and Sau Y Lee. A survey of search methodologies and automated system development for examination timetabling. *Journal of scheduling*, 12(1):55–89, 2009. → *p. 4.*

[85] Motakuri V Ramana, Levent Tunçel, and Henry Wolkowicz. Strong duality for semidefinite programming. *SIAM Journal on Optimization*, 7(3):641–662, 1997. → *p. 58.*

[86] F. Rendl, G. Rinaldi, and A. Wiegele. Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, 121(2):307–335, 2010. → *pp. 5 and 79.*

[87] Javier Salmeron, Robert F Dell, Gerald G Brown, and Anton Rowe. Capital investment planning aid (cipa)-an optimization-based decision-support tool to plan procurement and retirement of naval platforms. Technical report, DTIC Document, 2002. → *p. 107.*

[88] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975. → *p. 135.*

[89] Andrea Schaerf. A survey of automated timetabling. *Artificial intelligence review*, 13(2):87–127, 1999. → *p. 4.*

[90] Hanif D Sherali and Warren P Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430, 1990. → *p. 7.*

[91] Mathematical Optimization Society. 21st international symposium on mathematical programming. `http://ismp2012.mathopt.org/`. → *p. 3.*

[92] Michael M Sørensen. *Polyhedral computations for the simple graph partitioning problem.* Aarhus School of Business, Department of Accounting, Finance and Logistics, 2005. → *pp. 7, 32, and 40.*

[93] Jos F Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization methods and software*, 11(1-4):625–653, 1999. → *p. 81.*

[94] Levent Tunçel. On the slater condition for the sdp relaxations of nonconvex sets. *Operations Research Letters*, 29(4):181–186, 2001. → *p. 58.*

[95] Pál Turán. On an extremal problem in graph theory. *Mat. Fiz. Lapok*, 48(436-452):137, 1941. → *p. 28.*

[96] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, 1996. → *p. 58.*

[97] Radim Řehůřek. Gensim. `http://radimrehurek.com/gensim/`. → *p. 136.*

[98] Michael R Wagner and Zinovy Radovilsky. Optimizing boat resources at the US coast guard: Deterministic and stochastic models. *Operations Research*, 60(5):1035–1049, 2012. → *pp. xi, xi, xi, 107, 107, 108, 119, 120, 124, 124, 124, 125, 126, and 126.*

[99] Qian Wang, Xianyi Zhang, Yunquan Zhang, and Qing Yi. Augem: automatically generate high performance dense linear algebra kernels on x86 cpus. In *Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis*, page 25. ACM, 2013. → *p. 81.*

[100] Robert Weismantel. On the 0/1 knapsack polytope. *Mathematical Programming*, 77(3):49–68, 1997. → *p. 43.*

[101] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Englewood Cliffs, 2001. → *pp. 9, 41, and 54.*

[102] Henry Wolkowicz, Saigal Romesh, and Lieven Vandenberghe. *Handbook of semidefinite programming: theory, algorithms, and applications*, volume 27. Springer, 2000. → *p. 9.*

[103] Zhang Xianyi, Wang Qian, and Zhang Yunquan. Model-driven level 3 blas performance optimization on loongson 3a processor. In *Parallel and Distributed Systems (ICPADS), 2012 IEEE 18th International Conference on*, pages 684–691. IEEE, 2012. → *p. 81.*

[104] Makoto Yamashita, Katsuki Fujisawa, Mituhiro Fukuda, Kazuhiro Kobayashi, Kazuhide Nakata, and Maho Nakata. Latest developments in the sdpa family for solving large-scale sdps. In *Handbook on Semidefinite, Conic and Polynomial Optimization*, pages 687–713. Springer, 2012. → *pp. 81 and 82.*

[105] Makoto Yamashita, Katsuki Fujisawa, Mituhiro Fukuda, Kazuhide Nakata, and Maho Nakata. Algorithm 925: Parallel solver for semidefinite programming problem having sparse schur complement matrix. *ACM Transactions on Mathematical Software (TOMS)*, 39(1):6, 2012. → *pp. 81 and 82.*