12-2012

Polyhedral Approximations of Quadratic Semi-Assignment Problems, Disjunctive Programs, and Base-2 Expansions of Integer Variables

Frank Muldoon
Clemson University, fmuldoo@g.clemson.edu

Polyhedral Approximations of Quadratic Semi-Assignment Problems, Disjunctive Programs, and Base-2 Expansions of Integer Variables

---

A Dissertation
Presented to
the Graduate School of
Clemson University

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

---

by
Frank M. Muldoon
December 2012

Accepted by:
Dr. Warren P. Adams, Committee Chair
Dr. Margaret M. Wiecek
Dr. Matthew J. Saltzman
Dr. Pietro L. Belotti

Abstract

This research is concerned with developing improved representations for special families of mixed-discrete programming problems. Such problems can typically be modeled using different mathematical forms, and the representation employed can greatly influence the problem's ability to be solved. Generally speaking, it is desired to obtain mixed 0-1 linear forms whose continuous relaxations provide tight polyhedral outer-approximations to the convex hulls of feasible solutions. This dissertation makes contributions to three distinct problems, providing new forms that improve upon published works.

The first emphasis is on devising solution procedures for the classical *quadratic semi-assignment problem* (QSAP), which is an NP-hard 0-1 quadratic program. The effort begins by using a *reformulation-linearization technique* to recast the problem as a mixed 0-1 linear program. The resulting form provides insight into identifying special instances that are readily solvable. For the general case, the form is shown to have a tight continuous relaxation, as well as to possess a decomposable structure. Specifically, a *Hamiltonian decomposition* of a graph interpretation is devised to motivate a Lagrangian dual whose subproblems consist of families of separable acyclic minimum-cost network flows. The result is an efficient approach for computing tight lower bounds on the optimal objective value to the original discrete program. Extensive computational experience is reported to evaluate the tightness of the representation and the expedience of the algorithm.

The second contribution uses disjunctive programming arguments to model the convex hull of the union of a finite collection of polytopes. It is well known that the convex hull of the union of n polytopes can be obtained by lifting the problem into a higher-dimensional space using n auxiliary continuous (scaling) variables. When placed within a larger optimization problem, these variables must be restricted to be binary. This work examines an approach that uses fewer binary variables. The same scaling technique is employed, but the variables are treated as continuous by introducing a

logarithmic number of new binary variables and constraints. The scaling variables can now be substituted from the problem. Moreover, an emphasis of this work, is that specially structured polytopes lead to well-defined projection operations that yield more concise forms. These special polytopes consist of knapsack problems having SOS-1 and SOS-2 type restrictions. Different projections are defined for the SOS-2 case, leading to forms that serve to both explain and unify alternative representations for piecewise-linear functions, as well as to promote favorable computational experience.

The third contribution uses minimal cover and set covering inequalities to define the previously unknown convex hulls of special sets of binary vectors that are lexicographically lower and upper bounded by given vectors. These convex hulls are used to obtain ideal representations for base-2 expansions of bounded integer variables, and also afford a new perspective on, and extend convex hull results for, binary knapsack polytopes having weakly super-decreasing coefficients. Computational experience for base-2 expansions of integer variables exhibits a reduction in effort.

Dedication

I dedicate this to my father, Dr. John Dement Muldoon III, who was unable to see the completion of my work, but would have been proud of this achievement.

Acknowledgments

The completion of this work was made possible by the support of so many people. I would particular like to thank my wonderful girlfriend, Collin Emmel, for providing motivation and encouragement, my mom, Elizabeth Muldoon, for believing in me and supporting me throughout graduate school, my sister, Dr. Mary Beth Ross, who is the "real" doctor in the family, my sister, Dr. Tricia Brown, who showed me that a Ph.D. in math is possible, and my twin brother, William Muldoon, who is always there for me.

Also, I would like to thank my committee members, Dr. Margaret M. Wiecek, Dr. Matthew J. Saltzman, and Dr. Pietro L. Belotti, who always provided excellent probing questions, advised me in directions to pursue, and provided supportive feedback about my work.

Last, I own a huge debt of gratitude to my advisor Dr. Warren P. Adams who has worked tirelessly with me to complete my research. Without his knowledge, dedication, and support I would not be here today. I thank him for making my research much more enjoyable and stimulating.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Discrete optimization problems represent a large family of mathematical programs in which an objective function is to be optimized over a set of discrete variables. The objective function is usually linear, but can also be quadratic or polynomial. Mixed-discrete programs arise when a subset of the variables are restricted to lie within a discrete set, and the remaining variables are allowed to be continuous.

The difficulty with solving discrete programs is the combinatorial nature of the solution space. Given a pure 0-1 problem in n variables, so that each variable is restricted to be binary-valued, there exist 2^n possible solutions. In order to optimize discrete problems, each solution must be either explicitly enumerated, or implicitly examined. Here, binary vectors can be implicitly eliminated from consideration by deeming them to be non-optimal or infeasible. Since linear programs are typically far simpler to solve than their discrete counterparts, these former problems are often used to compute bounds on the latter, and thereby provide a fathoming mechanism for non-optimal solutions.

Discrete optimization problems have received considerable attention in the operations research literature [3, 5, 6, 7, 8] for two main reasons. First, there are a wide variety of important real-world problems that give rise to such forms. Examples include timetabling, scheduling, mixing, pooling, network, transportation, production planning, and cutting problems. Second, many of these problems are notoriously difficult to solve. Researchers and practitioners alike agree that there is a large discrepancy between the size and complexity of problems confronting society and those that can be adequately solved. The challenge here is to construct mixed 0-1 linear formulations of nonlinear and/or linear problems in such a way that the feasible regions to the continuous relaxations afford tight polyhedral relaxations. In this manner, the number of binary solutions that must be explicitly considered can be drastically reduced.

This dissertation contributes to the solving of discrete programs on three fronts. First, it presents a novel mixed 0-1 linear formulation for the famous quadratic semi-assignment problem, and examines and exploits the underlying mathematical structure. Second, it provides improved models for representing the union of a finite set of polytopes. Third, it gives an explicit convex hull representation for special sets of binary variables that are lexicographically bounded between two binary vectors.

The first contribution is presented in Chapter 2 with a study of the famous 0-1 quadratic program known as the *quadratic semi-assignment problem* (QSAP). This problem has applications in clustering, equipartitioning, coloring, and scheduling. Given m sets of n objects each, the problem is to select one object from each set so as to minimize an overall selection cost. The cost includes linear terms that reflect the choice of individual items, and quadratic terms that record interactions between pairs of items. Mathematically, the problem is NP-hard, and has n^m possible solutions. The study begins by using a reformulation-linearization technique [2] to rewrite the problem as a mixed 0-1 linear program in a higher-variable space. Then this formulation is used to identify several easily solvable special cases. These cases arise when the objective function is sparse relative to the number of nonzero coefficients. For general instances of the QSAP, a Hamiltonian decomposition of an underlying graph representation is strategically devised to create replicates of certain sets of variables, which are in turn equated using linking constraints. A Lagrangian dual is then formed by placing these linking constraints into the objective function. The resulting subproblems separate into a family of minimum-cost network flows. The dual multiplier adjustments in the master problem are accomplished via a combination of subgradient optimization and a dual ascent procedure. Computational results are reported for various data sets.

Chapter 3 presents the second contribution. Consider a collection of n polytopes, where it is desired that a set of decision variables $\boldsymbol{x}$ must lie within at least one such polytope. The convex hull of the union of these polytopes [4] can be obtained by multiplying each polytope by a distinct nonnegative scaling variable, by defining new variables to represent the scaled variables, by having the scaled variables sum to $\boldsymbol{x}$, and by restricting the scaling variables to sum to one. When contained within a larger optimization problem, these scaling variables must be binary. Our approach uses a method of [1] to relax the binary restrictions on the n scaling variables to nonnegativity by defining $\lceil \log_2 n \rceil$ new additional binary variables and constraints. The net effect is to reduce the number of binary variables from n to $\lceil \log_2 n \rceil$ at the expense of $\lceil \log_2 n \rceil$ new variables and equations. The continuous scaling variables can then be substituted from the problem. When the polytopes have special structures, suitable projection operations permit the rewriting of the problem in lower-dimensional spaces. Included within these special polytopes is the set of SOS-2 restrictions. Our projections serve to explain and unify different ideal representations for piecewise-linear functions. They also lead to favorable computational experience for balanced transportation problems having piecewise-linear objective functions.

Chapter 4 continues our study of tight polyhedral outer-approximations of discrete sets by providing an explicit algebraic description of the convex hull of the base-2 expansion of a bounded integer variable. Whereas base-2 expansions of bounded integer variables are classical in discrete optimization, and whereas tight polyhedral outer-approximations of 0-1 linear programs are widely-recognized as being a key ingredient for improved solution techniques, these two concepts have not been combined to motivate convex hull (ideal) representations of such expansions. Given an integer variable, this chapter provides an ideal form in the original variable space by explicitly describing the additional inequalities needed to capture the convex hull. The representation requires at most $\lceil \log_2 n \rceil - 1$ minimal-cover type inequalities, where n is the number of permissible realizations of the discrete variable.

Our arguments in Chapter 4 are based on a lexicographic ordering of binary vectors. Given a binary vector, the convex hull of the set of binary vectors that is lexicographically less than or equal to this chosen vector is described using minimal cover inequalities. This convex hull form is applied to model base-2 expansions of bounded integer variables. A similar description gives the convex hull for the set of vectors that is lexicographically greater than or equal to a given binary vector. We combine these results to characterize the convex hull of the set of 0-1 vectors that is lexicographically bounded between two binary vectors. This characterization leads to the ideal representation of binary knapsack polytopes having weakly super-decreasing coefficients, where the knapsack constraint can have both lower and upper bounds.

Bibliography

[1] Adams, W.P. and Henry, S.M., "Base-2 Expansions for Linearizing Products of Functions of Discrete Variables," *Operations Research*, to appear, (manuscript 2011).

[2] Adams, W.P., and Sherali, H.D., "A Tight Linearization and an Algorithm for Zero-One Quadratic Programming Problems," *Management Science*, Vol. 32, No. 10, pp 1274–1290, 1986.

[3] Ahuja, R.K., Magnanti, T.L., and Orlin, J.B., *Network Flows. Theory, Algorithms and Applications*, Prentice Hall, Englewood Cliffs, 1993.

[4] Balas, E., "Disjunctive Programming," *Annals of Discrete Mathematics*, Vol. 5, pp 3–51, 1979.

[5] Bazaraa, M.S., Jarvis, J.J., and Sherali, H.D., *Linear Programming and Network Flows*, John Wiley & Sons, Hoboken, NJ, 2010.

[6] Bazaraa, M.S., Sherali, H.D., and Shetty, C.M., *Nonlinear Programming: Theory and Algorithms*, John Wiley & Sons, Hoboken, NJ, 2006.

[7] Nemhauser, G.L. and Wolsey, L.A., *Integer and Combinatorial Optimization*, John Wiley & Sons, New York, NY, 1999.

[8] Wolsey, L.A., *Integer Programming*, John Wiley & Sons, New York, NY, 1998.

Chapter 2

Lower Bounds of the Quadratic Semi-Assignment Problem using a Graph Decomposition of the Level-1 RLT Formulation

2.1 Definition and Formulation

The *quadratic semi-assignment problem* (QSAP) is an NP-hard, nonlinear 0-1 optimization program that is expressed mathematically as:

$$P : \min \left\{ \sum_{i=1}^{m} \sum_{j=1}^{n} C_{ij} x_{ij} + \sum_{i=1}^{m-1} \sum_{j=1}^{n} \sum_{k=i+1}^{m} \sum_{l=1}^{n} D_{ijkl} x_{ij} x_{kl} : \boldsymbol{x} \in \boldsymbol{X},\ \boldsymbol{x} \text{ binary} \right\},$$

where

$$\boldsymbol{X} \equiv \left\{ \boldsymbol{x} \in \mathbb{R}^{mn} : \sum_{j=1}^{n} x_{ij} = 1 \;\forall\, i = 1, \ldots, m;\ x_{ij} \geq 0 \;\forall\, (i,j),\ i = 1, \ldots, m, j = 1, \ldots, n \right\}. \tag{2.1}$$

Problem P is interpreted as follows. Given m sets consisting of n objects each, the problem seeks to select exactly one object from each set so that the total cost of selection is minimized. Here, for each $i \in M \equiv \{1, \ldots, m\}$ and $j \in N \equiv \{1, \ldots, n\}$, the binary variable x_{ij} equals 1 if object

i from set j is selected, and 0 otherwise. The equality restrictions in $\boldsymbol{X}$ enforce that exactly one item be selected from each set. For each (i, j) with $i \in M$ and $j \in N$, the scalar C_{ij} is the cost of selecting item i from set j while for each (i, j, k, ℓ) with $i \in M$, $j \in N$, $k \in M$ and $\ell \in N$ having $i < k$, the scalar $D_{ijk\ell}$ records the cost of selecting item i from set j *and* item k from set ℓ. In this manner, the cost C_{ij} is incurred if and only if $x_{ij} = 1$ and the cost $D_{ijk\ell}$ is incurred if and only if $x_{ij} = x_{k\ell} = 1$. The quadratic terms in the objective function include, without loss of generality, only those (i, j, k, ℓ) terms having $i < k$ because $x_{ij}^2 = x_{ij}$ and because $x_{ij}x_{k\ell} = x_{k\ell}x_{ij}$. (For notational convenience and unless otherwise stated, the indices i and k lie in the set M and the indices j and ℓ lie in the set N.) Applications of the quadratic semi-assignment arise in such areas as clustering, equipartitioning, colorings and scheduling [9, 11, 24, 25].

The difficulty with solving Problem P is the combinatorial nature of the solution space. Given m sets of n items each, there exist n^m possible solutions that must be considered. The special case having $D_{ijk\ell} = 0$ is trivially solvable as m separable linear minimization problems. Some problem instances can be reformulated in terms of a network when certain subsets of the $D_{ijk\ell}$ coefficients are zero. For these cases, the problem is known to be polynomially solvable [14]. In addition, the authors [15, 16] demonstrated a method to solve special cases in which the QSAP is represented as a reducible graph. However, the general case is known to be NP-hard [20].

Solving the general case of the QSAP first involves finding tight lower bounds on a linear relaxation of this quadratic problem. This has been accomplished in the literature via an RLT formulation of the QSAP. A theoretical paper [21] proves which level of the RLT is necessary for the optimal value of the continuous relaxation to be the same as the optimal value of the original formulation. The paper [7] shows that the best reduction of the QSAP using a quadratic pseudo-Boolean function with nonnegative coefficients is the level-1 RLT. The papers [15, 16] give lower bounds for the QSAP by decomposing it into reducible graphs within a Lagrangian dual framework for special instances of the QSAP in which subsets of the $D_{ijk\ell}$ are zero. The authors [18] demonstrate that the level-1 RLT formulation of the *symmetric* QSAP gives integer optimal solutions in many cases. In addition, the paper [19] shows that tighter lower bounds of the QSAP are achieved by using the level-1 RLT formulation and introducing additional cut- and clique-inequalities.

In general, as demonstrated above, the level-1 RLT representation provides good lower bounds for the QSAP. But solving this RLT representation using powerful linear programming software such as CPLEX becomes more difficult as the size of the problem increases because the number of constraints and variables grow exponentially. As a result, it is necessary to develop an efficient algorithm to handle such large linear programming instances in order to obtain tight lower bounds for larger instances of the QSAP.

The rest of this chapter is organized as follows. Section 2 gives the standard level-1 RLT representation of the QSAP, shows how this formulation can be reduced via variable substitutions, gives a family of redundant constraints, and explains the network structure present. Section 3 follows with a novel explanation of the level-1 RLT formulation in terms of a graph and provides several new readily-solvable special cases. Next, the general case of the QSAP is considered in Section 4 where a *Hamiltonian decomposition* on the graph representation of the formulation is used to decompose the problem into separable networks. This decomposition requires replicate variables $\boldsymbol{x}$ with linking constraints, which are added to the objective function via dual multipliers. Section 5 describes a combination subgradient algorithm and dual ascent procedure to solve the Lagrangian dual created by the decomposition. This section concludes with computational comparisons on the lower bounds given by our method compared to the actual lower bounds for large instances of the QSAP provided by CPLEX. Section 6 offers concluding remarks.

2.2 Mixed 0-1 Linear Representation

Our approach for obtaining lower bounds on Problem P is to convert it into an equivalent mixed 0-1 linear program. There are various ways to construct such linear representations and the resulting forms can have different sizes and relaxation strengths. Indeed, some linear forms are available in the literature and we choose one given in [7] defined as the *best reduction* of the QSAP. This linearization has significant mathematical structure which can be exploited and dominates existing forms in terms of relaxation strength.

2.2.1 Level-1 RLT Form

We construct a mixed 0-1 linear representation of Problem P by applying the two-step, reformulation-linearization technique (RLT) of [22, 23]. Depending on the specific implementation, different level representations result. The level-1 RLT form [2, 3] (see also [23, pages 104–105] for treatment of equality restrictions) is obtained in the following manner. The *reformulation* step consists of multiplying every constraint $\sum_j x_{ij} = 1 \; \forall \; i$ and every nonnegativity restriction $x_{ij} \geq$

$0 \ \forall\ (i,j)$ found in $\boldsymbol{X}$ by each variable $x_{k\ell}$, and by appending these new (redundant) restrictions to the problem. It then substitutes $x_{ij}^2 = x_{ij}$ for all (i,j) and enforces $x_{ij}x_{k\ell} = x_{k\ell}x_{ij}$ for all (i,j,k,ℓ) with $i < k$. (This binary substitution eliminates each term of the form $x_{ij}x_{i\ell}$ from the problem by setting it equal to x_{ij} when $j = \ell$ and equal to 0 when $j \neq \ell$.) The *linearization* step substitutes a continuous variable $w_{ijk\ell}$ throughout the objective function and constraints for each occurrence of the product $x_{ij}x_{k\ell}$ having (i,j,k,ℓ) with $i \neq k$. The result is below, where we use the notation $LP1$ to denote the first *linear* representation of Problem P.

$$
\begin{aligned}
LP1: \quad & \min \quad \sum_i \sum_j C_{ij}x_{ij} + \sum_i \sum_j \sum_{k>i} \sum_\ell D_{ijk\ell}w_{ijk\ell} \\
& \text{s.t.} \quad \sum_j w_{ijk\ell} = x_{k\ell} && \forall\ (i,k,\ell), i \neq k \qquad (2.2) \\
& \qquad\quad w_{ijk\ell} = w_{k\ell ij} && \forall\ (i,j,k,\ell), i < k \qquad (2.3) \\
& \qquad\quad w_{ijk\ell} \geq 0 && \forall\ (i,j,k,\ell), i \neq k \qquad (2.4) \\
& \qquad\quad \boldsymbol{x} \in \boldsymbol{X},\ \boldsymbol{x} \text{ binary}
\end{aligned}
$$

Equations (2.2) of $LP1$ result from multiplying the constraints $\sum_j x_{ij} = 1$ for all i found within $\boldsymbol{X}$ by each variable $x_{k\ell}$. Those constraints having $i = k$ and those variables $w_{ijk\ell}$ having $i = k$ are effectively removed from consideration by the $x_{ij}^2 = x_{ij}$ substitutions. Inequalities (2.4) result from multiplying the $x_{ij} \geq 0$ restrictions of $\boldsymbol{X}$ by each $x_{k\ell}$ having $k \neq i$, and equations (2.3) are the linearized versions of the restrictions that $x_{ij}x_{k\ell} = x_{k\ell}x_{ij}$ for all (i,j,k,ℓ) having $i < k$.

The RLT theory stipulates that the nonlinear 0-1 program, Problem P, can be optimized by solving the mixed 0-1 linear program, Problem $LP1$. This follows since, for any $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{w}})$ feasible to $LP1$, the restrictions (2.2)–(2.4) enforce that $\hat{w}_{ijk\ell} = \hat{x}_{ij}\hat{x}_{k\ell}$ for all (i,j,k,ℓ) having $i \neq k$. Furthermore, $\nu(LP1) = \nu(P)$ and $\nu(\overline{LP1}) \leq \nu(P)$, where $\nu(LP1)$, $\nu(P)$, and $\nu(\overline{LP1})$ denote the optimal objective function values of Problems $LP1$, P, and $\overline{LP1}$ respectively, with $\overline{LP1}$ denoting the continuous relaxation of $LP1$ obtained by replacing the $\boldsymbol{x}$ binary restrictions with $\boldsymbol{x} \geq \boldsymbol{0}$. Throughout the remainder of the chapter, the notation $\nu(\bullet)$ and $\overline{\bullet}$ is used to denote the optimal objective function value and the continuous relaxation of the program $\bullet$, respectively, where the continuous relaxation replaces $\boldsymbol{x}$ binary with $\boldsymbol{x} \geq \boldsymbol{0}$.

The following lemma identifies sets of constraints that are redundant within $\boldsymbol{X}$ of LP1.

Lemma 1

Given any $i \in M$ and $k \in M$ with $i < k$, consider the two equations

$$\sum_j x_{ij} = 1 \quad \text{and} \quad \sum_\ell x_{k\ell} = 1 \tag{2.5}$$

found in $\boldsymbol{X}$, and the $2n$ equations

$$\sum_j w_{ijk\ell} = x_{k\ell} \;\forall\; \ell \quad \text{and} \quad \sum_\ell w_{ijk\ell} = x_{ij} \;\forall\; j \tag{2.6}$$

implied by (2.2) and (2.3), where the left equations in (2.6) result from multiplying the first restriction in (2.5) by each $x_{k\ell}$ for all ℓ, and where the right equations in (2.6) result from multiplying the second restriction in (2.5) by each x_{ij} for all j, invoking (2.3) where needed. Then in the presence of (2.6), either equation in (2.5) is implied by the other.

Proof

Follows directly by summing the left equations of (2.6) over ℓ and the right equations of (2.6) over j. $\square$

Lemma 1 gives rise to a useful result, stated as a corollary below.

Corollary 1

For any $p \in \{1, \ldots, m-1\}$, consider all $(m-p)$ possible (i, k) pairs of Lemma 1 having $i = p, \ldots, m-1$ and $k = i + 1$, and compute the $2(m-p)n$ associated equations of the type (2.6) for these $(m-p)$ pairs to obtain

$$\begin{aligned} &\sum_j w_{ij(i+1)\ell} = x_{(i+1)\ell} \;\forall\; (i, \ell),\; p \le i \le m-1 \quad \text{and} \\ &\sum_\ell w_{ij(i+1)\ell} = x_{ij} \;\forall\; (i, j),\; p \le i \le m-1. \end{aligned} \tag{2.7}$$

Then, in the presence of (2.7), the equation $\sum_j x_{pj} = 1$ implies the restrictions $\sum_j x_{ij} = 1$ for all $i \in \{p+1, \ldots, m\}$ of $\boldsymbol{X}$.

Proof

For each $i \in \{p, \ldots, m-1\}$, the $2n$ equations of (2.7) are the equations of (2.6) with $k = i+1$, so that Lemma 1 gives us $\sum_j x_{ij} = \sum_j x_{(i+1)j}$, establishing the result. $\square$

Problem $LP1$ has exploitable structure. The variables $w_{ijk\ell}$ having $i > k$ can be substituted from the problem using (2.3), and then (2.3) can be removed. This substitution reduces the size in terms of both the numbers of variables and constraints. Upon performing this substitution and then making the simplification of Corollary 1 for $p = 1$ to replace $\boldsymbol{x} \in \boldsymbol{X}$ with the single equation $\sum_j x_{1j} = 1$, as all associated $2(m-1)n$ equations in (2.7) are present in $LP1$, we obtain an equivalent mixed 0-1 linear representation of P which we denote by $LP2(T)$. Here, $LP2(S)$ is the optimization problem given below as a function of sets $S \subseteq T$ where

$$T \equiv \{(i,k) : i < k\}. \tag{2.8}$$

$$\begin{aligned}
LP2(S): \quad \min \quad & \sum_i \sum_j C_{ij} x_{ij} + \sum_{(i,k)\in S} \sum_j \sum_\ell D_{ijk\ell} w_{ijk\ell} & & \\
\text{s.t.} \quad & \sum_j w_{ijk\ell} = x_{k\ell} & \forall\, (i,k,\ell) \text{ with } (i,k) \in S \quad & (2.9) \\
& \sum_\ell w_{ijk\ell} = x_{ij} & \forall\, (i,j,k) \text{ with } (i,k) \in S \quad & (2.10) \\
& w_{ijk\ell} \geq 0 & \forall\, (i,j,k,\ell) \text{ with } (i,k) \in S \quad & (2.11) \\
& \sum_j x_{1j} = 1, \ \boldsymbol{x} \text{ binary} & &
\end{aligned}$$

These modifications to obtain $LP2(T)$, while reducing the problem size, do not change the set of feasible solutions to either $LP1$ or to its continuous relaxation, and they do not change the objective function value at any point. Thus, $\nu(P) = \nu(LP1) = \nu(LP2(T))$ and $\nu(P) \geq \nu(\overline{LP1}) = \nu(\overline{LP2(T)})$. Note that an interpretation of (2.9) and (2.10) is that each equation $\sum_j x_{ij} = 1$ of $\boldsymbol{X}$ is multiplied by every $x_{k\ell}$ with $k > i$ to obtain (2.9) and each equation $\sum_\ell x_{k\ell} = 1$ of $\boldsymbol{X}$ is multiplied by every x_{ij} with $i < k$ to obtain (2.10).

We pose two remarks relative to the general structure of $LP2(S)$ that will be later used.

Remark 1

Depending on the structure of the objective function coefficients $D_{ijk\ell}$, more concise RLT representations of Problem P than $LP2(T)$ may be available. In particular, suppose only a subset of the coefficients $D_{ijk\ell}$ are nonzero, and that instead of $LP2(T)$, we construct $LP2(S')$ where $S' \subseteq T$ has $S' \equiv \{(i,k) : D_{ijk\ell} \neq 0 \text{ for some } (j,l)\}$. Then $LP2(S')$ is a potentially reduced form of $LP2(T)$ that preserves two important properties. First, the replacement of $\boldsymbol{x} \in \boldsymbol{X}$ in $LP1$ with $\sum_j x_{1j} = 1$ remains valid if the conditions of Corollary 1 with $p = 1$ are met; that is, if the set S' contains all (i,k) pairs with $k = i+1$. Second, and again assuming the conditions of Corollary 1 are met, $LP2(S')$ is an equivalent linear formulation of P with the same relaxation strength as $LP2(T)$. Equivalence to P is established by showing that, for binary $\boldsymbol{x}$, $w_{ijk\ell} = x_{ij}x_{k\ell}$ for all (i,j,k,ℓ) with $(i,k) \in S'$. Consider any such $w_{ijk\ell}$, say w_{rstu}, and observe that (2.9) and (2.11) with $(i,k,\ell) = (r,t,u)$ imply $w_{rstu} = 0$ when $x_{tu} = 0$. (In fact, $w_{rst\ell} = 0$ for all ℓ such that $x_{t\ell} = 0$.) Similarly, (2.10) and (2.11) with $(i,j,k) = (r,s,t)$ imply $w_{rstu} = 0$ when $x_{rs} = 0$. On the other hand, suppose that $x_{rs} = x_{tu} = 1$. Then (2.10) with $(i,j,k) = (r,s,t)$ gives $\sum_\ell w_{rst\ell} = 1$. Since $w_{rst\ell} = 0$ for all $\ell \neq u$ as $x_{t\ell} = 0$ for all $\ell \neq u$, we have $w_{rstu} = 1$. The relaxation strength remains unchanged since, given any $(\widehat{\boldsymbol{x}}, \widehat{\boldsymbol{w}})$ feasible to $LP2(S')$, the point $(\widehat{\boldsymbol{x}}, \bar{\boldsymbol{w}})$ having

$$\bar{w}_{ijk\ell} = \begin{cases} \widehat{w}_{ijk\ell} & \text{if } (i,k) \in S' \\ \widehat{x}_{ij}\widehat{x}_{k\ell} & \text{otherwise} \end{cases} \qquad \forall\, (i,j,k,\ell) \text{ with } (i,k) \in T$$

is feasible to $LP2(T)$ with the same objective value.

Remark 2

Given any nonnegative $\widehat{\boldsymbol{x}} \in \boldsymbol{X}$ in $LP2(T)$, the reduced linear program over $\boldsymbol{w}$ decomposes into $(m(m-1))/2$ bipartite networks. Letting $LP2_{\widehat{\boldsymbol{x}}}(T)$ denote $LP2(T)$ with $\boldsymbol{x}$ fixed to some such $\widehat{\boldsymbol{x}}$, we have

$$\begin{aligned} LP2_{\widehat{\boldsymbol{x}}}(T) : \sum_i \sum_j C_{ij}\widehat{x}_{ij} + \sum_{(i,k)\in T} \min \Bigg\{ & \sum_j \sum_\ell D_{ijk\ell} w_{ijk\ell} : \sum_j w_{ijk\ell} = \widehat{x}_{k\ell} \;\forall\, \ell, \\ & \sum_\ell w_{ijk\ell} = \widehat{x}_{ij} \;\forall\, j, \;\; w_{ijk\ell} \geq 0 \;\forall\, (j,\ell) \Bigg\}. \end{aligned} \tag{2.12}$$

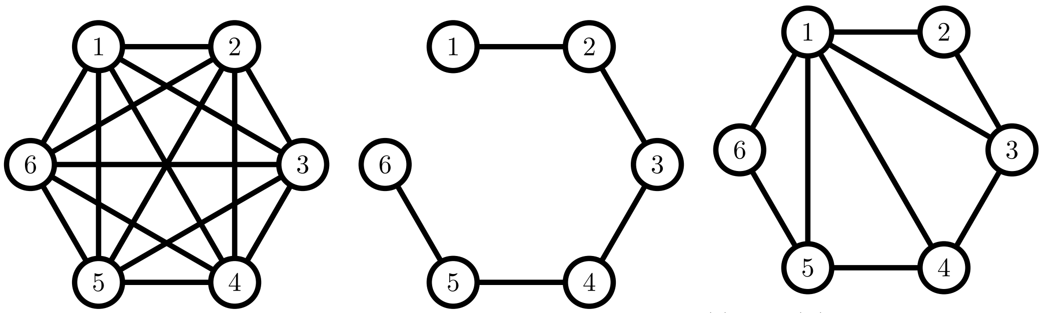
The authors of [17] also observed a variation of Remark 2 in terms of a level-1 RLT representation of the *single allocation hub location problem under congestion* (SAHLPC) which is a specific case of the *generalized quadratic assignment problem* (GQAP). The GQAP has the same constraints and objective function as the QSAP defined by Problem P but includes additional constraints other than those found in $\boldsymbol{X}$ of (2.1). For the SAHLPC, the additional constraints only include the $\boldsymbol{x}$ variables so when there exists a fixed $\hat{\boldsymbol{x}}$ these constraints are unnecessary leaving a similar formulation given by (2.12).

2.3 Graph Representation and Readily-Solvable Cases

The process of generating the restrictions of $LP2(S)$ from P via the RLT approach of multiplying the constraints in $\boldsymbol{X}$ by variables $x_{k\ell}$ can be envisioned using a graph representation. The graph provides insight into Corollary 1, the equivalence between $LP1$ and $LP2(S)$ for select sets S, and also promotes special cases that can be more easily solved. In addition, a factorization of this graph will serve in the next section to motivate a Lagrangian decomposition for solving $LP2(T)$, where the subproblems are comprised of acyclic shortest path networks.

To begin, recall the construction of (2.9) and (2.10) of $LP2(S)$ from $\boldsymbol{X}$. Given any $(i,k) \in S$, the n equations in (2.9) are computed by multiplying $\sum_j x_{ij} = 1$ with the variable $x_{k\ell}$ for each ℓ, while the n equations in (2.10) are computed by multiplying $\sum_\ell x_{k\ell} = 1$ with the variable x_{ij} for each j. Such multiplications can be represented by a graph as follows. Define a graph G on m nodes, where node i corresponds to the i^{th} equation of $\boldsymbol{X}$; that is, $\sum_j x_{ij} = 1$. Connect two nodes i and k, with $i < k$, of the graph if equation i of $\boldsymbol{X}$ is multiplied by $x_{k\ell}$ for each ℓ *and* equation k of $\boldsymbol{X}$ is multiplied by x_{ij} for each j. This defines G in terms of $LP2(S)$ by having an edge connecting nodes i and k if $(i,k) \in S$. In this manner, the graph corresponding to $LP2(T)$, with T as defined in (2.8), is complete on m nodes. The complete graph for $m = 6$ is given in Figure 2.1a.

This graph interpretation provides insight into Corollary 1 and the equivalence of $LP1$ with $LP2(S)$ for select sets S. Consider any $S \subseteq T$, and the associated graph G. Corollary 1 states that if that path which sequentially progresses from node p to m in ascending node order lies within G, then the equality $\sum_j x_{pj} = 1$ and the restrictions in (2.9) and (2.10) having $(i,k) = (i, i+1)$ for $i \in \{p, \ldots, m-1\}$ combine to imply $\sum_j x_{ij} = 1$ for all $i \in \{p+1, \ldots, m\}$. However, since the nodes can be arbitrarily numbered, the Corollary can be restated in terms of G as follows: Given any two nodes r and s that are connected by a path in G, the equality $\sum_j x_{rj} = 1$ and the restrictions (2.9)



(a) $LP2(T)$ with T of (2.8). (b) $LP2(S')$ with S' of (2.13). (c) $LP2(S)$ solvable via n networks.

Figure 2.1: Graph representations of Problem $LP2(S)$ with $m = 6$ for different sets S.

and (2.10) associated with those pairs $(i, k) \in T$ having node i connected to node k by an arc within this path, combine to imply that $\sum_j x_{tj} = 1$ for each node t on this path. As a consequence, the replacement of $\boldsymbol{x} \in \boldsymbol{X}$ in $LP1$ with any equation of the form $\sum_j x_{ij} = 1$ for $i \in M$ in forming $LP2(S)$ preserves equivalence between these two problems provided that the graph G, defined in terms of S, is connected.

Graph G can be used to identify readily solvable special cases of Problem P. Specifically, Problem $LP2(S)$ is an equivalent reformulation of $LP1$ that can be solved as a (directed) acyclic shortest path network when the set S has the associated graph G corresponding to a Hamiltonian path. This statement follows from two observations. Here, we assume without loss of generality via a possible renumbering of the nodes, that the path progresses sequentially from node 1 to node m so that $LP2(S)$ has $S = S'$ with

$$S' \equiv \{(i, k) : k = i + 1\}, \tag{2.13}$$

as in Figure 2.1b for the $m = 6$ case. First, since a Hamiltonian path is a connected graph, the argument above establishes $LP2(S')$ as an equivalent reformulation of $LP1$. Second, by construction of G, the number of edges incident with a node i denotes the number of occurrences of each variable x_{ij} for all j within (2.9) and (2.10), and the existence of an edge between two nodes i and k indicates the occurrence of each variable $w_{ijk\ell}$ for all (j, ℓ), twice with coefficient of 1. Thus, each of the variables $w_{ijk\ell}$ have the desired number of two nonzero entries in (2.9) and (2.10), and the Hamiltonian path ensures that for every i, each variable x_{ij} appears exactly once when $i \in \{1, m\}$ since nodes 1 and m are end nodes and exactly twice when $i \in \{2, \ldots, m-1\}$. Noting $\sum_j x_{1j} = 1$ from

$LP2(S')$, including the redundant equation $-\sum_j x_{mj} = -1$, and negating each equation in (2.9), an acyclic shortest path network from node 1 to m emerges. The shortest path network has $2n(m-1)+2$ nodes and $nm + n^2(m-1)$ arcs, as $LP2(S')$ has $2n(m-1)+2$ equality constraints, nm variables x_{ij}, and $n^2(m-1)$ variables $w_{ijk\ell}$. The network is acyclic because flow must progress along arc pairs x_{ij} to $w_{ijk\ell}$ and arc pairs $w_{ijk\ell}$ to $x_{k\ell}$ for those (i, j, k, ℓ) having $k = i + 1$. An example is given below.

Example

Consider an instance of Problem P having $m = n = 3$, where the nonzero $D_{ijk\ell}$ objective function coefficients are indexed by the set S' of (2.13). The acyclic shortest path problem in 14 nodes and 27 arcs is depicted in Figure 2.2, where the arc corresponding to each variable x_{ij} is suitably labeled, and where an arc connecting some x_{ij} to $x_{(i+1)\ell}$ represents the variable $w_{ij(i+1)\ell}$. Costs are suppressed for ease of presentation. Nodes 1 and 14 have supplies of $+1$ and -1 respectively, and all other nodes have supplies of 0. (The reader is referred to the appendix for an explicit listing of all constraints in $LP2(S')$, numbered to coincide with the node labels.)

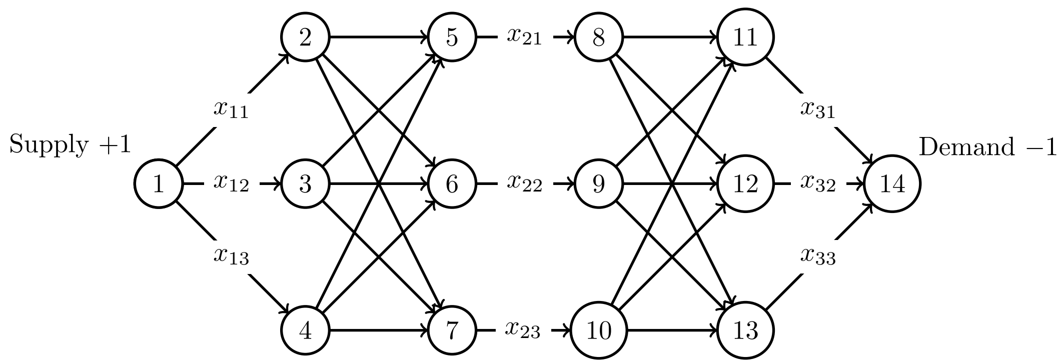


Figure 2.2: $LP2(S')$ network with S' of (2.13) when $m = n = 3$.

Observe that the representation of Problem $LP2(S')$ for S' of (2.13) as a Hamiltonian path on G identifies other solvable special cases. In particular, suppose if a single node i and all the incident edges of node i are removed from G and the remaining nodes and edges form a Hamiltonian path on $m-1$ nodes. Within Problem P itself, each of the n possible binary solutions to $\sum_j x_{ij} = 1$ will yield a QSAP with m reduced by 1. By definition of G, the graph associated with each of these reduced problems is the original graph G, less node i and all incident edges. Hence, Problem P can be solved via n shortest path networks. This scenario is depicted in Figure 2.1c for $i = 1$ and $m = 6$, where the reduced graph is the Hamiltonian path on nodes 2 through 6. The process extends to multiple nodes. Given that G contains a node-induced subgraph on p nodes that is a Hamiltonian path, Problem P can be solved by optimizing a shortest path network for each of the n^{m-p} feasible binary realizations. This number becomes computationally prohibitive as p decreases so that alternate general solution strategies are needed.

2.4 Decomposition Strategy

We use the readily-solvable case of the previous section to motivate a decomposition approach to solve Problem P. Unlike the instances where $S = S'$ of (2.13) within $LP2(S)$, this more general case with $S = T$ of (2.8) does not contain a network structure since each variable x_{ij} appears $(m-1)$ times in (2.9) and (2.10), as opposed to at most twice for S'. Our approach is to replicate the variables x_{ij} so that the problem decomposes. We then equate these replicates using "linking equations." The idea is to construct a Lagrangian dual to $LP(T)$ with the linking equations placed into the objective function as complicating constraints, and with the subproblems consisting of separable shortest path problems.

This decomposition of $LP(T)$ can be motivated in terms of the graph G. By construction, each node i of G corresponds to a constraint $\sum_j x_{ij} = 1$ of $\boldsymbol{X}$ and each edge (i,k) of G corresponds to a set of constraints $\sum_j w_{ijk\ell} = x_{k\ell}$ for all ℓ and $\sum_\ell w_{ijk\ell} = x_{ij}$ for all j of (2.9) and (2.10) respectively. By replicating the variables x_{ij} associated with the various nodes i, we can decompose G into separable factors so that each replicated variable x_{ij} and each variable $w_{ijk\ell}$ lies in exactly one factor. In particular, to take advantage of the shortest path network structure associated with Hamiltonian paths, we desire a *Hamiltonian decomposition* of G into Hamiltonian paths.

Recall from [8, 26] that a Hamiltonian decomposition of a complete graph G consists of a collection of graphs on this same set of nodes such that each graph is a Hamiltonian path, and each edge is used exactly once in some such graph. Also recall that for an even number m of nodes in G, Hamiltonian decompositions into $m/2$ graphs exist, and such a decomposition can be computed so that the r^{th} factor has node r as one endpoint and node $v = m+1-r$ as the other endpoint. (For the remainder of the chapter we assume that m is even so that $m/2$ is integral.) Specifically, these decompositions separate G with edge set $E \equiv \{(i,k) : i < k\}$ into factors $G_1, G_2, \ldots, G_{m/2}$ having edge sets $E_1, E_2, \ldots, E_{m/2}$ respectively so that each G_r is a Hamiltonian path defined on the same node set as G, $E_r \cap E_s = \emptyset$ for all $r \neq s$, and $\cup_{r=1}^{m/2} E_r = E$. (For notational convenience, the index r will lie in the set $\{1, \dots, m/2\}$.) Thus, by replicating each variable x_{ij} exactly $m/2$ times, once for each graph factor, Problem $LP(T)$ can be expressed in terms of $m/2$ separable shortest path problems, joined by linking constraints that equate these variables. Figure 3 below shows a Hamiltonian decomposition of the complete graph G on $m = 6$ nodes. An algebraic representation

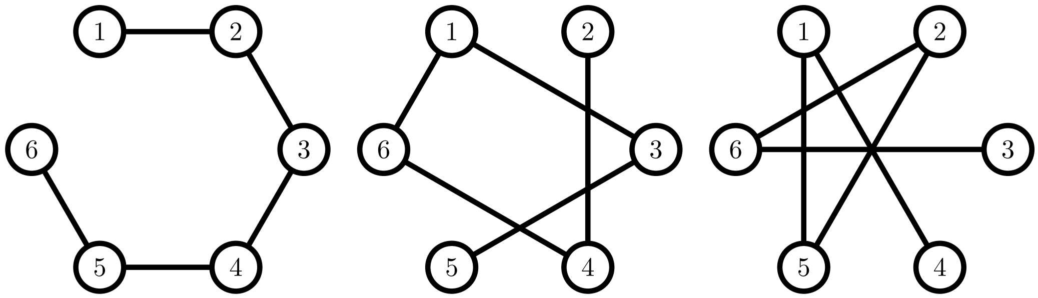


Figure 2.3: Hamiltonian decomposition of the complete graph G on $m = 6$ nodes.

of Problem $LP(T)$ that uses the edge sets E_r based on this factorization is given below. Here, each graph G_r has a distinct set of variables x_{ij}^r associated with it, and the costs C_{ij}^r can be any real numbers satisfying

$$\sum_{r=1}^{m/2} C_{ij}^r = C_{ij} \;\; \forall\, (i,j). \tag{2.14}$$

$$
\begin{array}{llll}
LP3(T) : \min & \displaystyle\sum_r \left(\sum_i \sum_j C_{ij}^r x_{ij}^r + \sum_{(i,k)\in E_r} \sum_j \sum_\ell D_{ijk\ell} w_{ijk\ell} \right) & & \\
\text{s.t.} & \displaystyle\sum_j w_{ijk\ell} = x_{k\ell}^r & \forall\, (i,k,\ell,r) \text{ with } (i,k) \in E_r & (2.15) \\
& \displaystyle\sum_\ell w_{ijk\ell} = x_{ij}^r & \forall\, (i,j,k,r) \text{ with } (i,k) \in E_r & (2.16) \\
& x_{ij}^1 = x_{ij}^2 = \cdots = x_{ij}^{\frac{m}{2}} & \forall\, (i,j) & (2.17) \\
& w_{ijk\ell} \geq 0 & \forall\, (i,j,k,\ell) \text{ with } (i,k) \in T & (2.18) \\
& \displaystyle\sum_j x_{rj}^r = 1 & \forall\, r & (2.19) \\
& \displaystyle\sum_j x_{vj}^r = 1 & \forall\, r \;\; \text{with } v = m+1-r & (2.20) \\
& \boldsymbol{x} \text{ binary} & &
\end{array}
$$

Equations (2.17) ensure that $LP3(T)$ is equivalent to $LP2(T)$ in the sense that, given a feasible solution to either problem, there exists a feasible solution to the other problem with the same objective value. While equations (2.19) are redundant for $r \geq 2$ by (2.17) and equations (2.20) are redundant from Corollary 1, they are useful in decomposing our upcoming Lagrangian dual. The Lagrangian dual is computed by placing equations (2.17) into the objective function using multipliers $\boldsymbol{\pi}$ so that for each $r \in \{1, \dots, m/2-1\}$, the multiplier π_{ij}^r corresponds to the constraint $x_{ij}^r = x_{ij}^{r+1}$. The Lagrangian dual is given below.

$$LD: \ \max \ \left\{\theta(\boldsymbol{\pi}): \ \boldsymbol{\pi} \in \mathbb{R}^{mn(m/2-1)}\right\}$$

where

$$\theta(\boldsymbol{\pi}) = \min \ \left\{\sum_r \left(\sum_i \ \sum_j (C_{ij}^r + \pi_{ij}^{r-1} - \pi_{ij}^r)x_{ij}^r + \sum_{(i,k)\in E_r} \sum_j \ \sum_\ell D_{ijk\ell} w_{ijk\ell}\right) : \right.$$
$$\left. (2.15), (2.16), (2.18), (2.19), (2.20), \ \mathbf{x} \text{ binary}\right\}.$$

Problem LD is separable over r for fixed $\boldsymbol{\pi}$ so that an optimal solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{w}})$ and optimal objective value $\nu(\theta(\boldsymbol{\pi}))$ can be computed by solving $m/2$ separable linear programs, as shown below.

$$\nu(\theta(\boldsymbol{\pi})) = \sum_r^{m/2} \gamma_r(\boldsymbol{\pi})$$

where, for each $r \in \{1, \dots, m/2\}$, the value $\gamma_r(\boldsymbol{\pi})$ is computed as follows.

$$\begin{aligned}
\gamma_r(\boldsymbol{\pi}) = \min \quad & \sum_i \sum_j (C_{ij}^r + \pi_{ij}^{r-1} - \pi_{ij}^r)x_{ij}^r + \sum_{(i,k)\in E_r} \sum_j \sum_\ell D_{ijk\ell} w_{ijk\ell} & \\
\text{s.t.} \quad & \sum_j w_{ijk\ell} = x_{k\ell}^r & \forall\ (i,k,\ell) \text{ with } (i,k) \in E_r \\
& \sum_\ell w_{ijk\ell} = x_{ij}^r & \forall\ (i,j,k) \text{ with } (i,k) \in E_r \\
& w_{ijk\ell} \geq 0 & \forall\ (i,j,k,\ell) \text{ with } (i,k) \in E_r \\
& \sum_j x_{rj}^r = 1, \ \sum_j x_{vj}^r = 1, & \text{with } v = m+1-r \\
& \boldsymbol{x} \text{ binary} &
\end{aligned}$$

Observe that for each r, the optimization problem to compute $\gamma_r(\boldsymbol{\pi})$ gives, for each (i,j), the value $\hat{x}_{ij}$ and the values $\hat{w}_{ijk\ell}$ for all (i,k,ℓ) having $(i,k) \in E_r$. Also observe that this problem has the same constraint structure as $LP2(E_r)$, with the added constraint $\sum_j x^r_{vj} = 1$ and the restriction $\sum_j x_{1j} = 1$ of $LP2(E_r)$ replaced with $\sum_j x^r_{rj} = 1$. Consequently, as each set E_r corresponds to a Hamiltonian path on the nodes of G with node r as one endpoint and node $v = m + 1 - r$ as the other, each problem $\gamma_r(\boldsymbol{\pi})$ can be solved as a shortest path network. Optimizing $\theta(\boldsymbol{\pi})$ for a fixed $\boldsymbol{\pi}$ thus reduces to $m/2$ shortest path networks. This structure is used in the following section for solving the continuous relaxation of $LP3(T)$ via LD.

2.5 Solving the Lagrangian Dual

The objective is to solve the continuous relaxation of $LP3(T)$, denoted $\overline{LP3(T)}$, which also solves $\overline{LP2(T)}$, by obtaining a vector $\boldsymbol{\pi} \in \mathbb{R}^{mn(m/2-1)}$ corresponding to equations (2.17) that maximizes LD. Since LD has the integrality property [12] because the continuous relaxation of the program to compute each $\gamma_r(\boldsymbol{\pi})$ has integral extreme points, an optimal $\boldsymbol{\pi}$ will be any set of optimal duals to (2.17). For such an optimal, $\nu(\theta(\boldsymbol{\pi})) = \nu(\overline{LP3(T)})$. Our approach initializes a set of values for $\boldsymbol{\pi}$, and then systematically updates these values based on the primal solutions to $\theta(\boldsymbol{\pi})$. Our solution method is a two step process in which a deflected subgradient algorithm is employed followed by a dual ascent procedure.

2.5.1 Deflected Subgradient Optimization

We improve the value of LD first by means of a subgradient method presented in [4, 6, 10, 13]. This is an iterative procedure that updates the dual multipliers $\boldsymbol{\pi}$ by taking steps based upon a subgradient of LD. An initial solution, $\boldsymbol{\pi}^0$, is selected for LD and iteratively updated based on solutions to the the primal subproblem $\theta(\boldsymbol{\pi}^t)$. In general, this update is given as

$$\boldsymbol{\pi}^{t+1} = \boldsymbol{\pi}^t + \lambda^t \boldsymbol{d}^t,\ t = 1, 2, \ldots, T,$$

where λ^t and $\boldsymbol{d}^t$ are the step size and direction, respectively, at iteration t with T being the total number of iterations. In general, the direction is given as a subgradient of $\theta(\boldsymbol{\pi}^t)$, denoted $\boldsymbol{\xi}^t$. We choose to normalized the subgradient so the direction $\boldsymbol{d}^t$ is given as

$$\boldsymbol{d}^t = \frac{\boldsymbol{\xi}^t}{||\boldsymbol{\xi}^t||},\ t = 1, 2, \ldots, T.$$

The step size, λ^t, is defined as

$$\lambda^t = \frac{\beta^t(\Theta - \theta(\boldsymbol{\pi}^t))}{||\boldsymbol{\xi}^t||},\ t = 1, 2, \ldots, T,$$

where Θ is an upper bound on $LP3(T)$ and β^t is given as some scalar in $0 < \beta^t < 2$. Various step sizes, λ^t, have been proposed and for specific problems it is shown in [13] that if $\lambda^t \to 0$ and $\sum_{t=0}^{\infty} \lambda^t = \infty$ then the sequence of dual multipliers $\boldsymbol{\pi}^t$ will optimally converge. In practice, the algorithm tends to stall before optimality is achieved and so a careful modification to the update method must be undertaken to prevent this and these modifications are generally problem dependent.

An advantage of the subgradient method is that it is not computationally expensive per iteration. The updates to the dual multipliers are readily available because $\theta(\boldsymbol{\pi})$ is linear and therefore differentiable (except at corner points) and so a subgradient is easily found from the optimal solution. In practice this procedure has difficultly achieving optimality because the dual multipliers tend to oscillate wildly and therefore result in a slow convergence rate. In addition, without a careful choice of λ^t this procedure can stall before optimality. In fact, the subgradient method is normally stopped after a fixed number of iterations when a suitable bound for $\theta(\boldsymbol{\pi})$ is found.

A variation on the previous algorithm, the deflected subgradient method [6], improves the traditional subgradient method. In general, near the optimal solution, it takes a relatively small step size in order to see improvement in LD. Towards this end, the direction, $\boldsymbol{d}^t$, is *deflected* at each iteration by the previous direction in order to prevent the step size from being reduced too much. For the deflected subgradient method, let

$$\boldsymbol{d}^t = \frac{\boldsymbol{\xi}^t + \phi^t \boldsymbol{d}^{t-1}}{||\boldsymbol{\xi}^t + \phi^t \boldsymbol{d}^{t-1}||},\ t = 1, 2, \ldots, T, \tag{2.21}$$

where ϕ^t deflects the current direction by the previous previous direction. Note that $\phi^1 = 0$ and

$\boldsymbol{d}^0 = \mathbf{0}$. Generally, ϕ^t is given as

$$\phi^t = \frac{||\boldsymbol{\xi}^t||}{||\boldsymbol{d}^{t-1}||},\ t = 2, 3, \ldots, T,$$

so that $\boldsymbol{d}^t$ bisects the angle between $\boldsymbol{\xi}^t$ and $\boldsymbol{d}^{t-1}$. The step size is also modified based on the new deflected direction and is given as

$$\lambda^t = \frac{\beta^t(\Theta - \theta(\boldsymbol{\pi}^t))}{||\boldsymbol{\xi}^t + \phi^t \boldsymbol{d}^{t-1}||},\ t = 1, 2, \ldots, T. \tag{2.22}$$

2.5.2 Dual Ascent

After the subgradient algorithm has terminated, we improve the lower bound by preforming a variation on a dual ascent procedure proposed in [1]. This procedure creates a nondecreasing sequence of lower bounds starting with the best solution found using the subgradient algorithm. Let $\hat{\boldsymbol{\pi}}$ be the set of multipliers that give the best objective value, denoted LB, after the subgradient algorithm is complete. We rewrite LD as

$$\begin{aligned} LD' = LB + \min \Bigg\{ \sum_r \Bigg(\sum_i \sum_j \overline{C}^r_{ij} x^r_{ij} + \sum_{(i,k)\in E_r} \sum_j \sum_\ell \overline{D}_{ijk\ell} w_{ijk\ell} \Bigg) &: \\ (2.15), (2.16), (2.18), (2.19), (2.20),\ \mathbf{x} \text{ binary} \Bigg\}&. \end{aligned}$$

where $\overline{C}^r_{ij}$ for all i, j, and r, and $\overline{D}_{ijk\ell}$ for all i, j, k, and ℓ, are the reduced costs on the arcs of the network $\theta(\hat{\boldsymbol{\pi}})$. The goal is to gradually increase the value of LD' by modifying the minimization problem. Observe that using the current reduced costs from $\theta(\hat{\boldsymbol{\pi}})$ results in the minimization problem having an optimal value of zero.

First, LD' is adjusted without affecting the optimal solution of the minimization problem. Modify the values of $\overline{C}^r_{ij}$ and $\overline{D}_{ijk\ell}$ by adding multiples of the constraints of (2.15) and (2.16) to the objective function. The goal is to increase the value of $\overline{C}^r_{ij}$ by reducing some of the $\overline{D}_{ijk\ell}$ to zero using the constraints given above. This is done by first considering every $\overline{C}^r_{k\ell}$ with the corresponding constraint $\sum_j w_{ijk\ell} = x^r_{k\ell}$ of (2.15). It is easy to find the smallest $\overline{D}_{ijk\ell}$ for some j, say j', such that $\overline{C}^r_{k\ell}$ can be increased by $\overline{D}_{ij'k\ell}$ and all $\overline{D}_{ijk\ell}$'s corresponding to the $w_{ijk\ell}$'s of the constraint are decreased by $\overline{D}_{ij'k\ell}$. A similar argument is used to increase some of the $\overline{C}^r_{ij}$'s again by decreasing the $\overline{D}_{ijk\ell}$'s corresponding to the constraint $\sum_\ell w_{ijk\ell} = x_{ij}^r$ of (2.16).

Adding multiples of the constraints (2.15) and (2.16) does not improve the optimal objective value of LD' because the minimization problem still has an optimal objective of zero. But, the $\overline{C}_{ij}^r$ can be modified within the LD framework. Recall that the linking constraints $x_{ij}^r = x_{ij}^{r+1}$ of (2.17) allow the original C_{ij}^r to be any set of real coefficients as long as $\sum_r C_{ij}^r = C_{ij}$. The same is true for the coefficients $\overline{C}_{ij}^r$. Define $\overline{C}_{ij}$ for all i and j as $\overline{C}_{ij} = \sum_r \overline{C}_{ij}^r$. The coefficients $\overline{C}_{ij}^r$ are allowed to be any nonnegative values, since they are reduced costs, as long as they sum to $\overline{C}_{ij}$. After the adjustments to these coefficients, the dual ascent procedure takes the following steps. If the minimization problem has a positive objective function value say $z^* > 0$ then redefine LB to be $LB = LB + z^*$. The reduced costs $\overline{C}_{ij}^r$ and $\overline{D}_{ijk\ell}$ of the current minimization problem are found and redistributed as described above. The new minimization problem is then solved again. This is process is repeated until the updates to LB fall below a certain threshold. As a result of the dual ascent procedure, a nondecreasing sequence of updates to LD' are created approaching the optimal objective function value.

2.5.3 Combination Algorithm

Our procedure for finding the solution to LD is to perform T iterations of the deflected subgradient algorithm and then implement the dual ascent procedure until it stalls. After some initial testing, a value of $T = 600$ subgradient iterations provided the best results results and is used for all trials. For the deflected subgradient algorithm, we initialize $\boldsymbol{\pi}^0 = \mathbf{0}$. At each iteration, $\boldsymbol{d}^t$ is defined as the deflected gradient given in (2.21). We find a naive upper bound Θ by finding C_{ip_i}, where p_i is defined as $p_i = \text{argmin}_j\{C_{ij}\}$, and setting $x_{ip_i} = 1$ with $x_{ij} = 0$ for $j \neq p_i$ for each $i = 1, \ldots, m$. A step length, λ^t, given by (2.22), is taken along the direction $\boldsymbol{d}^t$ of (2.21) with $\beta^t = 0.25$. If the algorithm fails to improve over 20 iterations, then the incumbent dual vector is reinstated and β^t is divided by two.

After the deflected subgradient algorithm terminates, the dual ascent procedure is started using the best solution found with the previous method. As mentioned above, the reduced cost coefficients $\overline{C}_{ij}^r$ and $\overline{D}_{ijk\ell}$ are found from the best solution provided by the deflected subgradient algorithm. In addition, the $\overline{C}_{ij}^r$ are increased as much as possible by decreasing the $\overline{D}_{ijk\ell}$ using (2.15) and (2.16). After this, we define $\overline{C}_{ij} = \sum_r \overline{C}_{ij}^r$ and evenly distribute $\overline{C}_{ij}$ to each $\overline{C}_{ij}^r$ by setting these coefficients as

$$\overline{C}_{ij}^r = \frac{\overline{C}_{ij}}{m/2}\ \forall\, i, j, r.$$

The dual ascent procedure is repeated until the update to LB falls below 0.5.

2.5.4 Computational Results

The LD solution technique described above is coded using Python 2.7 and executed it on a Dell Precision T3500 workstation with an Intel Xeon W3690 processor (6 cores at 3.46GHz) with 24 GB RAM. The $\frac{m}{2}$ network subproblems $\gamma_r(\boldsymbol{\pi})$ are efficiently solved using the *reaching algorithm* of [5] because each subproblem, for a fixed $\boldsymbol{\pi}$, is an acyclic minimum-cost network flow problem. Solving the network subproblems is then incorporated into the deflected subgradient algorithm and the dual ascent procedure described above. The optimal value of $\overline{LP3(T)}$, which is equivalent to the optimal value of $\overline{LP2(T)}$, is denoted $\nu(\overline{LP3(T)})$ and was found using ILOG CPLEX 12.0 on the same machine as given above.

We tested our method using two different data sets for various values of m and n. First, the coefficients $D_{ijk\ell}$ in the objective function of Problem P are replaced with the product terms $f_{ik}d_{j\ell}$ similar to objective function found in [15, 18]. This instance occurs in computer processing where m processes need to be assigned to n processors. First, the computation time required to run process i on processor j is C_{ij}. During the computation period, processes i and k exchange f_{ik} units of information where $d_{j\ell}$ is the time needed to move one unit of information from processor j to processor ℓ. We assume that the processors are arranged on a mesh of size $a \times a = n$ and the distance between processors j and ℓ is the mesh distance. In addition, d_{jj}, for $j = 1, \dots, n$, are defined to be the max mesh distance because assigning two processes to the same processor incurs a penalty cost. The objective is to minimize the total computational time of the m processes using the n processors and is given as

$$\sum_i \sum_j C_{ij} + \sum_i \sum_j \sum_{k>i} \sum_\ell f_{ik} d_{j\ell} x_{ij} x_{k\ell}.$$

The number of processors is fixed to be $n = 16, 25, 36$ corresponding to grid sizes of $4 \times 4 = 16$, $5 \times 5 = 25$, and $6 \times 6 = 36$, respectively. Different numbers of processes m are attempted for each fixed number of processors n and we note that as the number of processors increases the number of processes that can be solved in a reasonable amount of time decreases.

We observed that when trying to solve LD, the solution $\theta(\mathbf{0})$, that is, the initial solution with $\boldsymbol{\pi}^0 = \mathbf{0}$, provides a close approximation of the optimal relaxation value $\nu(\overline{LP3(T)})$ and so we did not attempt the deflected subgradient or dual ascent. The results are summarized in Table 2.1 which is divided into eight columns. The first and second columns give the values for the parameters n and m, respectively. Next, columns three and four show the optimal relaxation value, that is $\nu(\overline{LP3(T)})$, and the time in seconds it took CPLEX to find this value. Columns five and six give $\theta(\mathbf{0})$ and the time in seconds our code took to find these solutions. Last, columns seven and eight give the Gap defined as

$$\text{Gap} = \frac{\nu(\overline{LP3(T)}) - \theta(\mathbf{0})}{\nu(\overline{LP3(T)})} \times 100\%$$

which is the percentage difference between our solution and the optimal solution and the percentage decrease in time our algorithm achieved compared to CPLEX, respectively.

n	m	$\nu(\overline{LP3(T)})$	CPLEX Time(Sec.)	$\theta(\mathbf{0})$	$\theta(\mathbf{0})$ Time(Sec.)	Gap(%)	Decrease in Time(%)
16	16	679.00	1.24	664.25	0.12	2.17	90.32
16	20	1096.50	4.18	1073.20	0.19	2.12	95.45
16	26	1874.50	8.67	1850.85	0.32	1.26	96.31
16	30	2472.50	21.74	2448.47	0.42	0.97	98.07
16	36	3543.00	62.07	3499.28	0.67	1.23	98.92
16	40	4420.00	94.00	4382.20	0.74	0.86	99.21
16	46	5778.50	236.41	5722.70	1.00	0.97	99.58
16	50	6887.50	352.11	6826.96	1.26	0.88	99.64
16	56	8753.50	871.63	8684.46	1.56	0.79	99.82
16	60	9713.00	973.44	9647.60	1.75	0.67	99.82
16	66	11937.00	1840.43	11881.24	2.07	0.47	99.89
16	70	13240.50	2951.80	13148.26	2.29	0.70	99.92
25	26	1824.50	39.46	1797.77	0.72	1.47	98.18
25	30	2620.00	87.71	2586.47	0.99	1.28	98.87
25	36	3565.50	260.12	3516.06	1.43	1.39	99.45
25	40	4405.50	485.41	4357.80	1.75	1.08	99.64
25	46	5814.50	947.90	5763.83	2.36	0.87	99.75
25	50	6770.00	2029.35	6698.48	2.71	1.06	99.87
25	56	8799.00	3590.72	8721.43	3.56	0.88	99.90
25	60	10051.00	5511.00	9966.63	4.02	0.84	99.93
36	36	3622.50	620.54	3575.61	2.73	1.29	99.56
36	40	4373.00	1540.11	4320.35	3.46	1.20	99.78
36	46	5908.00	3859.92	5843.74	4.65	1.09	99.88
36	50	6828.50	6126.58	6763.04	5.50	0.96	99.91

Table 2.1: QSAP Instances for Processor Problems

The second set of problems considered are similar to those found in [7, 15] where the coefficients, C_{ij} and $D_{ijk\ell}$, of Problem P are selected uniform randomly from the integer set $\{0, 1, \ldots, 99\}$. For these instances, $\theta(\mathbf{0})$ provided a relatively weak bound on the optimal solution, which is demonstrated in Table 2.2, and so we preformed the deflected subgradient and dual ascent procedure described above. This table is divided into nine columns where the first two columns give the values for the parameters n and m, respectively. Next, columns three and four give the optimal relaxation value, that is $\nu(\overline{LP3(T)})$, and the time in seconds it took CPLEX to find this value. Column five gives $\theta(\mathbf{0})$. (The time to find $\theta(\mathbf{0})$ for all instances was never greater than 7 seconds.) Columns six and seven give the best bound, denoted LD Best, found using the combination subgradient and dual ascent procedure and the time in seconds, respectively. Next, column eight gives the Gap defined this time as

$$\text{Gap} = \frac{\nu(\overline{LP3(T)}) - LD \text{ Best}}{\nu(\overline{LP3(T)})} \times 100\%$$

which is the percentage difference between LD Best and the optimal solution provided by CPLEX. Last, column nine displays the the percentage decrease in time our algorithm achieved compared to CPLEX. A dash in column nine indicates that our algorithm took longer than CPLEX. For this set

n	m	$\nu(\overline{LP3(T)})$	CPLEX Time(Sec.)	$\theta(\mathbf{0})$	Best LD	Best LD Time(Sec.)	Gap(%)	Decrease in Time(%)
15	30	5419.00	42.00	2404.67	5299.33	61.89	2.21	-
15	36	7560.38	67.12	3283.00	7398.36	92.30	2.14	-
15	40	9100.87	91.18	3893.10	8954.39	116.77	1.61	-
15	46	11682.20	179.81	4921.96	11481.89	158.32	1.71	11.95
15	50	13684.73	285.77	5677.56	13497.65	188.02	1.37	34.21
15	56	16853.87	1198.59	6774.82	16567.94	248.01	1.70	79.31
15	60	19102.47	1637.25	7622.90	18819.25	285.11	1.48	82.59
10	50	18498.10	72.57	7797.52	18371.39	103.45	0.68	-
15	50	13684.73	285.77	5677.56	13497.65	188.02	1.37	34.21
20	50	11051.20	2041.50	4529.20	10738.66	323.26	2.83	84.17
25	50	9281.60	1886.01	3737.88	8920.78	490.78	3.89	73.98
30	50	8167.67	11480.25	3368.28	7762.21	708.91	4.96	93.82
35	50	7303.32	7116.42	2930.28	6901.47	958.09	5.50	86.54
40	50	6585.51	11689.52	2693.24	6042.09	1243.69	8.25	89.36

Table 2.2: QSAP Instances with Uniform Integer Random Coefficients

of problems, we ran $T = 600$ iterations of the deflected subgradient algorithm and then performed the dual ascent procedure until the update to LD was less than 0.5. The number of dual ascent steps for the instances of Table 2.2 were between 3 and 18.

For the first set of instances tested, found in Table 2.1, our decomposition provided by $\theta(\mathbf{0})$

gave a very close approximation to the optimal solution $\nu(\overline{LP3(T)})$ as is seen in columns 3 and 5 of Table 2.1. In addition, as m and n increase, the percentage Gap actually decreased between our solution and the solution given by CPLEX. In most cases, especially larger instances, our solution procedure took just a fraction of the time that CPLEX needed to find the optimal value.

The second set of problems tested, those with uniform integer coefficients found in Table 2.2, did not have a relatively close relaxation value for $\theta(\mathbf{0})$ so 600 iterations of the subgradient method was preformed for all instances and then the dual ascent procedure iterated until the updates to LD were less than 0.5. First, consider the cases where n is fixed to 15 and m is increased, it is seen in Table 2.2 that after termination of our algorithm, our best solution is always within 3% of $\nu(\overline{LP3(T)})$ for each m. In addition, as m increased, the percentage decrease in time for our algorithm took compared to the CPLEX decreased. In the instance where $m = 60$ and $n = 15$, our algorithm took 82.59% less time than CPLEX to get a solution that was within 1.48% of $\nu(\overline{LP3(T)})$.

For the cases where m is fixed to 50 and n increases, the results show that our algorithm finds a solution that is within 4% of $\nu(\overline{LP3(T)})$ when $n = 10, 15, 20, 25$, and between 4% and 9% for $n = 30, 35, 40$. In general, for $n = 20, 25, 30, 35, 40$, our algorithm obtains a solution that is between 2% and 9% of $\nu(\overline{LP3(T)})$ in 73% to 93% less time than CPLEX.

2.6 Conclusion

When solving difficult nonlinear optimization problems, such as the QSAP, the first step is generally to find a linear relaxation of the problem and then solve the new representation in order to obtain a tight lower bound. Reformulations using the RLT have proven effective in providing tight lower bounds for other difficult optimization problems so we applied the level-1 RLT representation to the QSAP. For this instance, the level-1 RLT representation of the QSAP has a network structure which aids in our solution technique. Regardless of the structure of any RLT formulation, as the size of the original problem grows, the number of constraints and variables in the RLT representation increases exponentially which hinders achieving optimal solutions of the relaxation using traditional linear programming software. To combat this, we provided a novel solution technique using a Lagrangian dual that exploits, with a Hamiltonian path decomposition, the network structure of the RLT formulation of the QSAP in order to achieve lower bounds for larger values of m and n in less time than using traditional methods like a powerful linear programming solver such as CPLEX.

Preliminary computational results suggest that our solution technique, for a linear relaxation of the level-1 RLT formulation of the QSAP for larger values of m and n, offers a reduction in computation time for a close approximation of the optimal value. For instances of the QSAP representing assigning processes to processors, our decomposition provides an extremely accurate approximation of the optimal value in a fraction of the time without using the combination subgradient and dual ascent procedure. For QSAP instances with more general integer coefficients, our solution method required solving the Lagrangian dual many times but still came relatively close to the optimal relaxation value in a shorter amount of time compared to CPLEX for larger values of m and n.

This chapter presents the first step for solving larger instances of the QSAP by finding tight lower bounds using the level-1 RLT formulation. Future research in the area will attempt to embed our solution technique within an efficient branch-and-bound routine in order to solve to integer optimality larger instances of the QSAP than have been previously attempted.

2.7 Appendix

The following gives all the constraints of Problem $LP2(S)$ corresponding to the nodes of the network displayed in Figure 2.2 for Problem P when $m = 3$ and $n = 3$.

$$
\begin{aligned}
x_{11} + x_{12} + x_{13} &= +1 \quad (1)\\
w_{1121} + w_{1122} + w_{1123} - x_{11} &= 0 \quad (2)\\
w_{1221} + w_{1222} + w_{1223} - x_{12} &= 0 \quad (3)\\
w_{1321} + w_{1322} + w_{1323} - x_{13} &= 0 \quad (4)\\
-w_{1121} - w_{1221} - w_{1321} + x_{21} &= 0 \quad (5)\\
-w_{1122} - w_{1222} - w_{1322} + x_{22} &= 0 \quad (6)\\
-w_{1123} - w_{1223} - w_{1323} + x_{23} &= 0 \quad (7)\\
w_{2131} + w_{2132} + w_{2133} - x_{21} &= 0 \quad (8)\\
w_{2231} + w_{2232} + w_{2233} - x_{22} &= 0 \quad (9)\\
w_{2331} + w_{2332} + w_{2333} - x_{23} &= 0 \quad (10)\\
-w_{2131} - w_{2231} - w_{2331} + x_{31} &= 0 \quad (11)\\
-w_{2132} - w_{2232} - w_{2332} + x_{32} &= 0 \quad (12)\\
-w_{2133} - w_{2233} - w_{2333} + x_{33} &= 0 \quad (13)\\
-x_{31} - x_{32} - x_{33} &= -1 \quad (14)\\
x_{ij} &\geq 0 \quad \forall\, (i,j)\\
w_{ijk\ell} &\geq 0 \quad \forall\, (i,j,k\ell) \text{ with } (i,k) \in S
\end{aligned}
$$

Bibliography

[1] Adams, W.P., and Johnson, T.A., "Improved Linear Programming-Based Lower Bounds for the Quadratic Assignment Problem," *Quadratic Assignment and Related Problems, DIMACS Series*, Vol. 16, pp 43–75, 1994.

[2] Adams, W.P., and Sherali, H.D., "A Tight Linearization and an Algorithm for Zero-One Quadratic Programming Problems," *Management Science*, Vol. 32, No. 10, pp 1274–1290, 1986.

[3] Adams, W.P., and Sherali, H.D., "Linearization Strategies for a Class of Zero-One Mixed Integer Programming Problems," *Operations Research*, Vol. 38, No. 2, pp 217–226, 1990.

[4] Adams, W.P., and Sherali, H.D., "Mixed-integer Bilinear Programming Problems," *Mathematical Programming*, Vol 59, pp 279–305.

[5] Ahuja, R.K., Magnanti, T.L., and Orlin, J.B., *Network Flows. Theory, Algorithms and Applications*, Prentice Hall, Englewood Cliffs, 1993.

[6] Bazaraa, M.S., Sherali, H.D., and Shetty, C.M., *Nonlinear Programming: Theory and Algorithms*, John Wiley & Sons, Inc., Hoboken, New Jersey, 2006.

[7] Billionnet, A. and Elloumi, S., "Best Reduction of the Quadratic Semi-Assignment Problem," *Discrete Applied Mathematics*, Vol. 109, Issue 3, pp 197–213, 2001.

[8] Chartrand, G., and Lesniak, L., *Graphs & Digraphs*, Chapman and Hall/CRC, Boca Raton, FL, 2004.

[9] Chretienne, P., "A Polynomial Algorithm to Optimally Schedule Tasks on a Virtual Distributed System under Tree-Like Precedence Constraints," *European Journal of Operations Research*, Vol. 43, Issue 2, pp 225–230, 1989.

[10] Fisher, M.L., "The Lagrangian Relaxation Method for Solving Integer Programming Problems," *Management Science*, Vol. 50, No 12, pp 1861–1871.

[11] Gallo, G. and Simeone, B., "Optimal Groupings of Researchers into Departments," *Ricera Operativa*, Vol. 57, pp 45–69, 1991.

[12] Geoffrion, A.M., "Lagrangean Relaxation for Integer Programming," *Mathematical Programming Study 2*, pp 82–114, 1974.

[13] Held, M., Wolfe, P., and Crowder, H.D., "Validation of Subgradient Optimization," *Mathimatical Programming 6*, pp 62–88, 1974.

[14] Malucelli, F., "A Polynomial Solvable Class of Quadratic Semi-Assignment Problems," *European Journal of Operational Research*, Vol. 91, Issue 3, pp 619–622, 1996.

[15] Malucelli, F. and Pretolani, D., "Lower Bounds for the Quadratic Semi-Assignment Problem," *European Journal of Operations Research*, Vol. 83, Issue 2, pp 365–375, 1995.

[16] Malucelli, F. and Pretolani, D., "Quadratic Semi-Assignment Problem on Structured Graphs," *Ricerca Operativa*, Vol. 69, pp 57-78, 1994.

[17] Pessoa, A.A., Hahn, P.M., Guignard, M., and Zhu, Y.R., "Algorithms for the Generalized Quadratic Assignment Problem Combining Lagrangean Decomposition and the Reformulation-Linearization Technique," *European Journal of Operations Research*, Vol. 206, No. 1, pp 54–63, 2010.

[18] Saito, H., "The Symmetric Quadratic Semi-Assignment Polytope," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E89-A, Issue 5, pp 1227–1232, 2006.

[19] Saito, H., Fujie, T., Matsui, T., and Matuura, S., "A Study of the Quadratic Semi-Assignment Polytope," *Discrete Optimization*, Vol. 6, Issue 1, pp 37–50, 2009.

[20] Sahni, S. and Gonzalez, T., "P-Complete Approximation Problems," *Journal of the ACM*, Vol. 23, Issue 3, pp 555–565, 1976.

[21] Schüle, I., Ewe, H., and Küfer, K., "Finding Tight RLT Formulations for Quadratic Semi-Assignment Problems," In Proceedings of CTW, pp 109–112, 2009.

[22] Sherali, H.D., and Adams, W.P., "A Hierarchy of Relaxations Between the Continuous and Convex Hull Representations for Zero-One Programming Problems," *SIAM Journal of Discrete Mathematics*, Vol. 3, No. 3, pp 411–430, 1990.

[23] Sherali, H.D., and Adams, W.P., "A Hierarchy of Relaxations and Convex Hull Characterizations for Mixed-Integer Zero-One Programming Problems," *Discrete Applied Mathematics*, Vol. 52, No. 1, pp 83–106, 1994 (manuscript 1989).

[24] Simeone, B., "An Asymptotically Exact Polynomial Algorithm for Equipartition Problems," *Discrete Applied Mathematics*, Vol. 14, No. 3, pp 283–293, 1985.

[25] Simeone, B., "Combinatorial Optimization," *Lecture Notes in Mathematics*, Vol. 1403, Springer-Verlag, 1986.

[26] Wallis, W.D., *A Beginner's Guide to Graph Theory*, 2nd Edition, Birkhäuser, Boston, 2007.

Chapter 3

Modeling Disjunctions of Polytopes with Application to Piecewise Linear Functions

3.1 Introduction

The feasible regions to a variety of mixed-discrete optimization problems can be modeled as disjunctions of polytopes. Consider a set X in variables $\boldsymbol{x} \in \mathbb{R}^q$ defined in terms of n disjunctions as

$$X \equiv \bigcup_{k=1}^{n} P_k, \tag{3.1}$$

where the n polytopes P_k are given by

$$P_k = \left\{\boldsymbol{x} : A^k\boldsymbol{x} = \boldsymbol{b}^k,\ \boldsymbol{x} \geq \boldsymbol{0}\right\}\ \forall\ k = 1, \ldots, n. \tag{3.2}$$

Here, A^k and $\boldsymbol{b}^k$ are appropriately-dimensioned matrices and vectors, respectively, and we assume without loss of generality that the restrictions defining the sets P_k consist of equality constraints in nonnegative variables. Then $\boldsymbol{x}$ must lie in at least one of the n polytopes P_k.

The n disjunctions defining X can be modeled using 0-1 variables within a higher-dimensional space as follows. For each $k \in \{1, \ldots, n\}$, multiply every constraint defining the polytope P_k by a binary variable λ_k, and substitute $\boldsymbol{w}_k = \boldsymbol{x}\lambda_k$. Then set $\boldsymbol{x} = \sum_{k=1}^n \boldsymbol{w}_k$ and $1 = \sum_{k=1}^n \lambda_k$. The following expression of X results, where $\boldsymbol{\lambda}$ represents the vector $(\lambda_1, \lambda_2, \ldots, \lambda_n)^T$ and $\boldsymbol{w}$ represents the vector $(\boldsymbol{w}_1^T, \boldsymbol{w}_2^T, \ldots, \boldsymbol{w}_n^T)^T$.

$$
\begin{aligned}
X' = \Bigg\{ (\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{w}) : \; & A^k \boldsymbol{w}_k = \boldsymbol{b}^k \lambda_k \;\forall\, k = 1, \ldots, n, \\
& \boldsymbol{w}_k \geq \boldsymbol{0} \;\forall\, k = 1, \ldots, n, \\
& \boldsymbol{x} = \sum_{k=1}^{n} \boldsymbol{w}_k, \\
& 1 = \sum_{k=1}^{n} \lambda_k, \\
& \lambda_k \text{ binary } \forall\, k = 1, \ldots, n \Bigg\}
\end{aligned}
\tag{3.3}
$$

Clearly, at every feasible solution, exactly one λ_k, say λ_p, will equal 1 and the remaining λ_k will equal 0, with $\boldsymbol{x}$ then given by $\boldsymbol{x} = \boldsymbol{w}_p$.

Denote the relaxed version of the set X' obtained by replacing the binary restrictions on λ_k with variable nonnegativity for all $k = 1, \ldots, n$, as the set $\overline{X}'$. A result of [2, 3] is that

$$\text{conv}(X) = \text{Proj}_{\boldsymbol{x}}(\overline{X}'),$$

where conv($\bullet$) represents the convex hull of $\bullet$ and

$$\text{Proj}_{\boldsymbol{x}}(\overline{X}') = \left\{ \boldsymbol{x} : \text{there exists a } (\boldsymbol{\lambda}, \boldsymbol{w}) \text{ so that } (\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{w}) \in \overline{X}' \right\}$$

represents the projection of the set X onto the space of the variables $\boldsymbol{x}$. This convex hull result of [2, 3] does not hold true using the relaxed set $\overline{X}'$ if X is contained within a larger optimization problem, so that the binary restrictions on λ_k for $k = 1, \ldots, n$ must be enforced in this more general case.

In this chapter, we begin by employing a result of [1] to reduce the number of binary variables within the set X' from n to $\lceil \log_2 n \rceil$. The method operates by defining $\lceil \log_2 n \rceil$ new binary variables and $\lceil \log_2 n \rceil$ linear equations that allow us to relax $\boldsymbol{\lambda}$ to be nonnegative. Section 2 reviews the approach. Section 3, which includes the main theoretical contributions of this chapter, provides methods for reducing problem size. To begin, the continuity of $\boldsymbol{\lambda}$ allows us to substitute these variables from the problem. Then, depending on the structure of the polytopes P_k, projection operations are characterized for further reductions. Interestingly, for the polytopes considered, these projections do not increase the number of constraints, so that smaller forms are obtained. The special polytopes consist of knapsack restrictions taking the form of SOS-1 and SOS-2 type restrictions. Section 4 uses the known relationships between SOS-2 restrictions and piecewise-linear functions to explain two alternate representations of the latter, both of which use logarithmic numbers of binary variables. The first form is a direct consequence of one of our models, though ours requires roughly half the number of constraints in the same number of variables. The second form is a classical approach for piecewise-linear functions, cast in a different variable space from the first. Our projections can thus be viewed as a theoretical linkage joining together two otherwise unrelated forms. Section 5 provides computational experience to compare the relative merits of two of our representations with these alternate two. Finally, Section 6 provides a summary of results and concluding remarks.

3.2 Reduction of Binary Variables

A method of [1] can reduce the number of binary variables within X'. Given a collection of n binary variables that is restricted to sum to 1, the paper [1] provides a method for rewriting the variables as continuous by introducing a logarithmic number of new binary variables and constraints. Specifically, given a set of n variables λ_k, $k = 1, \ldots, n$, that is restricted to satisfy

$$\sum_{k=1}^{n} \lambda_k = 1, \ \lambda_k \text{ binary for } k = 1, \ldots, n, \tag{3.4}$$

the binary restrictions on the variables λ_k can be relaxed to nonnegativity as follows. First, form n binary vectors $\boldsymbol{v}_k \in \mathbb{R}^{\lceil \log_2 n \rceil}$, $k = 1, \ldots, n$, as

$$\begin{aligned} \boldsymbol{v}_k \in \mathbb{R}^{\lceil \log_2 n \rceil} = \{ &\text{the binary expansion of the number } k-1, \text{ where position} \\ &i \text{ corresponds to the value } 2^{i-1}\} \text{ for all } k = 1, \ldots, n. \end{aligned} \tag{3.5}$$

Then construct a new vector of binary variables $\boldsymbol{u} \in \mathbb{R}^{\lceil \log_2 n \rceil}$ to associate in a one-to-one fashion with the vectors $\boldsymbol{v}_k$, and enforce

$$\sum_{k=1}^{n} \lambda_k = 1,\ \lambda_k \geq 0 \text{ for } k = 1, \ldots, n,\ \sum_{k=1}^{n} \lambda_k \boldsymbol{v}_k = \boldsymbol{u},\ \boldsymbol{u} \text{ binary.} \tag{3.6}$$

System (3.6) has $\lceil \log_2 n \rceil$ binary variables as opposed to the n variables of (3.4). Moreover, the continuous relaxation of (3.6) obtained by relaxing the $\boldsymbol{u}$ binary restrictions to $\mathbf{0} \leq \boldsymbol{u} \leq \mathbf{1}$ has all binary extreme points. This reduction of binary variables was achieved at the expense of an additional $\lceil \log_2 n \rceil$ equations which enforce that, given any feasible $(\boldsymbol{\lambda}, \boldsymbol{u})$ with $\boldsymbol{u}$ binary, the vector $\boldsymbol{\lambda}$ must also be binary. (The vectors $\boldsymbol{v}_k$ need not be defined as in (3.5), but can instead comprise any distinct set of binary vectors in $\mathbb{R}^{\lceil \log_2 n \rceil}$.)

Example 1

Consider an instance of (3.4) having $n = 10$ so that

$$\sum_{k=1}^{10} \lambda_k = 1,\ \lambda_k \text{ binary for } k = 1, \ldots, 10.$$

Then the representation (3.6) uses $4 = \lceil \log_2(10) \rceil$ binary variables $\boldsymbol{u}$ and ten vectors $\boldsymbol{v}_k \in \mathbb{R}^4$ of the form (3.5) to relax the binary restrictions on λ_k to nonnegativity. The resulting system is expressed in matrix form as follows.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \\ \lambda_6 \\ \lambda_7 \\ \lambda_8 \\ \lambda_9 \\ \lambda_{10} \end{bmatrix} = \begin{bmatrix} 1 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix},\ \boldsymbol{\lambda} \geq \mathbf{0},\ \boldsymbol{u} \text{ binary}$$

The paper [1] uses (3.6) as the foundation for linearizing products of functions of discrete variables. Here, we employ (3.6) within a disjunctive programming context. Specifically, we apply this idea to the set X' to obtain the form below.

$$X'' = \left\{ \begin{aligned} (\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{w}, \boldsymbol{u}) : A^k \boldsymbol{w}_k = \boldsymbol{b}^k \lambda_k \ \forall\ k = 1, \ldots, n, \\ \boldsymbol{w}_k \geq \boldsymbol{0} \ \forall\ k = 1, \ldots, n, \\ \boldsymbol{x} = \sum_{k=1}^{n} \boldsymbol{w}_k, \\ 1 = \sum_{k=1}^{n} \lambda_k, \\ \lambda_k \geq 0 \ \forall\ k = 1, \ldots, n, \\ \sum_{k=1}^{n} \lambda_k \boldsymbol{v}_k = \boldsymbol{u},\ \boldsymbol{u} \text{ binary} \end{aligned} \right\} \tag{3.7}$$

3.3 Problem Reduction and Exploitation of Structure

The continuity of the variables $\boldsymbol{\lambda}$ in X'' allows for their elimination via substitution, a characteristic not shared by X'. In addition, special structures of the polytopes P_k can lead to simpler forms.

The variable substitutions operate as follows. For each k, select any nonzero entry of the vector $\boldsymbol{b}^k$ occuring in some row i, and express λ_k in terms of the variables $\boldsymbol{w}_k$ via equation i of $A^k \boldsymbol{w}_k = \boldsymbol{b}^k \lambda_k$. Use this representation to substitute all occurrences of λ_k from the problem, and then remove this equation. Upon performing this substitution for all k, the problem will have removed all n variables λ_k, and will have one equation found in each system $A^k \boldsymbol{w}_k = \boldsymbol{b}^k \lambda_k$, namely the associated equation i, replaced by an inequality restriction in the variables $\boldsymbol{w}_k$ only. (As discussed below, these new inequality constraints will be redundant for special A^k.)

We mention here that for any $j \in \{1, \ldots, n\}$, since P_j is a polytope, if all entries of $\boldsymbol{b}^j$ were to equal 0, then P_j would restrict $\boldsymbol{x} = \boldsymbol{0}$. If this case were to occur, then $\boldsymbol{w}_j = \boldsymbol{0}$ in X' and X'' regardless of the value of λ_j, and this variable could be removed from both sets using the substitution $\lambda_j = 1 - \sum_{k \neq j} \lambda_k$. By letting the vector $\boldsymbol{v}_j$ associated with P_j be all zeroes, the substitution is equivalent to relaxing the restriction $\sum_{k=1}^{n} \lambda_k = 1$ to $\sum_{k \neq j} \lambda_k \leq 1$. Logically, we can assume without loss of generality that at most one set P_j has all entries of $\boldsymbol{b}^j$ equal to 0.

Now, special structure can allow for a further reduction in problem size. Suppose that each set P_k of (3.2) has a single "knapsack" equality restriction defined on a subset $J_k \subseteq Q \equiv \{1,\ldots,q\}$ of the q variables $x_1,\ldots,x_q$, with positive coefficients α_{kj} and with positive righthand-side b_k so that

$$P_k = \left\{ \boldsymbol{x} : \sum_{j\in J_k} \alpha_{kj}x_j = b_k,\ x_j \geq 0\ \forall\, j \in J_k,\ x_j = 0\ \forall\, j \notin J_k \right\}\ \forall\, k = 1,\ldots,n. \tag{3.8}$$

(We assume that each index j appears in at least one set J_k since otherwise x_j must equal 0 at all solutions to (3.2).) Using these sets P_k, the variable substitutions described above reduce each system $A^k\boldsymbol{w}_k = \boldsymbol{b}^k\lambda_k$ to the inequality restriction $\sum_{j\in J_k}\frac{\alpha_{kj}}{b_k}w_{kj} \geq 0$, where $\lambda_k = \sum_{j\in J_k}\frac{\alpha_{kj}}{b_k}w_{kj}$ and where variable w_{kj} represents component j of $\boldsymbol{w}_k$ resulting from the product $\lambda_k x_j$. As all α_{kj} with $j \in J_k$ and b_k are positive, and as all the associated w_{kj} are nonnegative, each such inequality is redundant. The net result is to effectively remove all n equations $A^k\boldsymbol{w}_k = \boldsymbol{b}^k\lambda_k$ and all n nonnegativity restrictions on λ_k from X''. The below formulation results, where the variables w_{kj} for all (j,k) having $j \notin J_k$ are by definition 0 and are therefore not explicitly stated.

$$\begin{aligned} X''' = \Bigg\{ & (\boldsymbol{x},\boldsymbol{w},\boldsymbol{u}) : w_{kj} \geq 0\ \forall\, j \in J_k,\ \forall\, k = 1,\ldots,n, \\ & x_j = \sum_{k:j\in J_k} w_{kj}\ \forall\, j = 1,\ldots,n, \\ & 1 = \sum_{k=1}^{n}\left(\sum_{j\in J_k}\frac{\alpha_{kj}}{b_k}w_{kj}\right), \\ & \sum_{k=1}^{n}\left(\sum_{j\in J_k}\frac{\alpha_{kj}}{b_k}w_{kj}\right)\boldsymbol{v}_k = \boldsymbol{u},\ \boldsymbol{u} \text{ binary} \Bigg\} \end{aligned} \tag{3.9}$$

Two specific forms of the sets P_k of (3.8) are considered in the next two subsections.

3.3.1 Representation of a Discrete Variable

Consider the instance of a discrete variable x that can realize values in some finite set $\{\theta_1,\ldots,\theta_n\}$ of positive values. Then X of (3.1) is defined in terms of the single variable x_1 so that $q = 1$ and $X = \cup_{k=1}^{n} P_k$, where each set P_k takes the form of (3.8) with $P_k = \{x_1 : x_1 = \theta_k\}$ giving us, for each k, that $J_k = \{1\}$, $\alpha_{k1} = 1$, and $\boldsymbol{b}^k = b_k = \theta_k$. The associated restrictions $A^k\boldsymbol{w}_k = \boldsymbol{b}^k\lambda_k$ of X' and X'' become $w_{k1} = \theta_k\lambda_k$. Here, the inequality $x_1 \geq 0$ is not found within any set P_k so no

$w_{k1} \geq 0$ inequalities result in (3.9). But for each $k = 1, \ldots, n$, substituting $\lambda_k = \frac{w_{k1}}{\theta_k}$ into $\lambda_k \geq 0$ gives us that $\frac{w_{k1}}{\theta_k} \geq 0$. Then X''' of (3.9) becomes the following, denoted by DV to identify the modeling of a discrete variable.

$$
\begin{aligned}
DV = \Bigg\{ (x_1, \boldsymbol{w}, \boldsymbol{u}) : \frac{w_{k1}}{\theta_k} &\geq 0 \; \forall \, k = 1, \ldots, n, \\
x_1 &= \sum_{k=1}^{n} w_{k1}, \\
1 &= \sum_{k=1}^{n} \left(\frac{w_{k1}}{\theta_k} \right), \\
\sum_{k=1}^{n} \left(\frac{w_{k1}}{\theta_k} \right) \boldsymbol{v}_k &= \boldsymbol{u}, \; \boldsymbol{u} \text{ binary} \Bigg\}
\end{aligned}
\tag{3.10}
$$

For this special case in which each set P_k defines a single point, we have the option to remove the variables w_{k1} instead of λ_k from X'' using the substitution $w_{k1} = \theta_k \lambda_k$ for each k. The resulting form is given below as DV'.

$$
\begin{aligned}
DV' = \Bigg\{ (x_1, \boldsymbol{\lambda}, \boldsymbol{u}) : x_1 &= \sum_{k=1}^{n} \theta_k \lambda_k, \\
1 &= \sum_{k=1}^{n} \lambda_k, \\
\lambda_k &\geq 0 \; \forall \, k = 1, \ldots, n, \\
\sum_{k=1}^{n} \lambda_k \boldsymbol{v}_k &= \boldsymbol{u}, \; \boldsymbol{u} \text{ binary} \Bigg\}
\end{aligned}
\tag{3.11}
$$

Observe that (3.10) and (3.11) are equivalent via a scaling of variables. For any $(\hat{x}_1, \hat{\boldsymbol{w}}, \hat{\boldsymbol{u}})$ feasible to (3.10), the point $(\hat{x}_1, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{u}})$ is feasible to (3.11) with $\hat{\lambda}_k = \frac{\hat{w}_{k1}}{\theta_k}$ for all $k = 1, \ldots, n$. Similarly, for any $(\hat{x}_1, \hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{u}})$ feasible to (3.11), the point $(\hat{x}_1, \hat{\boldsymbol{w}}, \hat{\boldsymbol{u}})$ is feasible to (3.10) with $\hat{w}_{k1} = \theta_k \hat{\lambda}_k$ for all $k = 1, \ldots, n$. A notable difference between (3.10) and (3.11), however, is that the derivation of the latter does not require the scalars b_k of (3.8) to be nonnegative. As a result, the set of permissible values $\theta_1, \ldots, \theta_n$ for x_1 can be arbitrary within (3.11). Formulation DV' is as found in [1].

This convex hull argument extends to any collection of polytopes P_k consisting of single points. Given that each P_k can be expressed as $P_k = \{\boldsymbol{x} : \boldsymbol{x} = \boldsymbol{b}^k\}$ where $\boldsymbol{x}$ is not restricted to be nonnegative, the set X'' of (3.7) has $\boldsymbol{w}_k = \boldsymbol{b}^k \lambda_k$ for all k, with $\boldsymbol{w}_k$ unrestricted. Then $\boldsymbol{w}_k$ can be substituted from the problem to obtain the set

$$\left\{(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}) : \boldsymbol{x} = \sum_{k=1}^{n} \boldsymbol{b}^k \lambda_k,\ 1 = \sum_{k=1}^{n} \lambda_k,\ \lambda_k \geq 0\ \forall\, k = 1, \ldots, n,\ \sum_{k=1}^{n} \lambda_k \boldsymbol{v}_k = \boldsymbol{u},\ \boldsymbol{u} \text{ binary}\right\},$$

which reduces to DV' when $\boldsymbol{x} = x_1$ and $\boldsymbol{b}^k = \theta_k$.

3.3.2 Representation of SOS-d Restrictions

Given a set of q nonnegative, continuous variables $x_1, \ldots, x_q$ that is restricted to sum to one, an SOS-d restriction enforces that at most d variables can be positive, and that these variables must be selected from a consecutive subset. Such a restriction can be naturally modeled by (3.1) using (3.8). The $n = q - d + 1$ polytopes P_k of (3.8) have the sets J_k defined by

$$J_k = \{k, \ldots, k + d - 1\} \text{ for each } k = 1, \ldots, q - d + 1, \tag{3.12}$$

with $\alpha_{kj} = 1$ for all $k = 1, \ldots, n$ and $j \in J_k$, and with all $b_k = 1$. Then the sets (3.8) take the form

$$P_k = \left\{\boldsymbol{x} : \sum_{j \in J_k} x_j = 1,\ x_j \geq 0\ \forall\, j \in J_k,\ x_j = 0\ \forall\, j \notin J_k\right\}\ \forall\, k = 1, \ldots, q - d + 1, \tag{3.13}$$

with the sets J_k as given in (3.12) so that X''' of (3.9) becomes the set X_d below, where $\boldsymbol{w}$ denotes the collection of $d(q - d + 1)$ variables w_{kj} such that $j \in J_k$ and $k \in \{1, \ldots, q - d + 1\}$.

$$\begin{aligned} X_d = \Bigg\{ & (\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{u}) : w_{kj} \geq 0\ \forall\, j = k, \ldots, d + k - 1,\ \forall\, k = 1, \ldots, q - d + 1, \\ & x_j = \sum_{k=\max\{1, j-d+1\}}^{\min\{j, q-d+1\}} w_{kj}\ \forall\, j = 1, \ldots, q, \\ & 1 = \sum_{k=1}^{q-d+1} \left(\sum_{j=k}^{k+d-1} w_{kj} \right), \\ & \sum_{k=1}^{q-d+1} \left(\sum_{j=k}^{k+d-1} w_{kj} \right) \boldsymbol{v}_k = \boldsymbol{u},\ \boldsymbol{u} \text{ binary} \Bigg\} \end{aligned} \tag{3.14}$$

Observe how (3.14) simplifies when $d = 1$ and $d = 2$ to correspond to SOS-1 and SOS-2 type restrictions respectively. When $d = 1$, we obtain $n = q$ and

$$X_1 = \left\{ (\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{u}) : w_{kk} \geq 0 \; \forall \; k = 1, \ldots q, \right.$$

$$x_j = w_{jj} \; \forall \; j = 1, \ldots, q,$$

$$1 = \sum_{k=1}^{q} w_{kk},$$

$$\left. \sum_{k=1}^{q} w_{kk} \boldsymbol{v}_k = \boldsymbol{u}, \; \boldsymbol{u} \text{ binary} \right\} \tag{3.15}$$

Then we can remove $\boldsymbol{w}$ from X_1 using the identity $x_j = w_{jj}$ for all j to obtain the projection of X_1 onto the space of the variables $(\boldsymbol{x}, \boldsymbol{u})$ as

$$\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_1) = \left\{ (\boldsymbol{x}, \boldsymbol{u}) : \sum_{k=1}^{q} x_k = 1, \; x_k \geq 0 \text{ for } k = 1, \ldots, q, \; \sum_{k=1}^{q} x_k \boldsymbol{v}_k = \boldsymbol{u}, \; \boldsymbol{u} \text{ binary} \right\},$$

which is of the form (3.6) with $n = q$.

When $d = 2$, we have (3.14) simplifying to

$$X_2 = \left\{ (\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{u}) : w_{kk}, w_{k(k+1)} \geq 0 \; \forall \; k = 2, \ldots q - 1, \right. \tag{3.16}$$

$$x_1 = w_{11}, \tag{3.17}$$

$$x_j = w_{(j-1)j} + w_{jj} \; \forall \; j = 1, \ldots q - 1, \tag{3.18}$$

$$x_q = w_{(q-1)q}, \tag{3.19}$$

$$1 = \sum_{k=1}^{q-1} \left(w_{kk} + w_{k(k+1)} \right), \tag{3.20}$$

$$\left. \sum_{k=1}^{q-1} \left(w_{kk} + w_{k(k+1)} \right) \boldsymbol{v}_k = \boldsymbol{u}, \; \boldsymbol{u} \text{ binary} \right\}. \tag{3.21}$$

3.3.3 Alternate Reductions for SOS-2 Restrictions

The SOS-2 restrictions have a special structure that allows us to alternately express the corresponding set X'' of (3.7) in terms of only the variables $(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u})$ or $(\boldsymbol{x}, \boldsymbol{u})$, as opposed to the variables $(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{u})$ of X_2 in (3.16)–(3.21). To see this, observe that the constraints $\boldsymbol{x} = \sum_{k=1}^{n} \boldsymbol{w}_k$ and $A^k \boldsymbol{w}_k = \boldsymbol{b}^k \lambda_k \; \forall \; k = 1, \ldots, q - 1$ of X'' found in (3.7) take the form of (3.17)–(3.19) and $\lambda_k = w_{kk} + w_{k(k+1)} \; \forall \; k = 1, \ldots, q - 1$, respectively. Since $\boldsymbol{w}_k \geq \boldsymbol{0} \; \forall \; k = 1, \ldots, q - 1$, the $\boldsymbol{\lambda} \geq \boldsymbol{0}$

restrictions are implied and can be omitted. Now, let us define a new variable $w_0 = 0$ and modify the first constraint to $x_1 = w_0 + w_{11}$ so that $\boldsymbol{w}$ becomes a vector of size $2q - 1$. Then these $2q$ restrictions define a nonsingular linear transformation between the $2q - 1$ variables $(\boldsymbol{x}, \boldsymbol{\lambda})$ and the resulting $2q - 1$ variables $\boldsymbol{w}$. Specifically, these $2q$ constraints can be represented in matrix form as

$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{bmatrix} = Q\boldsymbol{w},\ w_0 = 0, \tag{3.22}$$

where $\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{bmatrix} = (\boldsymbol{x}^T, \boldsymbol{\lambda}^T)^T$ and $\boldsymbol{w}$ are column vectors in $\mathbb{R}^{2q-1}$ with $\boldsymbol{w}$ having w_0 in position 1 and having w_{kj} in position $k + j$, and where Q is a $(2q - 1) \times (2q - 1)$ matrix.

Suppose that we wish to reorder the rows of Q so that that row associated with each variable x_i appears in row $2i - 1$ and that row associated with each variable λ_i appears in row $2i$. In this manner, that variable in position i *after* the permutation originated in position $\frac{i+1}{2}$ when i is odd, and originated in position $\frac{i}{2} + q$ when i is even, for all i. This operation can be expressed in matrix form by defining a $(2q - 1) \times (2q - 1)$ permutation matrix P whose $(i, j)^{th}$ element, P_{ij}, is given by

$$P_{ij} = \begin{cases} 1 & \text{if } j = \frac{i+1}{2} \text{ (so that } i \text{ is odd)} \\ 1 & \text{if } j = \frac{i}{2} + q \text{ (so that } i \text{ is even)} \\ 0 & \text{otherwise} \end{cases} \quad \forall\, i = 1, \ldots, 2q - 1,\ j = 1, \ldots, 2q - 1, \tag{3.23}$$

and computing PQ. This matrix PQ has Jordan block form with ones along the diagonal so that $(PQ)^{-1}$ exists and is upper triangular with

$$(PQ)^{-1}_{ij} = \begin{cases} (-1)^{i+j} & \text{if } j \geq i \\ 0 & \text{otherwise} \end{cases} \quad \forall\, i = 1, \ldots, 2q - 1,\ j = 1, \ldots, 2q - 1. \tag{3.24}$$

Now, left-multiply the matrix system of (3.22) by the $(2q - 1) \times (2q - 1)$ matrix $P^T(PQ)^{-1}P$ to obtain

$$P^T(PQ)^{-1}P \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{bmatrix} = P^T(PQ)^{-1}PQ\boldsymbol{w},\ w_0 = 0. \tag{3.25}$$

System (3.25) simplifies to

$$\left[\begin{array}{ccccc|cccc} 1 & 1 & \cdots & \cdots & 1 & -1 & -1 & \cdots & -1 \\ & 1 & \cdots & \cdots & 1 & & -1 & \cdots & -1 \\ & & \ddots & \ddots & \vdots & & & \ddots & \vdots \\ & & & \ddots & \vdots & & & & -1 \\ & & & & 1 & & & & \\ \hline & -1 & \cdots & \cdots & -1 & 1 & \cdots & \cdots & 1 \\ & & \ddots & \ddots & \vdots & & \ddots & \ddots & \vdots \\ & & & \ddots & \vdots & & & \ddots & \vdots \\ & & & & -1 & & & & 1 \end{array}\right] \left[\begin{array}{c} x_1 \\ \vdots \\ x_q \\ \hline \lambda_1 \\ \vdots \\ \lambda_{q-1} \end{array}\right] = P^T \left[\begin{array}{c} w_0 \\ w_{11} \\ \vdots \\ w_{(q-1)q} \end{array}\right], \; w_0 = 0, \tag{3.26}$$

where the $(2q-1) \times (2q-1)$ matrix follows from (3.24) and the property that the permutation matrix P of (3.23) relocates each entry (i,j) of $(PQ)^{-1}$ into entry $(p(i), p(j))$ of $P^T(PQ)^{-1}P$, where $p(\alpha) = \frac{\alpha+1}{2}$ if α is odd and $p(\alpha) = \frac{\alpha}{2} + q$ if α is even, for all $\alpha = 1, \ldots, 2q-1$.

The nonnegativity of $\boldsymbol{w}$ in X'' of (3.7) allows us to rewrite X_2 in terms of the variables $(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u})$ as below, where we use X_2' to denote the modified version of X_2 in a different variable space. Here, we set $w_0 = 0$ and accordingly remove this variable from the problem. The first equation is written separately since it remains an equality.

$$X_2' = \left\{ (\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}) : \sum_{k=1}^{q} x_k - \sum_{k=1}^{q-1} \lambda_k = 0, \; \sum_{k=1}^{q} x_k = 1, \; \sum_{k=1}^{q-1} \lambda_k \boldsymbol{v}_k = \boldsymbol{u}, \; \boldsymbol{u} \text{ binary}, \right.$$
$$\left. \left[\begin{array}{cccc|cccc} 1 & \cdots & \cdots & 1 & & -1 & \cdots & -1 \\ & \ddots & \ddots & \vdots & & & \ddots & \vdots \\ & & \ddots & \vdots & & & & -1 \\ & & & 1 & & & & \\ \hline -1 & \cdots & \cdots & -1 & 1 & \cdots & \cdots & 1 \\ & \ddots & \ddots & \vdots & & \ddots & \ddots & \vdots \\ & & \ddots & \vdots & & & \ddots & \vdots \\ & & & -1 & & & & 1 \end{array}\right] \left[\begin{array}{c} x_2 \\ \vdots \\ x_q \\ \hline \lambda_1 \\ \vdots \\ \lambda_{q-1} \end{array}\right] \geq \boldsymbol{0} \right\} \tag{3.27}$$

Example 2

Consider the case where X in (3.1) has $q = 4$ variables so that $\boldsymbol{x}^T = (x_1, x_2, x_3, x_4)$, and is defined in terms of sets P_k of the form (3.13) having $d = 2$ so that the SOS-2 restrictions look as follows.

$$\begin{aligned} P_1 &= \{\boldsymbol{x} : x_1 + x_2 = 1, x_3 = x_4 = 0\} \\ P_2 &= \{\boldsymbol{x} : x_2 + x_3 = 1, x_1 = x_4 = 0\} \\ P_3 &= \{\boldsymbol{x} : x_3 + x_4 = 1, x_1 = x_2 = 0\} \end{aligned} \tag{3.28}$$

The set X'' of (3.7) takes the form below, where we have included the variable w_0 set to 0 as in (3.22), and where $\boldsymbol{v}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $\boldsymbol{v}_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, and $\boldsymbol{v}_3 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

$$X'' = \Bigg\{ (\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{w}, \boldsymbol{u}) : \; w_0, w_{11}, w_{12}, w_{22}, w_{23}, w_{33}, w_{34}, \lambda_1, \lambda_2, \lambda_3 \geq 0,$$

$$w_0 = 0, \; u_1, u_2 \text{ binary},$$

$$\left.\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} w_0 \\ w_{11} \\ w_{12} \\ w_{22} \\ w_{23} \\ w_{33} \\ w_{34} \end{bmatrix}, \begin{bmatrix} 1 \\ u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} \right\}$$

Here, the first matrix equation looks as $\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{bmatrix} = Q\boldsymbol{w}$ of (3.22).

The set X_2 of (3.16)–(3.21) is given below, and is obtained from the above system by removing the implied inequalities $\lambda_1, \lambda_2, \lambda_3 \geq 0$ and the variable w_0 fixed to 0.

$$X_2 = \left\{ (\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{u}) : w_{11}, w_{12}, w_{22}, w_{23}, w_{33}, w_{34} \geq 0,\ u_1, u_2 \text{ binary}, \right.$$

$$\left. \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ 1 \\ u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} w_{11} \\ w_{12} \\ w_{22} \\ w_{23} \\ w_{33} \\ w_{34} \end{bmatrix} \right\}$$

The permutation matrix P of (3.23) and the matrix $P^T(PQ)^{-1}P$ of (3.24) have

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \text{ and } P^T(PQ)^{-1}P = \left[\begin{array}{cccc|ccc} 1 & 1 & 1 & 1 & -1 & -1 & -1 \\ 0 & 1 & 1 & 1 & 0 & -1 & -1 \\ 0 & 0 & 1 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & -1 & -1 & -1 & 1 & 1 & 1 \\ 0 & 0 & -1 & -1 & 0 & 1 & 1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 \end{array}\right].$$

Then X_2' of (3.27) is given below, where again $\boldsymbol{v}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $\boldsymbol{v}_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, and $\boldsymbol{v}_3 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

$$\begin{aligned} X_2' = \{(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}) : {} & \lambda_2 + \lambda_3 = u_1,\ \lambda_3 = u_2,\ u_1, u_2 \text{ binary}, \\ & 0 \leq x_4 \leq \lambda_3 \leq x_3 + x_4 \leq \lambda_2 + \lambda_3 \leq x_2 + x_3 + x_4 \\ & \leq \lambda_1 + \lambda_2 + \lambda_3 = x_1 + x_2 + x_3 + x_4 = 1\} \end{aligned}$$

Now, returning to (3.27), suppose we wish to project the set X_2' onto the space of the variables $(\boldsymbol{x}, \boldsymbol{u})$. The task can be accomplished by computing all the extreme directions of the projection cone

$$\left[\begin{array}{ccc|ccc|ccc} -1 & & & 1 & & & v_{11} & \cdots & v_{\lceil\log_2(q-1)\rceil,1} \\ \vdots & \ddots & & \vdots & \ddots & & \vdots & & \vdots \\ -1 & \cdots & -1 & 1 & \cdots & 1 & v_{1,(q-1)} & \cdots & v_{\lceil\log_2(q-1)\rceil,(q-1)} \end{array}\right]\boldsymbol{\pi} = \mathbf{0},\ \pi_j \geq 0\ \forall\ j = 2, \ldots 2q-2, \tag{3.29}$$

where $\boldsymbol{\pi} \in \mathbb{R}^{2q-2+\lceil\log_2(q-1)\rceil}$, and to generate the facets using these directions. Here, the first two sets of columns are the transpose of the coefficient matrix on the variables $\boldsymbol{\lambda}$ in (3.26), less the null row q, while the last set of columns represents the transpose of the $(\lceil\log_2(q-1)\rceil) \times (q-1)$ coefficient matrix on the variables $\boldsymbol{\lambda}$ found in equations $\sum_{k=1}^{q-1} \lambda_k \boldsymbol{v}_k = \boldsymbol{u}$ of (3.27). The notation v_{ik} is used to represent the binary entry in position i of $\boldsymbol{v}_k$.

Up to this point, the vectors $\boldsymbol{v}_k$ are any set of distinct binary vectors. It is necessary to put some restrictions on these vectors before characterizing the extreme directions of (3.29). The following definition describes a restriction on a general set of vectors $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_p \in \mathbb{R}^m$.

Definition

A set of vectors $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_p \in \mathbb{R}^m$ is in a *compatible order* if for any two adjacent vectors, say $\boldsymbol{\beta}_j$ and $\boldsymbol{\beta}_{j+1}$ for any $j = 1, \ldots, p-1$, then $\boldsymbol{\beta}_j$ and $\boldsymbol{\beta}_{j+1}$ differ by at most one component.

Note, for the special case when the set of binary vectors, $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{q-1} \in \mathbb{R}^{\lceil\log_2(q-1)\rceil}$, are distinct, a compatible order corresponds to a Hamiltonian path on the $\lceil\log_2(q-1)\rceil$-dimensional hypercube. The following lemma gives a consequence of a set of vectors that are in a compatible order.

Lemma

Let a set of vectors $\boldsymbol{\beta}_1 \ldots \boldsymbol{\beta}_p \in \mathbb{R}^m$ be in a compatible order with S defined as $S \equiv \{1, \ldots, m\}$. For any sets $S^1, S^2 \subseteq S$, with $S^1 \bigcap S^2 = \emptyset$, and the given constants $c_1, \ldots, c_m \geq 0$, then the following equation

$$\begin{aligned} &\sum_{i \in S^1} c_i \max\{0, \beta_{ij} - \beta_{i(j+1)}\} + \sum_{i \in S^2} c_i \max\{0, \beta_{i(j+1)} - \beta_{ij}\} \\ &\qquad = \max\left\{0, \sum_{i \in S^1} c_i \left(\beta_{ij} - \beta_{i(j+1)}\right) + \sum_{i \in S^2} c_i \left(\beta_{i(j+1)} - \beta_{ij}\right)\right\} \end{aligned} \tag{3.30}$$

is true for all $j = 1, \ldots, p-1$ with β_{ij} being the i^{th} component of the vector $\boldsymbol{\beta}_j$.

Proof

Let a set of vectors $\boldsymbol{\beta}_1 \ldots \boldsymbol{\beta}_p \in \mathbb{R}^m$ be in a compatible order and consider any two adjacent vectors, say $\boldsymbol{\beta}_j$ and $\boldsymbol{\beta}_{j+1}$ for any $j = 1, \ldots, p-1$. Let S^1 and S^2 be any disjoint subsets of S and let $c_1, \ldots, c_m$ be any set of nonnegative constants.

Case 1

If $\beta_{ij} = \beta_{i(j+1)}$ for all $i \in S^1 \bigcup S^2$, then (3.30) is verified below.

$$\begin{aligned}
\sum_{i \in S^1} c_i \max\{0, \beta_{ij} - \beta_{i(j+1)}\} + \sum_{i \in S^2} c_i \max\{0, \beta_{i(j+1)} - \beta_{ij}\} \\
&= \sum_{i \in S^1} c_i \max\{0, 0\} + \sum_{i \in S^2} c_i \max\{0, 0\} \\
&= \max\left\{0, \sum_{i \in S^1} 0 + \sum_{i \in S^2} 0\right\} \\
&= \max\left\{0, \sum_{i \in S^1} c_i \left(\beta_{ij} - \beta_{i(j+1)}\right) + \sum_{i \in S^2} c_i \left(\beta_{i(j+1)} - \beta_{ij}\right)\right\}
\end{aligned}$$

Case 2

There exists a $k \in S^1 \bigcup S^2$ such that $\beta_{kj} \neq \beta_{k(j+1)}$ and $\beta_{ij} = \beta_{i(j+1)}$ for all $i \in S^1 \bigcup S^2$ with $i \neq k$. Without the loss of generality, assume that $k \in S^1$ (the case with $k \in S^2$ is analogous). Thus, (3.30) is verified below.

$$
\begin{aligned}
&\sum_{i\in S^1} c_i \max\{0, \beta_{ij}-\beta_{i(j+1)}\} + \sum_{i\in S^2} c_i \max\{0, \beta_{i(j+1)}-\beta_{ij}\} \\
&\quad = \sum_{i\in S^1/k} c_i \max\{0,0\} + \max\{0, c_k(\beta_{kj}-\beta_{k(j+1)})\} + \sum_{i\in S^2} c_i \max\{0,0\} \\
&\quad = \max\{0, c_k(\beta_{kj}-\beta_{k(j+1)})\} \\
&\quad = \max\left\{0, \sum_{i\in S^1/k} 0 + c_k(\beta_{kj}-\beta_{k(j+1)}) + \sum_{i\in S^2} 0\right\} \\
&\quad = \max\left\{0, \sum_{i\in S^1/k} c_i(\beta_{ij}-\beta_{i(j+1)}) + c_k(\beta_{kj}-\beta_{k(j+1)}) + \sum_{i\in S^2} c_i(\beta_{i(j+1)}-\beta_{ij})\right\} \\
&\quad = \max\left\{0, \sum_{i\in S^1} c_i\left(\beta_{ij}-\beta_{i(j+1)}\right) + \sum_{i\in S^2} c_i\left(\beta_{i(j+1)}-\beta_{ij}\right)\right\}
\end{aligned}
$$

□

The following example shows that a set of vectors not in a compatible order does not necessarily satisfy the lemma.

<u>Example 3</u>

Let $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2 \in \mathbb{R}^2$ with $\boldsymbol{\beta}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\boldsymbol{\beta}_2 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$. The set S is defined as $S \equiv \{1,2\}$ and let $S^1 = \{1\}$ with $S^2 = \{2\}$. Assume that $c_1 = c_2 = 1$, with $j = 1$ and $j+1 = 2$, then,

$$\sum_{i\in S^1} c_i \max\{0, \beta_{ij}-\beta_{i(j+1)}\} + \sum_{i\in S^2} c_i \max\{0, \beta_{i(j+1)}-\beta_{ij}\} = \max\{0, 1-0\} + \max\{0, -1-0\} = 1,$$

and

$$\max\left\{0, \sum_{i\in S^1} c_i\left(\beta_{ij}-\beta_{i(j+1)}\right) + \sum_{i\in S^2} c_i\left(\beta_{i(j+1)}-\beta_{ij}\right)\right\} = \max\{0, (1-0)+(-1-0)\} = 0,$$

which are not equal.

The following theorem characterizes all extreme directions of (3.29) using the definition and lemma above.

Theorem 1

Given any set of distinct binary vectors $\boldsymbol{v}_1 \dots \boldsymbol{v}_{q-1} \in \mathbb{R}^{\lceil \log_2(q-1) \rceil}$ that are in a compatible order, then there are exactly $q-1+2\lceil \log_2(q-1) \rceil$ extreme directions, given as $\boldsymbol{\pi}_1^0, \dots, \boldsymbol{\pi}_{q-1}^0$, $\boldsymbol{\pi}_1^+, \dots, \boldsymbol{\pi}_{\lceil \log_2(q-1) \rceil}^+$, $\boldsymbol{\pi}_1^-, \dots, \boldsymbol{\pi}_{\lceil \log_2(q-1) \rceil}^- \in \mathbb{R}^{2q-2+\lceil \log_2(q-1) \rceil}$, of the projection cone (3.29), and these directions are

$$\boldsymbol{\pi}_i^0 = \boldsymbol{e}_i + \boldsymbol{e}_{i+q-1} \;\forall\; i = 1, \dots, q-1,$$

where $\boldsymbol{e}_i \in \mathbb{R}^{2q-2+\lceil \log_2(q-1) \rceil}$ is the unit vector of all zeros with a 1 in the i^{th} component,

$$\boldsymbol{\pi}_i^+ = \begin{bmatrix} \pi_{1i}^+ \\ \pi_{2i} \\ \vdots \\ \pi_{(q-1),i}^+ \\ \pi_{qi}^+ \\ \pi_{(q+1),i}^+ \\ \vdots \\ \pi_{(2q-2),i}^+ \\ \pi_{(2q-1),i}^+ \\ \vdots \\ \pi_{(i+2q-2),i}^+ \\ \vdots \\ \pi_{(2q-2+\lceil \log_2(q-1) \rceil),i}^+ \end{bmatrix} = \begin{bmatrix} v_{i1} \\ \max\{0, v_{i2} - v_{i1}\} \\ \vdots \\ \max\{0, v_{i,(q-1)} - v_{i,(q-2)}\} \\ 0 \\ \max\{0, v_{i1} - v_{i2}\} \\ \vdots \\ \max\{0, v_{i,(q-2)} - v_{i,(q-1)}\} \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} \;\forall\; i = 1, \dots, \lceil \log_2(q-1) \rceil,$$

and

$$\boldsymbol{\pi}_i^- = \begin{bmatrix} \pi_{1i}^- \\ \pi_{2i}^- \\ \vdots \\ \pi_{(q-1),i}^- \\ \pi_{qi}^- \\ \pi_{(q+1),i}^- \\ \vdots \\ \pi_{(2q-2),i}^- \\ \pi_{(2q-1),i}^- \\ \vdots \\ \pi_{(i+2q-2),i}^- \\ \vdots \\ \pi_{(2q-2+\lceil \log_2(q-1) \rceil),i}^- \end{bmatrix} = \begin{bmatrix} -v_{i1} \\ \max\{0, v_{i1} - v_{i2}\} \\ \vdots \\ \max\{0, v_{i,(q-2)} - v_{i,(q-1)}\} \\ 0 \\ \max\{0, v_{i2} - v_{i1}\} \\ \vdots \\ \max\{0, v_{i,(q-1)} - v_{i,(q-2)}\} \\ 0 \\ \vdots \\ -1 \\ \vdots \\ 0 \end{bmatrix} \forall\ i = 1, \ldots, \lceil \log_2(q-1) \rceil,$$

where π_{ji}^+, π_{ji}^-, and v_{ji} denote the j^{th} components of the vectors $\boldsymbol{\pi}_i^+$, $\boldsymbol{\pi}_i^-$, and $\boldsymbol{v}_i$, respectively.

Proof

First, for ease of notation, define the set $S \equiv \{2q-1, \ldots, 2q-2+\lceil \log_2(q-1) \rceil\}$. Now, consider any extreme directions $\boldsymbol{\pi}_i$ with $\pi_{ji} = 0$ for all $j \in S$. Thus, only consider the first $2q-2$ columns of the constraint matrix of (3.29). The first $q-1$ columns of the constraint matrix of (3.29) give a lower triangular matrix of all -1's while the next $q-1$ columns give a lower triangular matrix of all $+1$'s. There is one extreme direction $\boldsymbol{\pi}_i$ with $\pi_{ii} = \pi_{(i+q-1),i} = 1$ for $i = 1, \ldots, q-1$, or $\boldsymbol{\pi}_i = \boldsymbol{e}_i + \boldsymbol{e}_{i+q-1}$ which are given by $\boldsymbol{\pi}_1^0, \ldots, \boldsymbol{\pi}_{q-1}^0$.

Next, observe for any extreme direction $\hat{\boldsymbol{\pi}}$ with $\hat{\boldsymbol{\pi}} \neq \boldsymbol{\pi}_1$ that $\hat{\pi}_q$ must equal zero. To see that the remaining extreme directions have $\hat{\pi}_q = 0$, consider any $\hat{\boldsymbol{\pi}} \neq \boldsymbol{\pi}_1$ satisfying the restrictions of (3.29) with $\hat{\pi}_q = \epsilon > 0$. Then $\tilde{\boldsymbol{\pi}} = (\hat{\boldsymbol{\pi}} + \epsilon \boldsymbol{e}_1 - \epsilon \boldsymbol{e}_q)$ and $\bar{\boldsymbol{\pi}} = (\hat{\boldsymbol{\pi}} - \epsilon \boldsymbol{e}_1 + \epsilon \boldsymbol{e}_q)$ both satisfy (3.29) with $\hat{\boldsymbol{\pi}} = \frac{\tilde{\boldsymbol{\pi}}}{2} + \frac{\bar{\boldsymbol{\pi}}}{2}$.

Before identifying the last set of extreme directions, we give a general set of solutions to (3.29) and then eliminate the ones that are not extreme directions. Suppose we find a solution $\hat{\boldsymbol{\pi}}$ to (3.29) where the $\hat{\pi}_j$ are given for all $j \in S$ and at least one of these $\hat{\pi}_j$ is not equal to zero.

Rewriting (3.29) based on the set of $\hat{\pi}_j$ given results in finding a solution to

$$\left[\begin{array}{ccc|ccc} -1 & & & 1 & & \\ \vdots & \ddots & & \vdots & \ddots & \\ -1 & \cdots & -1 & 1 & \cdots & 1 \end{array}\right] \boldsymbol{\pi}' + \left[\begin{array}{c} \sum\limits_{j \in S} \hat{\pi}_j v_{j'1} \\ \vdots \\ \sum\limits_{j \in S} \hat{\pi}_j v_{j',(q-1)} \end{array}\right] = \mathbf{0},\ \pi'_j \geq 0\ \forall\, j = 2, \ldots 2q-2, \tag{3.31}$$

where $j' = j - (2q-2)$ and $\boldsymbol{\pi}'$ is the vector of the first $2q-2$ elements of the vector $\boldsymbol{\pi}$. For ease of notation, let the vector $\boldsymbol{\alpha} \in \mathbb{R}^{q-1}$ be

$$\boldsymbol{\alpha} = \left[\begin{array}{c} \alpha_1 \\ \vdots \\ \alpha_{q-1} \end{array}\right] = \left[\begin{array}{c} \sum\limits_{j \in S} \hat{\pi}_j v_{j'1} \\ \vdots \\ \sum\limits_{j \in S} \hat{\pi}_j v_{j',(q-1)} \end{array}\right],$$

and so we solve the system

$$\left[\begin{array}{ccc|ccc} -1 & & & 1 & & \\ \vdots & \ddots & & \vdots & \ddots & \\ -1 & \cdots & -1 & 1 & \cdots & 1 \end{array}\right] \boldsymbol{\pi}' = -\boldsymbol{\alpha},\ \pi'_j \geq 0\ \forall\, j = 2, \ldots 2q-2, \tag{3.32}$$

for $\boldsymbol{\pi}'$. From the system (3.32), the constraint matrix on $\boldsymbol{\pi}'$ has special structure which allows for an easy calculation of $\boldsymbol{\pi}'$, which, with the $\hat{\pi}_j$ already defined for $j \in S$, give a solution to (3.29). The goal is to define the components of $\boldsymbol{\pi}'$ in the order $\pi'_1, \pi'_q, \pi'_2, \pi'_{q+1}, \ldots, \pi'_{q-1}, \pi'_{2q-2}$, such that the components of $-\boldsymbol{\alpha}$ are satisfied in the order $-\alpha_1, -\alpha_2, \ldots, -\alpha_{q-1}$. This is done by considering the first equation, $-\pi'_1 + \pi'_q = -\alpha_1$, which is satisfied by setting $\pi'_1 = \alpha_1$, since π'_1 is unrestricted, and $\pi'_q = 0$. Now, the second equation to be satisfied is $-\pi'_1 - \pi'_2 + \pi'_q + \pi'_{q+1} = -\alpha_2$. The variables π'_1 and π'_q are already defined so rewrite the second equation as $-\pi'_2 + \pi'_{q+1} = -\alpha_2 + \pi'_1 - \pi'_q$. If the right side of this equation is negative, then set $\pi'_2 = |-\alpha_2 + \pi'_1 - \pi'_q|$ and $\pi'_{q+1} = 0$. If the right side is zero, then set $\pi'_2 = \pi'_{q+1} = 0$. If the right side is positive, then set $\pi'_{q+1} = -\alpha_2 + \pi'_1 - \pi'_q$ and $\pi'_2 = 0$. Continue in this fashion by defining the pair π'_i and π'_{i+q-1} for $i = 3, \ldots, q-1$, until all constraints of (3.32) are satisfied and all $\boldsymbol{\pi}'$ are defined. In general, this procedure can be written as

$$\left.\begin{array}{rl} \pi'_i & = \begin{cases} \left| -\alpha_i + \sum\limits_{j=1}^{i-1}(\pi'_j - \pi'_{j+q-1}) \right| & \text{if } -\alpha_i + \sum\limits_{j=1}^{i-1}(\pi'_j - \pi'_{j+q-1}) < 0 \\ 0 & \text{if } -\alpha_i + \sum\limits_{j=1}^{i-1}(\pi'_j - \pi'_{j+q-1}) = 0 \\ 0 & \text{if } -\alpha_i + \sum\limits_{j=1}^{i-1}(\pi'_j - \pi'_{j+q-1}) > 0 \end{cases} \\ \pi'_{i+q-1} & = \begin{cases} 0 & \text{if } -\alpha_i + \sum\limits_{j=1}^{i-1}(\pi'_j - \pi'_{j+q-1}) < 0 \\ 0 & \text{if } -\alpha_i + \sum\limits_{j=1}^{i-1}(\pi'_j - \pi'_{j+q-1}) = 0 \\ -\alpha_i + \sum\limits_{j=1}^{i-1}(\pi'_j - \pi'_{j+q-1}) & \text{if } -\alpha_i + \sum\limits_{j=1}^{i-1}(\pi'_j - \pi'_{j+q-1}) > 0 \end{cases} \end{array}\right\}, \ \forall\, i = 2, \ldots, q-1.$$

Alternatively, the multipliers π'_i for $i \neq 1$ and $i \neq q$ are found by rewriting the above definitions using a max function as

$$\left.\begin{array}{rl} \pi'_i & = \max\left\{0, -\left(-\alpha_i + \sum\limits_{j=1}^{i-1}(\pi'_j - \pi'_{j+q-1})\right)\right\} \\ \pi'_{i+q-1} & = \max\left\{0, -\alpha_i + \sum\limits_{j=1}^{i-1}(\pi'_j - \pi'_{j+q-1})\right\} \end{array}\right., \ \forall\, i = 2, \ldots, q-1.$$

Observe that based on this definition of $\boldsymbol{\pi}'$, at most one of each pair, π'_i and π'_{i+q-1}, is not equal to zero for $i = 2, \ldots, q-1$. In addition, note that the sum $\sum_{j=1}^{i-1}(\pi'_j - \pi'_{j+q-1})$ is equal to α_{i-1} for $i = 2, \ldots, q-1$, because from the $i-1$ row of (3.32) we have $-\sum_{j=1}^{i-1} \pi'_j + \sum_{j=1}^{i-1} \pi'_{j+q-1} = -\alpha_{i-1}$. Making this final substitution, the form of the π'_i's for $i \neq 1$ and $i \neq q$ are

$$\left.\begin{array}{rl} \pi'_i & = \max\{0, \alpha_i - \alpha_{i-1}\} \\ \pi'_{i+q-1} & = \max\{0, \alpha_{i-1} - \alpha_i\} \end{array}\right., \ \forall\, i = 2, \ldots, q-1. \tag{3.33}$$

Letting $\hat{\pi}_i = \pi'_i$ for $i = 1, \ldots, 2q-2$ and including the $\hat{\pi}_i$ already given for $i \in S$, the solution $\hat{\boldsymbol{\pi}}$ is feasible to (3.29).

For any set of defined multipliers $\hat{\pi}_j$ for $j \in S$ we have shown a way to define the $\hat{\pi}_j$ for $j = 1, \ldots, 2q-2$ such that $\hat{\boldsymbol{\pi}}$ is feasible to (3.29). Not all of the feasible directions $\hat{\boldsymbol{\pi}}$ are extreme directions (3.29), but it is easy to determine them. Of the $q-1$ extreme directions identified so far, none of them have a nonzero π_j for all $j \in S$. We look at the special cases where the extreme directions have exactly one $\pi_j \neq 0$ for some $j \in S$.

First, consider the extreme direction $\boldsymbol{\pi}_i$ with $\pi_{ji} > 0$ where $j = i + 2q - 2$. Without the loss of generality, assume $\pi_{ji} = 1$ and $\pi_{ki} = 0$ for all $k \in S$ with $k \neq j$. For this instance, the $\boldsymbol{\alpha}$ vector is

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_{q-1} \end{bmatrix} = \begin{bmatrix} \pi_{ji} v_{j'1} \\ \vdots \\ \pi_{ji} v_{j',(q-1)} \end{bmatrix} = \begin{bmatrix} v_{j'1} \\ \vdots \\ v_{j',(q-1)} \end{bmatrix},$$

where $j' = j - (2q-2)$. Using the procedure above, define π_{ji} for $j \notin S$ as $\pi_{1i} = v_{j'1}$, $\pi_{qi} = 0$, with (3.33) giving

$$\left.\begin{aligned} \pi_{ki} &= \max\left\{0, v_{j'k} - v_{j',(k-1)}\right\} \\ \pi_{(k+q-1),i} &= \max\left\{0, v_{j',(k-1)} - v_{j'k}\right\} \end{aligned}\right., \ \forall\ k = 2, \dots, q-1.$$

There are $\lceil \log_2(q-1) \rceil$ choices for $\pi_{ji} = 1$, that is, $i = 1, \dots, \lceil \log_2(q-1) \rceil$, with $j = i + 2q - 2 \in S$, which correspond to the $\lceil \log_2(q-1) \rceil$ extreme directions $\boldsymbol{\pi}_1^+, \dots, \boldsymbol{\pi}_{\lceil \log_2(q-1) \rceil}^+$.

Next, consider the extreme direction $\boldsymbol{\pi}_i$ with $\pi_{ji} < 0$ where $j = i + 2q - 2$. Without the loss of generality, assume $\pi_{ji} = -1$ and $\pi_{ki} = 0$ for all $k \in S$ with $k \neq j$. For this instance, the $\boldsymbol{\alpha}$ vector is

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_{q-1} \end{bmatrix} = \begin{bmatrix} \pi_{ji} v_{j'1} \\ \vdots \\ \pi_{ji} v_{j',(q-1)} \end{bmatrix} = \begin{bmatrix} -v_{j'1} \\ \vdots \\ -v_{j',(q-1)} \end{bmatrix},$$

where $j' = j - (2q-2)$. Using the procedure above, define π_{ji} for $j \notin S$ as $\pi_{1i} = -v_{j'1}$, $\pi_{qi} = 0$, with (3.33) giving

$$\left.\begin{aligned} \pi_{ki} &= \max\left\{0, -v_{j'k} + v_{j',(k-1)}\right\} \\ \pi_{(k+q-1),i} &= \max\left\{0, -v_{j',(k-1)} + v_{j'k}\right\} \end{aligned}\right., \ \forall\ k = 2, \dots, q-1.$$

There are $\lceil \log_2(q-1) \rceil$ choices for $\pi_{ji} = 1$, that is, $i = 1, \dots, \lceil \log_2(q-1) \rceil$, with $j = i + 2q - 2 \in S$, which correspond to the $\lceil \log_2(q-1) \rceil$ solutions $\boldsymbol{\pi}_1^-, \dots, \boldsymbol{\pi}_{\lceil \log_2(q-1) \rceil}^-$ to (3.29). All of these solutions are extreme directions because for any solution, say $\boldsymbol{\pi}_i^-$ with $\pi_{(i+2q-2),i} = -1$, given above, the only other extreme direction identified so far with a with a nonzero component there is $\boldsymbol{\pi}_i^+$ which has $\pi_{(i+2q-2),i}^+ = 1$. By definition of the other π_{ji}^- and π_{ji}^+ for $j \neq i + 2q - 2$ using the max function, it is clear that $\boldsymbol{\pi}_i^- \neq (-1)\boldsymbol{\pi}_i^+$ and so $\boldsymbol{\pi}_1^-, \dots, \boldsymbol{\pi}_{\lceil \log_2(q-1) \rceil}^-$ are extreme directions to (3.29).

Last, consider any solution $\hat{\boldsymbol{\pi}}$ to (3.29) with at least two nonzero multipliers $\hat{\pi}_j \neq 0$ for some $j \in S$. Partition the set S into S^+, S^0, and S^- defined as $S^+ \equiv \{i \in S : \hat{\pi}_i > 0\}$, $S^0 \equiv \{i \in S : \hat{\pi}_i = 0\}$, and $S^- \equiv \{i \in S : \hat{\pi}_i < 0\}$. Note, that by assumption, $|S^+| + |S^-| \geq 2$. Let $c_j = |\hat{\pi}_j|$ for all $j \in S$. The $\boldsymbol{\alpha}$ vector is

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_{q-1} \end{bmatrix} = \begin{bmatrix} \sum_{j \in S} \hat{\pi}_j v_{j'1} \\ \vdots \\ \sum_{j \in S} \hat{\pi}_j v_{j',(q-1)} \end{bmatrix}$$

where $j' = j - (2q - 2)$. Using the procedure above to find a feasible solution to (3.29), the other $\hat{\pi}_i$ for $i = 1, \ldots, 2q - 2$ are $\hat{\pi}_1 = \alpha_1 = \sum_{j \in S} \hat{\pi}_j v_{j'1}$, $\hat{\pi}_q = 0$, with

$$\left.\begin{aligned} \hat{\pi}_i &= \max\left\{0, \sum_{j \in S} \hat{\pi}_j v_{j'i} - \sum_{j \in S} \hat{\pi}_j v_{j',(i-1)}\right\} \\ \hat{\pi}_{i+q-1} &= \max\left\{0, \sum_{j \in S} \hat{\pi}_j v_{j',(i-1)} - \sum_{j \in S} \hat{\pi}_j v_{j'i}\right\} \end{aligned}\right\}, \; \forall\, i = 2, \ldots, q-1.$$

Now, we show that $\hat{\boldsymbol{\pi}}$ is a linear combination of a subset of the extreme directions $\boldsymbol{\pi}_1^+, \ldots, \boldsymbol{\pi}_{\lceil \log_2(q-1) \rceil}^+$ and $\boldsymbol{\pi}_1^-, \ldots, \boldsymbol{\pi}_{\lceil \log_2(q-1) \rceil}^-$ First, consider $\hat{\pi}_1$ which can be rewritten as

$$\begin{aligned} \hat{\pi}_1 &= \sum_{j \in S} \hat{\pi}_j v_{j'1} \\ &= \sum_{j \in S^+} \hat{\pi}_j v_{j'1} + \sum_{j \in S^0} \hat{\pi}_j v_{j'1} + \sum_{j \in S^-} \hat{\pi}_j v_{j'1} \\ &= \sum_{j \in S^+} c_j v_{j'1} + \sum_{j \in S^0} (0) v_{j'1} + \sum_{j \in S^-} -c_j v_{j'1} \\ &= \sum_{j \in S^+} c_j \pi_{1j'}^+ + \sum_{j \in S^-} c_j \pi_{1j'}^- \end{aligned}$$

where $j' = j - (2q-2)$. Next, consider $\hat{\pi}_i$ for $i = 2, \ldots, q-1$ which are defined as

$$\begin{aligned}
\hat{\pi}_i &= \max\left\{0, \sum_{j\in S} \hat{\pi}_j v_{j'i} - \sum_{j\in S} \hat{\pi}_j v_{j',(i-1)}\right\} \\
&= \max\left\{0, \sum_{j\in S^+} \hat{\pi}_j (v_{j'i} - v_{j',(i-1)}) + \textstyle\sum_{j\in S^0} \hat{\pi}_j (v_{j'i} - v_{j',(i-1)}) + \displaystyle\sum_{j\in S^-} \hat{\pi}_j (v_{j'i} - v_{j',(i-1)})\right\} \\
&= \max\left\{0, \sum_{j\in S^+} c_j (v_{j'i} - v_{j',(i-1)}) + \sum_{j\in S^0} 0(v_{j'i} - v_{j',(i-1)}) + \sum_{j\in S^-} -c_j (v_{j'i} - v_{j',(i-1)})\right\} \\
&= \max\left\{0, \sum_{j\in S^+} c_j (v_{j'i} - v_{j',(i-1)}) + \sum_{j\in S^-} c_j (v_{j'(i-1)} - v_{j'i})\right\} \qquad (3.34) \\
&= \sum_{j\in S^+} \max\left\{0, c_j (v_{j'i} - v_{j',(i-1)})\right\} + \sum_{j\in S^-} \max\left\{0, c_j (v_{j'(i-1)} - v_{j'i})\right\} \qquad (3.35) \\
&= \sum_{j\in S^+} c_j \max\left\{0, (v_{j'i} - v_{j',(i-1)})\right\} + 0 + \sum_{j\in S^-} c_j \max\left\{0, (v_{j'(i-1)} - v_{j'i})\right\} \\
&= \sum_{j\in S^+} c_j \pi^+_{ij'} + \sum_{j\in S^-} c_j \pi^-_{ij'},
\end{aligned}$$

where $j' = j - (2q-2)$ with the equivalence of (3.34) and (3.35) begin a result of the lemma. It is also shown that

$$\hat{\pi}_{i+q-1} = \sum_{j\in S^+} c_j \pi^+_{(i+q-1),j'} + \sum_{j\in S^-} c_j \pi^-_{(i+q-1),j'},\ \forall\, i = 2, \ldots q-1,$$

by interchanging i and $i-1$ in the above equations where $j' = j - (2q-2)$.

Next, $\hat{\pi}_q = \sum_{j\in S^+} c_j \pi^+_{qj'} + \sum_{j\in S^-} c_j \pi^-_{qj'} = 0$, with $j' = j - (2q-2)$, since $\boldsymbol{\pi}^+_{qi} = 0$ and $\boldsymbol{\pi}^-_{qi} = 0$ for all $i = 1, \ldots, \lceil \log_2(q-1) \rceil$. Finally, consider any $\hat{\pi}_i$ for any $i \in S$ and recall that the extreme directions $\boldsymbol{\pi}^+_{j'}$ and $\boldsymbol{\pi}^-_{j'}$ where $j' = i - (2q-2)$, have $\pi^+_{ij'} = 1$ and $\pi_{ij'} = -1$, with $\pi^+_{k\ell} = 0$ and $\pi^-_{k\ell} = 0$ for all $k \in S$ with $k \neq i$ and all $\ell = 1, \ldots, \lceil \log_2(q-1) \rceil$ with $\ell \neq j'$. In addition, $c_i = |\hat{\pi}_i|$ for all $i \in S$, thus

$$\hat{\pi}_i = \sum_{j\in S^+} c_j \pi^+_{ij'} + \sum_{j\in S^-} c_j \pi^-_{ij'},\ \forall\, i = 2q-1, \ldots, 2q-2+\lceil \log_2(q-1) \rceil,$$

where $j' = i - (2q-2)$. Thus, all components of $\hat{\boldsymbol{\pi}}$ have been written as a linear combination of the components of other extreme directions which is now given in concise form as

$$\hat{\boldsymbol{\pi}} = \sum_{j \in S^+} c_j \boldsymbol{\pi}_{j'}^{+} + \sum_{j \in S^-} c_j \boldsymbol{\pi}_{j'}^{-}$$

with $j' = j - (2q - 2)$. $\square$

The following Theorem relates the extreme directions of (3.29) to the facets of the projection of X_2' of (3.27) onto the $(\boldsymbol{x}, \boldsymbol{u})$ space denoted $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2')$.

Theorem 2

Every extreme direction of (3.29), that is, $\boldsymbol{\pi}_i^0$ for $i = 1, \ldots, q-1$, $\boldsymbol{\pi}_i^+$ for $i = 1, \ldots, \lceil \log_2(q-1) \rceil$, and $\boldsymbol{\pi}_i^-$ for $i = 1, \ldots, \lceil \log_2(q-1) \rceil$, found in Theorem 1 define facets on $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2')$.

Proof

Two facets are already known for $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2')$. These come from the the constraints of (3.27) that do not contain any $\boldsymbol{\lambda}$ variables and are $\sum_{k=1}^{q} x_k = 1$ and $x_q \geq 0$. To find the remaining facets, we consider the system of equations from (3.27) given as

$$\left[\begin{array}{ccccc|cccc}
1 & 1 & \cdots & 1 & 1 & -1 & -1 & \cdots & -1 \\
 & 1 & \cdots & 1 & 1 & & -1 & \cdots & -1 \\
 & & \ddots & \vdots & \vdots & & & \ddots & \vdots \\
 & & & 1 & 1 & & & & -1 \\
\hline
 & -1 & \cdots & \cdots & -1 & 1 & \cdots & \cdots & 1 \\
 & & \ddots & \ddots & \vdots & & \ddots & \ddots & \vdots \\
 & & & \ddots & \vdots & & & \ddots & \vdots \\
 & & & & -1 & & & & 1 \\
\hline
\multicolumn{5}{c|}{\mathbf{0}^{\lceil \log_2(q-1) \rceil \times q}} & \multicolumn{4}{c}{V}
\end{array}\right]
\left[\begin{array}{c}
x_1 \\ \vdots \\ x_q \\ \hline \lambda_1 \\ \vdots \\ \lambda_{q-1}
\end{array}\right]
\left[\begin{array}{c}
= \\ \geq \\ \vdots \\ \geq \\ \hline \geq \\ \vdots \\ \vdots \\ \geq \\ \hline =
\end{array}\right]
\left[\begin{array}{c}
0 \\ 0 \\ \vdots \\ 0 \\ \hline 0 \\ \vdots \\ \vdots \\ 0 \\ \hline \boldsymbol{u}
\end{array}\right],$$

where $\mathbf{0}^{\lceil \log_2(q-1) \rceil \times q}$ is a $\lceil \log_2(q-1) \rceil \times q$ matrix whose elements are all zeros and V is a matrix of size $\lceil \log_2(q-1) \rceil \times (q-1)$ where the k^{th} column of V corresponds to the vector $\boldsymbol{v}_k$. In addition, the two constraints corresponding to zero rows of the $\boldsymbol{\lambda}$ variables have been removed from the system. Now, multiply this system by $\boldsymbol{\pi}_i^0$ for $i = 1, \ldots, q-1$, $\boldsymbol{\pi}_i^+$ for $i = 1, \ldots, \lceil \log_2(q-1) \rceil$, and $\boldsymbol{\pi}_i^-$ for $i = 1, \ldots, \lceil \log_2(q-1) \rceil$, recalling that since each $\boldsymbol{\pi}_i^0$, $\boldsymbol{\pi}_i^+$, and $\boldsymbol{\pi}_i^-$, are extreme directions of (3.29) the $\boldsymbol{\lambda}$ variables will not appear in the resulting constraints.

First, consider the extreme directions $\boldsymbol{\pi}_i^0 = \boldsymbol{e}_i + \boldsymbol{e}_{i+q-1}$ for $i = 1, \ldots, q-1$. For each of these cases, the two nonzero multipliers of $\boldsymbol{\pi}_i^0$ are $\pi_{ii}^0 = \pi_{(i+1-q),i}^0 = 1$ which give $\sum_{k=i}^q x_i + \sum_{k=i+1}^q -x_i \geq 0$, or, more concisely, $x_i \geq 0$. Observe that each constraint $x_i \geq 0$ created by $\boldsymbol{\pi}_i^0$ for $i = 1, \ldots, q-1$, is a facet of $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2')$ because this constraint cannot be represented by a linear combination the other facets. Thus, the facets found so far for $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2')$ are $\sum_{k=1}^q x_k = 1$, $x_q \geq 0$, and $x_k \geq 0$ for $k = 1, \ldots, q-1$ found using the multipliers $\boldsymbol{\pi}_1^0, \ldots, \boldsymbol{\pi}_{q-1}^0$.

Next, none of the facets identified for $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2')$ contain any of the $\boldsymbol{u}$ variables so consider the extreme directions $\boldsymbol{\pi}_i^+$ for $i = 1, \ldots, \lceil \log_2(q-1) \rceil$ where, exactly one $\boldsymbol{u}$ variables is selected. Specifically, for $\boldsymbol{\pi}_i^+$, the variable u_i has a multiplier of 1 with all other $\boldsymbol{u}$ variables having multipliers of zero. The valid constraints given for $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2')$ are of the form

$$v_{i1} \sum_{j=1}^q x_j + \sum_{j=2}^{q-1} \left(\max\{0, v_{ij} - v_{i,(j-1)}\} \left(\sum_{k=j}^q x_k \right) + \max\{0, v_{i,(j-1)} - v_{ij}\} \left(\sum_{k=j+1}^q -x_i \right) \right) \geq u_i,$$

for each $i = 1, \ldots, \lceil \log_2(q-1) \rceil$. These constraints are facets as u_i appears in exactly one of the above constraints and none other. The above representation of the facets is cumbersome so we rewrite them in close form without use of a max function.

Recall that each $v_{ij} \in \{0, 1\}$ for $i = 1, \ldots, \lceil \log_2(q-1) \rceil$, and $j = 1, \ldots, q-1$, so for a given binary row, $[v_{i1}, \ldots, v_{i,(q-1)}]$ of V, consider the first v_{ij} with $v_{ij} = 1$, which adds $\sum_{k=j}^q x_k$ to the left side of the constraint. Next, find the index of the row vector, say ℓ with $\ell > j$, such that $v_{i\ell} = 0$ and so $v_{ik} = 1$ for $j \leq k < \ell$. This adds $-\sum_{k=\ell+1}^q x_k$ to the left side of the constraint. Combining the two sums gives $\sum_{k=j}^{\ell} x_k$ on the left side of the constraint. Continuing, look for the next index p with $p > \ell$ such that $v_{ip} = 1$ and $v_{ik} = 0$ for $\ell \leq k < p$, which adds $\sum_{k=p}^q x_k$ to the left side of the constraint and gives $\sum_{k=j}^{\ell} x_k + \sum_{k=p}^q x_k$. Repeating the above steps until each component of $[v_{i1}, \ldots, v_{i,(q-1)}]$ is considered gives the x_k that are on the left side of the constraint. This process provides insight into rewriting the constraints. From the above description, the variable x_1 is on the left side of the constraint if $v_{i1} = 1$, x_j is on the left side of the constraint if $v_{ij} = 1$ *or* $v_{i,(j-1)} = 1$ for $j = 2, \ldots, q-1$, and x_q is on the left side of the constraint if $v_{i,(q-1)} = 1$. Using this information, define new binary coefficients $b^1_{ij} \in \{0,1\}$ for $i = 1,\ldots,\lceil \log_2(q-1)\rceil$ and $j = 1,\ldots,q$ as

$$b^1_{ij} = \left\{ \begin{array}{ll} 1 & \text{if } v_{ij} = 1 \text{ } or \text{ } v_{i,(j-1)} = 1 \\ 0 & \text{otherwise} \end{array} \right\} \ \forall\ (i,j),\ i = 1,\ldots,\lceil \log_2(q-1)\rceil,\ j = 2,\ldots,q-1,$$

with $b^1_{i1} = v_{i1}$ and $b^1_{iq} = v_{i,(q-1)}$ for all $i = 1,\ldots,\lceil \log_2(q-1)\rceil$. Thus, the facets above are rewritten as

$$\sum_{j=1}^{q} b^1_{ij} x_j \geq u_i,\ \forall\ i = 1,\ldots,\lceil \log_2(q-1)\rceil.$$

For example, let $q = 9$ and consider the binary row vector $[v_{i1},\ldots,v_{i8}] = [0,1,1,0,0,1,1,0]$. The coefficients are given as $[b^1_{i1}, b^1_{i2}, b^1_{i3}, b^1_{i4}, b^1_{i5}, b^1_{i6}, b^1_{i7}, b^1_{i8}, b^1_{i9}] = [0,1,1,1,0,1,1,1,0]$ and the facet is $x_2 + x_3 + x_4 + x_6 + x_7 + x_8 \geq u_i$.

The last set of facets of $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2')$ come from the extreme directions $\boldsymbol{\pi}_i^-$ for $i = 1,\ldots,\lceil \log_2(q-1)\rceil$ where, exactly one $\boldsymbol{u}$ variables is selected. In this case, for $\boldsymbol{\pi}_i^-$, the variable u_i has a multiplier of -1 with all other $\boldsymbol{u}$ variables having multipliers of zero. The valid constraints given for $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2')$ are of the form

$$-v_{i1} \sum_{j=1}^{q} x_j + \sum_{j=2}^{q-1} \left(\max\{0, v_{i,(j-1)} - v_{ij}\} \left(\sum_{k=j}^{q} x_k \right) + \max\{0, v_{ij} - v_{i,(j-1)}\} \left(\sum_{k=j+1}^{q} -x_i \right) \right) \geq -u_i,$$

for each $i = 1,\ldots,\lceil \log_2(q-1)\rceil$. These constraints are facets as $-u_i$ appears on the right side in exactly one of the above constraints and none other.

Again, we simplify the notation for the constraint given above by looking at the row vector $[v_{i1},\ldots,v_{i,(q-1)}]$. Consider the first v_{ij} with $v_{ij} = 1$, which adds $\sum_{k=j+1}^{q} -x_k$ if $j \neq 1$ and $\sum_{k=1}^{q} -x_k$ if $j = 1$, to the left side of the constraint. Next, find the first index of the row vector, say ℓ with $\ell > j$, such that $v_{i\ell} = 0$ and so $v_{ik} = 1$ for $j \leq k < \ell$. This adds $\sum_{k=\ell}^{q} x_k$ to the left side of the constraint. Combining the two sums gives $\sum_{k=j+1}^{\ell-1} -x_k$ if $j \neq 1$ and $\sum_{k=1}^{\ell-1} -x_k$ if $j = 1$, on the left side of the constraint. Continuing, look for the next index p with $p > \ell$ such that $v_{ip} = 1$ and, thus, $v_{ik} = 0$ for $\ell \leq k < p$. This adds $\sum_{k=p+1}^{q} -x_k$ to the left side of the constraint and gives $\sum_{k=j}^{\ell-1} -x_k + \sum_{k=p+1}^{q} -x_k$ if $j \neq 1$, and $\sum_{k=1}^{\ell-1} -x_k + \sum_{k=p+1}^{q} -x_k$, if $j = 1$. Repeating the above steps until each component of $[v_{i1},\ldots,v_{i,(q-1)}]$ is considered, gives the $-x_k$ that are on the left side of the constraint. From the above description, the variable $-x_1$ is on the left side of the constraint if $v_{i1} = 1$, $-x_j$ is on the left side of the constraint if $v_{ij} = 1$ *and* $v_{i,(j-1)} = 1$ for $j = 2,\ldots,q-1$, and

$-x_q$ is on the left side of the constraint if $v_{i,(q-1)} = 1$. Using this information, define new binary coefficients $b^2_{ij} \in \{0, 1\}$ for $i = 1, \ldots, \lceil \log_2(q-1) \rceil$ and $j = 1, \ldots, q$ as

$$b^2_{ij} = \left\{ \begin{array}{ll} 1 & \text{if } v_{ij} = 1 \textit{ and } v_{i,(j-1)} = 1 \\ 0 & \text{otherwise} \end{array} \right\} \;\forall\, (i, j),\; i = 1, \ldots, \lceil \log_2(q-1) \rceil,\; j = 2, \ldots, q-1,$$

with $b^2_{i1} = v_{i1}$ and $b^2_{iq} = v_{i,(q-1)}$ for all $i = 1, \ldots, \lceil \log_2(q-1) \rceil$. Thus, the facets above are rewritten as

$$\sum_{j=1}^{q} -b^2_{ij} x_j \geq -u_i,\; \forall\, i = 1, \ldots, \lceil \log_2(q-1) \rceil.$$

For example, let $q = 9$ and consider the binary row vector $[v_{i1}, \ldots, v_{i8}] = [1, 0, 0, 1, 1, 0, 0, 1]$. The coefficients are given as $[b^2_{i1}, b^2_{i2}, b^2_{i3}, b^2_{i4}, b^2_{i5}, b^2_{i6}, b^2_{i7}, b^2_{i8}, b^2_{i9}] = [1, 0, 0, 0, 1, 0, 0, 0, 1]$ and the facet is $-x_1 - x_5 - x_9 \geq -u_i$. $\Box$

As Theorem 2 showed, all the extreme directions of (3.29) give facets to $\mathrm{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X'_2)$. This projection of X'_2 onto the $(\boldsymbol{x}, \boldsymbol{u})$ is also the projection of X_2 onto the $(\boldsymbol{x}, \boldsymbol{u})$ since there exists a nonsingular linear transformations between X'_2 and X_2. So, we denote the projection of X_2 onto the $(\boldsymbol{x}, \boldsymbol{u})$ space as $\mathrm{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ which is given as

$$\begin{aligned} \mathrm{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2) = \Bigg\{ (\boldsymbol{x}, \boldsymbol{u}) :\; & \sum_{j=1}^{q} x_j = 1,\; \boldsymbol{x} \geq \boldsymbol{0},\; \boldsymbol{u} \text{ binary}, \\ & \sum_{j=1}^{q} b^1_{ij} x_j \geq u_i \;\forall\, i = 1, \ldots, \lceil \log_2(q-1) \rceil, \\ & \sum_{j=1}^{q} -b^2_{ij} x_j \geq -u_i \;\forall\, i = 1, \ldots, \lceil \log_2(q-1) \rceil \Bigg\}, \end{aligned} \tag{3.36}$$

where

$$b^1_{ij} = \left\{ \begin{array}{ll} 1 & \text{if } v_{ij} = 1 \text{ or } v_{i,(j-1)} = 1 \\ 0 & \text{otherwise} \end{array} \right\} \;\forall\, (i, j),\; i = 1, \ldots, \lceil \log_2(q-1) \rceil,\; j = 2, \ldots, q-1,$$

with $b^1_{i1} = v_{i1}$ and $b^1_{iq} = v_{i,(q-1)}$ for all $i = 1, \ldots, \lceil \log_2(q-1) \rceil$, and

$$b^2_{ij} = \left\{ \begin{array}{ll} 1 & \text{if } v_{ij} = 1 \text{ and } v_{i,(j-1)} = 1 \\ 0 & \text{otherwise} \end{array} \right\} \; \forall\, (i,j),\; i = 1, \ldots, \lceil \log_2(q-1) \rceil,\; j = 2, \ldots, q-1,$$

with $b^2_{i1} = v_{i1}$ and $b^2_{iq} = v_{i,(q-1)}$ for all $i = 1, \ldots, \lceil \log_2(q-1) \rceil$. Note, that the vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{q-1} \in \mathbb{R}^{\lceil \log_2(q-1) \rceil}$ must be in a compatible order in order for $\mathrm{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ to be a valid representation. This formulation has q nonnegative variables $\boldsymbol{x}$, $\lceil \log_2(q-1) \rceil$ binary variables $\boldsymbol{u}$ with $1+2\lceil \log_2(q-1) \rceil$ constraints.

In summary, when a collection of distinct binary vectors, $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{q-1}, \in \mathbb{R}^{\lceil \log_2(q-1) \rceil}$, are in a compatible order, then the characterization of all extreme directions of the projection cone of (3.29) is easily obtained and gives a representation of $\mathrm{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ using $1 + 2\lceil \log_2(q-1) \rceil$ constraints. However, if a non-compatible order of binary vectors is used, then the number of constraints defining the convex hull of $\mathrm{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ is not necessary $1 + 2\lceil \log_2(q-1) \rceil$. We end this section by giving the following example of the projection, $\mathrm{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$, obtained from a set of binary vectors not in a compatible order.

Example 4

Let $q = 5$ and define the binary vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4 \in \mathbb{R}^2$ as $\boldsymbol{v}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $\boldsymbol{v}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $\boldsymbol{v}_3 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, and $\boldsymbol{v}_4 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$. We seek the extreme directions of

$$\left[\begin{array}{cccc|cccc|cc} -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ -1 & -1 & -1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 0 \end{array}\right] \left[\begin{array}{c} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \\ \hline \pi_5 \\ \pi_6 \\ \pi_7 \\ \pi_8 \\ \hline \pi_9 \\ \pi_{10} \end{array}\right] = \mathbf{0},\ \pi_i \geq 0,\ \forall\, i = 2, \ldots, 8,$$

which are

$$\left[\begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \\ \hline 1 \\ 0 \\ 0 \\ 0 \\ \hline 0 \\ 0 \end{array}\right], \left[\begin{array}{c} 0 \\ 1 \\ 0 \\ 0 \\ \hline 0 \\ 1 \\ 0 \\ 0 \\ \hline 0 \\ 0 \end{array}\right], \left[\begin{array}{c} 0 \\ 0 \\ 1 \\ 0 \\ \hline 0 \\ 0 \\ 1 \\ 0 \\ \hline 0 \\ 0 \end{array}\right], \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ 1 \\ \hline 0 \\ 0 \\ 0 \\ 1 \\ \hline 0 \\ 0 \end{array}\right], \left[\begin{array}{c} 0 \\ 1 \\ 0 \\ 0 \\ \hline 0 \\ 0 \\ 0 \\ 1 \\ \hline 0 \\ 1 \end{array}\right], \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ 1 \\ \hline 0 \\ 1 \\ 0 \\ 0 \\ \hline 0 \\ -1 \end{array}\right], \left[\begin{array}{c} 1 \\ 0 \\ 1 \\ 0 \\ \hline 0 \\ 0 \\ 0 \\ 2 \\ \hline 1 \\ 1 \end{array}\right], \left[\begin{array}{c} -1 \\ 0 \\ 0 \\ 2 \\ \hline 0 \\ 0 \\ 1 \\ 0 \\ \hline -1 \\ -1 \end{array}\right], \left[\begin{array}{c} 1 \\ 0 \\ 1 \\ 0 \\ \hline 0 \\ 2 \\ 0 \\ 0 \\ \hline 1 \\ -1 \end{array}\right], \left[\begin{array}{c} -1 \\ 2 \\ 0 \\ 0 \\ \hline 0 \\ 0 \\ 1 \\ 0 \\ \hline -1 \\ 1 \end{array}\right].$$

Notice that some extreme directions have two nonzero components on the constraints $\sum_{k=1}^{4} \lambda_k \boldsymbol{v}_k = \boldsymbol{u}$ which does not occur when the vectors $\boldsymbol{v}_1$, $\boldsymbol{v}_2$, $\boldsymbol{v}_3$, and $\boldsymbol{v}_4$ are in a compatible order. The projection having 5 nonnegative variables $\boldsymbol{x}$, 2 binary variables $\boldsymbol{u}$, and 7 constraints is

$$\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2) = \left\{ (\boldsymbol{x},\boldsymbol{u}) : \begin{array}{c} x_1+x_2+x_3+x_4+x_5 = 1 \\ \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 1 & 1 & 2 & 2 & 0 \\ -1 & -1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \geq \begin{bmatrix} u_2 \\ -u_2 \\ u_1+u_2 \\ -u_1-u_2 \\ u_1-u_2 \\ -u_1+u_2 \end{bmatrix} \end{array}, \ \boldsymbol{x} \geq \boldsymbol{0}, \ \boldsymbol{u} \text{ binary} \right\},$$

which has more than the $1 + 2\lceil \log_2(5-1) \rceil = 5$ constraints obtained when using binary vectors in a compatible order.

3.4 Modeling Piecewise-Linear Functions using SOS-2 Restrictions

SOS-2 restrictions easily model piecewise-linear functions. Consider such a function $f(y)$ defined over an interval $[\theta_1, \theta_q]$, and having q break points $\theta_1 < \theta_2 < \cdots < \theta_q$ so that

$$f(y) = f(\theta_k) + (y - \theta_k) \left(\frac{f(\theta_{k+1}) - f(\theta_k)}{\theta_{k+1} - \theta_k} \right) \text{ when } y \in [\theta_k, \theta_{k+1}] \text{ for } k \in \{1, \ldots, q-1\}.$$

This function is represented by enforcing SOS-2 restrictions on a set of nonnegative variables $\boldsymbol{x}$ along with the constraint $\sum_{k=1}^{q} x_k = 1$, as

$$PW = \left\{ (y, f(y), \boldsymbol{x}) : \ y = \sum_{k=1}^{q} \theta_k x_k, \ f(y) = \sum_{k=1}^{q} f(\theta_k) x_k, \ \sum_{k=1}^{q} x_k = 1, \ \boldsymbol{x} \geq \boldsymbol{0}, \text{ SOS-2 on } \boldsymbol{x} \right\}. \quad (3.37)$$

The SOS-2 restriction on the variables $\boldsymbol{x}$ along with the constraints $\sum_{k=1}^{q} x_k = 1$ and $\boldsymbol{x} \geq \boldsymbol{0}$ can be enforced using X_2 of (3.16)–(3.21) by including the additional variables $\boldsymbol{w}$ and $\boldsymbol{u}$ within PW as

$$PW_1 = \left\{ (y, f(y), \boldsymbol{x}, \boldsymbol{w}, \boldsymbol{u}) : \ y = \sum_{k=1}^{q} \theta_k x_k, \ f(y) = \sum_{k=1}^{q} f(\theta_k) x_k, \ (\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{u}) \in X_2 \text{ of } (3.16) - (3.21) \right\}, \quad (3.38)$$

or X_2' of (3.27) by including the variables $\boldsymbol{\lambda}$ and $\boldsymbol{u}$ within PW as

$$PW_2 = \left\{ (y, f(y), \boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}) : \ y = \sum_{k=1}^{q} \theta_k x_k, \ f(y) = \sum_{k=1}^{q} f(\theta_k) x_k, \ (\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{u}) \in X_2' \text{ of } (3.27) \right\}, \tag{3.39}$$

or $\mathrm{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ of (3.36) by including the additional variables $\boldsymbol{u}$ within PW as

$$PW_3 = \left\{ (y, f(y), \boldsymbol{x}, \boldsymbol{u}) : \ y = \sum_{k=1}^{q} \theta_k x_k, \ f(y) = \sum_{k=1}^{q} f(\theta_k) x_k, \ (\boldsymbol{x}, \boldsymbol{u}) \in \mathrm{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2) \text{ of } (3.36) \right\}. \tag{3.40}$$

Observe that the restrictions $x_1 = w_{11}$, $x_j = w_{(j-1)j} + w_{jj}$ for $j = 2, \ldots, q-1$, and $x_q = w_{(q-1)q}$ in X_2 allow for a reduction in size of PW_1 via a substitution of all variables $\boldsymbol{x}$ from the problem to give the formulation PW_1' below.

$$PW_1' = \left\{ \begin{aligned} (y, f(y), \boldsymbol{w}, \boldsymbol{u}) : y = \theta_1 w_{11} + \sum_{k=2}^{q-1} \theta_k (w_{(k-1)k} + w_{kk}) + \theta_q w_{(q-1)q}, \\ f(y) = f(\theta_1) w_{11} + \sum_{k=2}^{q-1} f(\theta_k)(w_{(k-1)k} + w_{kk}) + f(\theta_q) w_{(q-1)q}, \\ w_{kk}, w_{k(k+1)} \geq 0 \ \forall \ k = 1, \ldots, q-1, \\ \sum_{k=1}^{q-1} (w_{kk} + w_{k(k+1)}) = 1, \\ \sum_{k=1}^{q-1} (w_{kk} + w_{k(k+1)}) \boldsymbol{v}_k = \boldsymbol{u}, \ \boldsymbol{u} \text{ binary} \end{aligned} \right\} \tag{3.41}$$

In summary, PW_1' of (3.41), PW_2 of (3.39), and PW_3 of (3.40) all model a single piecewise-linear function $f(y)$ having q break points. The formulation PW_1' uses $2(q-1)$ nonnegative and continuous variables $\boldsymbol{w}$, $\lceil \log_2(q-1) \rceil$ binary variables $\boldsymbol{u}$, and $3 + \lceil \log_2(q-1) \rceil$ constraints. PW_2 requires 1 nonnegative and continuous variable x_q with $2(q-1)$ continuous variables $x_1, \ldots, x_{q-1}$ and $\boldsymbol{\lambda}$, $\lceil \log_2(q-1) \rceil$ binary variables $\boldsymbol{u}$, and $4 + \lceil \log_2(q-1) \rceil + 2(q-1)$ constraints. PW_3 only needs q nonnegative and continuous variables $\boldsymbol{x}$, $\lceil \log_2(q-1) \rceil$ binary variables $\boldsymbol{u}$, and $3 + 2\lceil \log_2(q-1) \rceil$ constraints. These forms are exhibited in the example below.

<u>Example 5</u>

Consider the piecewise-linear function $f(y)$ defined on the interval $[0, 6]$, and having the $q = 5$ break points $y = 0, 1, 3, 4, 6$, with $f(0) = 0$, $f(1) = 5$, $f(3) = 9$, $f(4) = 10$, and $f(6) = 11$. The function is depicted in Figure 1. Define the compatible vectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4 \in \mathbb{R}^2$ to be $\boldsymbol{v}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$,

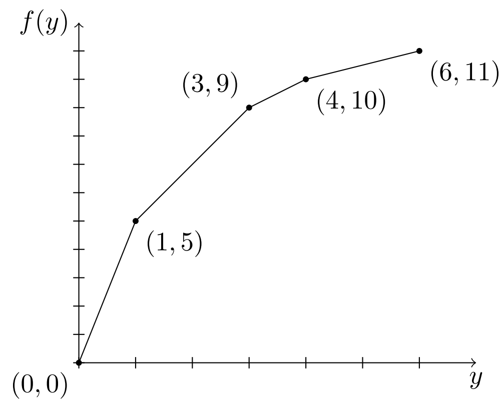


Figure 3.1: Piecewise-linear function $f(y)$ with $q = 5$ break points.

$\boldsymbol{v}_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $\boldsymbol{v}_3 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, and $\boldsymbol{v}_4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

First, consider formulation PW_1'. The five equations of (3.41) in nonnegative, continuous $(w_{11}, w_{12}, w_{22}, w_{23}, w_{33}, w_{34}, w_{44}, w_{45})$ and binary (u_1, u_2) are listed in matrix notation as

$$\begin{bmatrix} 0 & 1 & 1 & 3 & 3 & 4 & 4 & 6 \\ 0 & 5 & 5 & 9 & 9 & 10 & 10 & 11 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} w_{11} \\ w_{12} \\ w_{22} \\ w_{23} \\ w_{33} \\ w_{34} \\ w_{44} \\ w_{45} \end{bmatrix} = \begin{bmatrix} y \\ f(y) \\ 1 \\ u_1 \\ u_2 \end{bmatrix}.$$

Next, consider the 14 constraints of PW_2 where $x_5 \geq 0$ with continuous variables $(x_1, x_2, x_3, x_4, x_5)$

and $(\lambda_1, \lambda_2, \lambda_3, \lambda_4)$, and binary variables (u_1, u_2) listed in matrix notation as

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & -1 & -1 & -1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & -1 \\ 0 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ 0 & 0 & -1 & -1 & -1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & -1 & -1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{bmatrix} \geq \mathbf{0},$$

and

$$\begin{bmatrix} 0 & 1 & 3 & 4 & 6 & 0 & 0 & 0 & 0 \\ 0 & 5 & 9 & 10 & 11 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{bmatrix} = \begin{bmatrix} y \\ f(y) \\ 1 \\ 0 \\ u_1 \\ u_2 \end{bmatrix}.$$

Last, the 7 constraints of PW_3 with the nonnegative and continuous variables $(x_1, x_2, x_3, x_4, x_5)$ and the binary variables (u_1, u_2) are given as

$$\begin{bmatrix} 0 & 1 & 3 & 4 & 6 \\ 0 & 5 & 9 & 10 & 11 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} y \\ f(y) \\ 1 \end{bmatrix}, \quad \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \geq \begin{bmatrix} u_1 \\ u_2 \\ -u_1 \\ -u_2 \end{bmatrix}.$$

3.4.1 Alternate Approaches for Piecewise-Linear Functions using SOS-2 Restrictions

Many alternative approaches are found in [5, 6] to model piecewise-linear functions with SOS-2 restrictions using any number of auxiliary binary variables and constraints. Computational trials by [6] demonstrate that formulations using a logarithmic number of auxiliary binary variables and constraints are most efficient in respect to solution times for larger problem instances. Thus, we only compare our models, X_2 of (3.16)–(3.21) and $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ of (3.36), for SOS-2 restrictions with two models found in [5, 6] that use a logarithmic number of constraints and binary variables.

The first approach we mention found in [5, 6] applicable to SOS-2 restrictions uses the variables $(\boldsymbol{x}, \boldsymbol{u})$ and is, in fact, very close to $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$. First, define q^* as the smallest integer with $q^* \geq q$ such that $q^* - 1$ is a power of 2. That is, $\lceil \log_2(q^* - 1) \rceil = \log_2(q^* - 1)$. This formulation uses the vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{q^*-1} \in \mathbb{R}^{\log_2(q^*-1)}$ and requires them to be in a compatible order. These vectors give a Hamiltonian path using every vertex on the $\log_2(q^* - 1)$-dimensional hypercube. Now, define $\hat{b}^1_{ij}$ and $\hat{b}^2_{ij}$ as

$$\hat{b}^1_{ij} = \left\{ \begin{array}{ll} 1 & \text{if } v_{ij} = 1 \text{ or } v_{i,(j-1)} = 1 \\ 0 & \text{otherwise} \end{array} \right\} \ \forall (i,j),\ i = 1, \ldots, \log_2(q^* - 1),\ j = 2, \ldots, q^* - 1,$$

with $\hat{b}^1_{i1} = v_{i1}$ and $\hat{b}^1_{iq^*} = v_{i,(q^*-1)}$ for all $i = 1, \ldots, \log_2(q^* - 1)$, and

$$\hat{b}^2_{ij} = \left\{ \begin{array}{ll} 1 & \text{if } v_{ij} = 1 \text{ and } v_{i,(j-1)} = 1 \\ 0 & \text{otherwise} \end{array} \right\} \ \forall (i,j),\ i = 1, \ldots, \log_2(q^* - 1),\ j = 2, \ldots, q^* - 1,$$

with $\hat{b}_{i1}^2 = v_{i1}$ and $\hat{b}_{iq^*}^2 = v_{i,(q^*-1)}$ for all $i = 1, \ldots, \log_2(q^*-1)$. The formulation denoted *Log* in [6] is

$$Log = \left\{ (\boldsymbol{x}, \boldsymbol{u}) : \begin{array}{l} \sum_{j=1}^{q^*} x_j = 1, \ \boldsymbol{x} \geq \boldsymbol{0}, \ \boldsymbol{u} \text{ binary}, \\ \sum_{j=1}^{q^*} (1 - \hat{b}_{ij}^1) x_j \leq 1 - u_i \ \forall \ i = 1, \ldots, \log_2(q^* - 1), \\ \sum_{j=1}^{q^*} \hat{b}_{ij}^2 x_j \leq u_i \ \forall \ i = 1, \ldots, \log_2(q^* - 1), \\ x_i = 0 \ \forall \ i = q+1, \ldots, q^* \end{array} \right\}.$$

Obviously, this formulation can be reduced in size when $q < q^*$ which the authors of [5, 6] do by substituting out the variables $x_{q+1}, \ldots, x_{q^*}$ which gives

$$Log = \left\{ (\boldsymbol{x}, \boldsymbol{u}) : \begin{array}{l} \sum_{j=1}^{q} x_j = 1, \ \boldsymbol{x} \geq \boldsymbol{0}, \ \boldsymbol{u} \text{ binary}, \\ \sum_{j=1}^{q} (1 - \hat{b}_{ij}^1) x_j \leq 1 - u_i \ \forall \ i = 1, \ldots, \log_2(q^* - 1), \\ \sum_{j=1}^{q} \hat{b}_{ij}^2 x_j \leq u_i \ \forall \ i = 1, \ldots, \log_2(q^* - 1) \end{array} \right\}. \tag{3.42}$$

Notice, that even though we have defined $\hat{b}_{ij}^1$ and $\hat{b}_{ij}^2$ for all $i = 1, \ldots, \log_2(q^*-1)$, and $j = 1, \ldots, q^*$, the reduced formulation given in (3.42) only uses those $\hat{b}_{ij}^1$ and $\hat{b}_{ij}^2$ for $i = 1, \ldots, \lceil \log_2(q-1) \rceil$, and $j = 1, \ldots, q$. To conclude, the simplified formulation for *Log* of (3.42), just like $\mathrm{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$, uses q nonnegative continuous variables $\boldsymbol{x}$, $\lceil \log_2(q-1) \rceil = \log_2(q^*-1)$ binary variables $\boldsymbol{u}$, and $1 + 2\lceil \log_2(q-1) \rceil$ constraints.

The second model for SOS-2 restrictions for piecewise-linear functions using a logarithmic number of binary variables and constraints is given in [6] and is similar to X_2 of (3.16)–(3.21) because it uses the same variables $(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{u})$. This formulation, denoted as in [6] as $DLog$, is given as

$$DLog = \left\{ \begin{array}{l} (\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{u}) : w_{kk}, w_{k(k+1)} \geq 0 \; \forall \; k = 1 \ldots, q-1, \\ \\ \qquad\qquad x_1 = w_{11}, \\ \qquad\qquad x_j = w_{(j-1)j} + w_{jj} \; \forall \; j = 1, \ldots, q-1, \\ \qquad\qquad x_q = w_{(q-1)q}, \\ \qquad\qquad \sum_{k=1}^{q-1} (w_{kk} + w_{k(k+1)}) = 1, \\ \qquad\qquad \sum_{k=1}^{q-1} (w_{kk} + w_{k(k+1)}) \boldsymbol{v}_k \leq \boldsymbol{u}, \\ \qquad\qquad \sum_{k=1}^{q-1} (w_{kk} + w_{k(k+1)}) (\mathbf{1} - \boldsymbol{v}_k) \leq \mathbf{1} - \boldsymbol{u}, \; \boldsymbol{u} \text{ binary} \end{array} \right\}. \tag{3.43}$$

Both formulations provided in this section are able to model SOS-2 restrictions over a set of nonnegative $\boldsymbol{x}$ variables having $\sum_{k=1}^{q} x_k = 1$. Thus, each formulation can model a piecewise-linear function represented by (3.37). We assume that, if possible, reductions to the models, such as substituting out the variables $\boldsymbol{x}$ as in PW_1 when X_2 is used, are made. Any substitutions do not affect the SOS-2 restrictions, but provide a reduced number of constraints and variables within the formulation.

3.4.2 Comparisons of SOS-2 Models for Piecewise-Linear Functions

Of the given formulations, we compare X_2 of (3.16)–(3.21), $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ of (3.36), Log of (3.42), $DLog$ of (3.43) because each of these formulations only requires a logarithmic number of binary variables and constraints to model SOS-2 restrictions. It is known that the formulations Log and $DLog$ are *locally ideal* from [5, 6]. That is, conv(Log)= $\overline{Log}$, and conv($DLog$)= $\overline{DLog}$ where $\overline{Log}$ and $\overline{DLog}$ are the continuous relaxations of Log and $DLog$, respectively. The formulation X_2 of (3.16)–(3.21) is trivially ideal. To explain, let $\overline{X}_2$ denote the continuous relaxation of X_2 obtained by relaxing the $\boldsymbol{u}$ binary restrictions to $\mathbf{0} \leq \boldsymbol{u} \leq \mathbf{1}$. As with the system found in (3.6), the relaxation $\overline{X}_2$ has only binary extreme points. Observe that the constraints $\sum_{k=1}^{q-1}(w_{kk} + w_{k(k+1)}) = 1, \; \boldsymbol{w} \geq \mathbf{0}$ found within $\overline{X}_2$ of (3.16)–(3.21) have $2(q-1)$ binary extreme points and $\sum_{k=1}^{q-1}(w_{kk}+w_{k(k+1)})\boldsymbol{v}_k = \boldsymbol{u}$

with (3.17)–(3.19) not restrictive in $\boldsymbol{x}$ or $\boldsymbol{w}$. Also, $(\boldsymbol{x}, \boldsymbol{w})$ binary implies $\boldsymbol{u}$ binary, giving $conv(X_2) = \overline{X}_2$ (so that X_2 of (3.16)–(3.21) is locally ideal). Since X_2 is locally ideal, we also conclude that $\text{conv}(X_2') = \overline{X}_2'$ and $\text{conv}(\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)) = \overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$, with $\overline{X}_2'$ and $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ being the continuous relaxations of X_2' and $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$, respectively. To explain, X_2' is obtained from X_2 via a non-singular linear transformation which means X_2' is locally ideal. In addition, $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$, is also ideal, because it is obtained from X_2' by projecting out the $\boldsymbol{\lambda}$ variables.

Continuing with the comparisons of the formulations given above, we have shown that each formulation is locally ideal and so we first look at X_2 of (3.16)–(3.21) and $DLog$ of (3.43) because they are in the same variables space but contain a different number of constraints. It is shown in [1] that $\overline{X}_2 = \overline{DLog}$ where $\overline{X}_2$ and $\overline{DLog}$ are the continuous relaxations of X_2 and $DLog$, respectively. This results holds for any set of vectors $\boldsymbol{v}_k$.

Next, consider $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ and Log which are both in the $(\boldsymbol{x}, \boldsymbol{u})$ variable space. Both formulations require the binary vectors $\boldsymbol{v}_k$ to be in a compatible order. As shown above, both formulations are locally ideal, but are different for any q such that $\log_2(q-1) < \lceil \log_2(q-1) \rceil$. The following theorem describes the relationship between $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ and Log.

Theorem 3

Let $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ and Log be defined using the binary vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{q-1} \in \mathbb{R}^{\lceil \log_2(q-1) \rceil}$ in a compatible order. Also, let $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ and $\overline{Log}$ be the continuous relaxations of $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ and Log, respectively. Given any $q = q^*$, with q^* satisfying the equation $\lceil \log_2(q^*-1) \rceil = \log_2(q^*-1)$, then $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2) = \overline{Log}$. For any $q < q^*$, then $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2) \subset \overline{Log}$.

Proof

First, consider the case for any $q = q^*$, with q^* satisfying the equation $\lceil \log_2(q^*-1) \rceil = \log_2(q^*-1)$. Observe that $b_{ij}^1 = \hat{b}_{ij}^1$ and $b_{ij}^2 = \hat{b}_{ij}^2$ for all $i = 1, \ldots, \log_2(q^*-1)$, and $j = 1, \ldots, q^*$. Thus, all constraints of $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ and $\overline{Log}$ are the same and so $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2) = \overline{Log}$.

Now, consider any q such that $q < q^*$ and the binary vectors $\boldsymbol{v}_q$ and $\boldsymbol{v}_{q+1}$. Since the vectors are in a compatible order, we know that $\boldsymbol{v}_q$ and $\boldsymbol{v}_{q+1}$ differ by one component, say component k.

Case 1

Let $v_{kq} = 1$ and $v_{k,(q+1)} = 0$, then, by definition, $b_{kq}^2 = 1$, since $v_{kq} = 1$, but $\hat{b}_{kq}^2 = 0$, because $v_{k,(q+1)} = 0$. In addition, $b_{ij}^2 = \hat{b}_{ij}^2$ for all $i = 1, \ldots, \lceil \log_2(q-1) \rceil$, and $j = 1, \ldots, q$ with $i \neq k$ and

$j \neq q$, and $b^1_{ij} = \hat{b}^1_{ij}$ for all $i = 1, \dots, \lceil \log_2(q-1) \rceil$, and $j = 1, \dots, q$. As a result, the formulations $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ and $\overline{Log}$ differ by exactly one constraint. That is, $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ uses the constraint $\sum_{j=1}^{q} -b^2_{kj}x_j \geq -u_k$, while $\overline{Log}$ uses $\sum_{j=1}^{q} \hat{b}^2_{kj}x_j \leq u_k$ which is equivalent to $\sum_{j=1}^{q} -\hat{b}^2_{kj}x_j \geq -u_k$ by multiplying $\sum_{j=1}^{q} \hat{b}^2_{kj}x_j \leq u_k$ by -1. Now, consider any feasible $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}})$ to $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ which has the constraint $\sum_{j=1}^{q} -b^2_{kj}\hat{x}_j \geq -\hat{u}_k$. Note that $-\hat{u}_k \leq \sum_{j=1}^{q} -b^2_{kj}\hat{x}_j \leq \sum_{j=1}^{q-1} -b^2_{kj}\hat{x}_j = \sum_{j=1}^{q} -\hat{b}^2_{kj}\hat{x}_j$ since $\hat{b}^2_{kq} = 0$ and $\hat{b}^2_{kj} = b^2_{kj}$ for $j = 1, \dots, q-1$. Thus, $\sum_{j=1}^{q} -\hat{b}^2_{kj}\hat{x}_j \geq -\hat{u}_k$ or by multiplying it by -1 gives the constraint found in $\overline{Log}$, $\sum_{j=1}^{q} \hat{b}^2_{kj}\hat{x}_j \leq \hat{u}_k$. All other constraints of $\overline{Log}$ are the same as $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ so $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}}) \in \overline{Log}$ and thus $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2) \subseteq \overline{Log}$. In addition, the solution $x_q = 1$, $x_i = 0$ for $i \neq q$, and $\boldsymbol{u} = \boldsymbol{v}_{q+1}$ is feasible to $\overline{Log}$, but not to $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ so $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2) \subset \overline{Log}$.

Case 2

Now, let $v_{kq} = 0$ and $v_{k,(q+1)} = 1$, then, by definition, $b^1_{kq} = 0$, since $v_{kq} = 0$, but $\hat{b}^1_{kq} = 1$, because $v_{k,(q+1)} = 1$. In addition, $b^1_{ij} = \hat{b}^1_{ij}$ for all $i = 1, \dots, \lceil \log_2(q-1) \rceil$, and $j = 1, \dots, q$ with $i \neq k$ and $j \neq q$ and $b^1_{ij} = \hat{b}^1_{ij}$ for all $i = 1, \dots, \lceil \log_2(q-1) \rceil$, and $j = 1, \dots, q$. Again, the formulations $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ and $\overline{Log}$ differ by exactly one constraint. That is, $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ uses the constraint $\sum_{j=1}^{q} b^1_{kj}x_j \geq u_k$, while $\overline{Log}$ uses $\sum_{j=1}^{q}(1 - \hat{b}^1_{kj})x_j \leq 1 - u_k$ which is equivalent to $\sum_{j=1}^{q} \hat{b}^1_{kj}x_j \geq u_k$ by taking $\sum_{j=1}^{q}(1 - \hat{b}^1_{kj})x_j \leq 1 - u_k$ and multiplying it by -1 and then adding $\sum_{j=1}^{q} x_j = 1$ to it. Now, consider any feasible $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}})$ to $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ which has the constraint $\sum_{j=1}^{q} b^1_{kj}\hat{x}_j \geq \hat{u}_k$. Note that $\hat{u}_k \leq \sum_{j=1}^{q} b^1_{kj}\hat{x}_j = \sum_{j=1}^{q-1} b^1_{kj}\hat{x}_j \leq \sum_{j=1}^{q} \hat{b}_{kj}\hat{x}_j$ since $b^1_{kq} = 0$, $\hat{b}^1_{kq} = 1$, and $\hat{b}^1_{kj} = b^1_{kj}$ for $j = 1, \dots, q-1$. Thus, $\sum_{j=1}^{q} \hat{b}^1_{kj}\hat{x}_j \geq \hat{u}_k$ or by multiplying this constraint by -1 and adding the constraint $\sum_{j=1}^{q} x_j = 1$ gives the constraint in $\overline{Log}$, $\sum_{j=1}^{q}(1 - \hat{b}^1_{kj})\hat{x}_j \leq 1 - \hat{u}_k$. All of the other constraints of $\overline{Log}$ are the same as $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ so $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}}) \in \overline{Log}$ and thus $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2) \subseteq \overline{Log}$. In addition, the solution $x_q = 1$, $x_i = 0$ for $i \neq q$, and $\boldsymbol{u} = \boldsymbol{v}_{q+1}$ is feasible to Log, but not to $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ so $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2) \subset \overline{Log}$. □

The next example illustrates the strict containment of the set $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ within $\overline{Log}$ when $q = 4$ and $q^* = 5$ resulting in $q < q^*$.

<u>Example 6</u>

Let $q = 4$, then the formulation $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ with the compatible vectors, $\boldsymbol{v}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $\boldsymbol{v}_2 =$

$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $\boldsymbol{v}_3 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, and $\boldsymbol{v}_4 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, is

$$\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2) = \left\{ (\boldsymbol{x},\boldsymbol{u}) : \begin{array}{c} x_1 + x_2 + x_3 + x_4 = 1 \\ \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \geq \begin{bmatrix} u_1 \\ u_2 \\ -u_1 \\ -u_2 \end{bmatrix} \end{array}, \ \boldsymbol{x} \geq \boldsymbol{0}, \ \boldsymbol{0} \leq \boldsymbol{u} \leq \boldsymbol{1} \right\}.$$

We rewrite $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ in a similar form to $\overline{Log}$ by multiplying the four inequality constraints by -1 and then add $x_1 + x_2 + x_3 + x_4 = 1$, to the first two constraints which gives the formulation

$$\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2) = \left\{ (\boldsymbol{x},\boldsymbol{u}) : \begin{array}{c} x_1 + x_2 + x_3 + x_4 = 1 \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leq \begin{bmatrix} 1-u_1 \\ 1-u_2 \\ u_1 \\ u_2 \end{bmatrix} \end{array}, \ \boldsymbol{x} \geq \boldsymbol{0}, \ \boldsymbol{0} \leq \boldsymbol{u} \leq \boldsymbol{1} \right\}.$$

The formulation for $\overline{Log}$ using the same $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{v}_4$ as $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ is

$$\overline{Log} = \left\{ (\boldsymbol{x},\boldsymbol{u}) : \begin{array}{c} x_1 + x_2 + x_3 + x_4 = 1 \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \leq \begin{bmatrix} 1-u_1 \\ 1-u_2 \\ u_1 \\ u_2 \end{bmatrix} \end{array}, \ \boldsymbol{x} \geq \boldsymbol{0}, \ \boldsymbol{0} \leq \boldsymbol{u} \leq \boldsymbol{1} \right\}.$$

Observe that the binary realization of the vector $\hat{\boldsymbol{u}} = \boldsymbol{v}_4 = (0,1)^T$ is feasible to $\overline{Log}$ and gives the point $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}}) = (\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{x}_4, \hat{u}_1, \hat{u}_2) = (0,0,0,1,0,1) \in \overline{Log}$. Now consider the point $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}})$ for $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$. The realization $\hat{\boldsymbol{u}} = (0,1)$ in $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ is not feasible because it forces $x_1 = x_2 = x_3 = x_4 = 0$ with $x_1 + x_2 + x_3 + x_4 = 1$ so $(\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{x}_4, \hat{u}_1, \hat{u}_2) = (0,0,0,1,0,1) \notin \overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$. The theorem above demonstrated that all other feasible points $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}})$ to $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ also have $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}}) \in \overline{Log}$ so $\overline{\text{Proj}}_{(\boldsymbol{x},\boldsymbol{u})}(X_2) \subset \overline{Log}$ when $q < q^*$.

To conclude the comparisons of the various models we see that the size of the formulations adapted to modeling a single piecewise-linear function are given in Table 3.1. A similar reduction to formulation *DLog* (as is done in Section 4 to X_2) to remove the $\boldsymbol{x}$ variables using (3.17)–(3.19) is performed again. In addition, the removal of the variables y and $f(y)$ and their associated constraints is done when these piecewise-linear functions are used within a larger optimization problem (as is demonstrated in Section 5). To summarize, all formulations use the same logarithmic number of binary variables $\boldsymbol{u}$, but the number of constraints and continuous variables required to enforce the SOS-2 restrictions vary. The formulation X_2 uses roughly half the number of constraints than the formulations *Log*, $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$, and *DLog*, but requires more variables. In addition, formulation *Log* and $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ require $q+2$ continuous variables while formulations X_2 and *DLog* require almost twice as many, specifically $2(q-1)+2$.

Model	Constraints	Variables	Binaries
X_2	$3+\lceil\log_2(q-1)\rceil$	$2(q-1)+2$	$\lceil\log_2(q-1)\rceil$
$\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$	$3+2\lceil\log_2(q-1)\rceil$	$q+2$	$\lceil\log_2(q-1)\rceil$
Log	$3+2\lceil\log_2(q-1)\rceil$	$q+2$	$\lceil\log_2(q-1)\rceil$
DLog	$3+2\lceil\log_2(q-1)\rceil$	$2(q-1)+2$	$\lceil\log_2(q-1)\rceil$

Table 3.1: Size of formulations for a single piecewise-linear function.

3.5 Computational Experience

The formulations using a logarithmic number of additional binary variables and constraints for SOS-2 restrictions modeling piecewise-linear functions presented in Subsections 3.2, 3.3, and 4.1 are tested computationally using similar problems in structure to the transportation problems found in [6] which were first formulated in [4]. These balanced transportation problems with 10 supply nodes and 10 demand nodes take the form

$$\min \sum_{i=1}^{10}\sum_{j=1}^{10} f_{ij}(y_{ij}) \text{ s.t. } \boldsymbol{y} \in \boldsymbol{Y}, \tag{3.44}$$

where $\boldsymbol{Y}$ is the feasible set for a 10×10 balanced transportation problem given as

$$\boldsymbol{Y} = \left\{ \boldsymbol{y} \in \mathbb{R}^{10} \times \mathbb{R}^{10} : \sum_{i=1}^{10} y_{ij} = d_j,\ j = 1,\ldots,10,\ \sum_{j=1}^{10} y_{ij} = s_i,\ i = 1,\ldots,10,\ \boldsymbol{y} \geq \boldsymbol{0} \right\}, \tag{3.45}$$

with supply nodes s_i, $i = 1, \ldots, 10$, and demand nodes d_j, $j = 1, \ldots, 10$. The objective function is the sum over continuous concave piecewise-linear functions where f_{ij} is the cost associated with arc y_{ij}. Each arc y_{ij} and cost function f_{ij} for $i, j = 1, \ldots, 10$, are partitioned into $q - 1$ segments with break points $\theta_1^{ij}, \ldots, \theta_q^{ij}$. Using (3.37) for each individual piecewise-linear function and adding the nonnegative variables $\boldsymbol{x}$ gives the formulation of the transportation problem as

$$\min \sum_{i=1}^{10} \sum_{j=1}^{10} \sum_{k=1}^{q} f_{ij}(\theta_k^{ij}) x_k^{ij} \text{ s.t. } \boldsymbol{x} \in \boldsymbol{X}, \tag{3.46}$$

with

$$\boldsymbol{X} = \left\{ \boldsymbol{x} \in \mathbb{R}^{10} \times \mathbb{R}^{10} \times \mathbb{R}^{q} : \begin{array}{l} \sum_{i=1}^{10} \sum_{k=1}^{q} \theta_k^{ij} x_k^{ij} = d_j, \; j = 1, \ldots, 10, \\ \sum_{j=1}^{10} \sum_{k=1}^{q} \theta_k^{ij} x_k^{ij} = s_i, \; i = 1, \ldots, 10, \\ \sum_{k=1}^{q} x_k^{ij} = 1, \; i, j = 1, \ldots, 10, \; \boldsymbol{x} \geq \boldsymbol{0}, \\ \text{SOS-2 on } x_1^{ij}, \ldots, x_q^{ij}, \forall\, i = 1, \ldots, 10, \; j = 1, \ldots, 10 \end{array} \right\}, \tag{3.47}$$

where the variables y_{ij} and $f_{ij}(y_{ij})$ are substituted out to reduce the size of the problem. As shown previously, these SOS-2 restrictions can be provided by X_2 of (3.16)–(3.21), $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ of (3.36), *Log* of (3.42), or *DLog* of (3.43).

For the computational experiments, the supply and demand nodes were initially set as uniform integer random variables between 1 and 21 inclusively. They were then adjusted so that the transportation problem was balanced by randomly selecting either a supply or a demand node and increasing or decreasing it by 1 depending on whether the gap between the total supply and total demand decreased (nodes could only be adjusted if their value remained between 1 and 21 inclusively). This was done until the total supply equaled the total demand. Each arc y_{ij} in the transportation problem was bounded on the interval $[0, \min\{s_i, d_j\}]$ to create a bounded set for the partitioning of the arc costs meaning $\theta_q^{ij} = \min\{s_i, d_j\}$ for all i and j. The intervals $[0, \theta_q^{ij}]$ were randomly partitioned into four sections and these sections were evenly divided until the desired number of $q - 1$ segments was achieved.

The piecewise-linear functions had $q - 1$ uniform random slopes found by selecting some $a \in \{1, \ldots, 2000\}$ and defining the slope to be $\frac{a}{2000}$. The $q - 1$ slopes were arranged in nonincreasing order to give the desired concavity. Each concave piecewise-linear function begins at the origin

$(f_{ij}(\theta_1^{ij}) = 0$ with $\theta_1^{ij} = 0$ for all i and j) and increases until $\theta_q^{ij} = \min\{s_i, d_j\}$. An example is shown in Figure 3.2 with $q = 5$ and 4 randomly generated slopes.

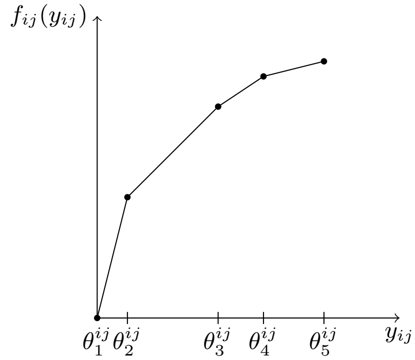


Figure 3.2: Example of the cost function f_{ij} for arc y_{ij} with $q = 5$.

We investigate the computational time required for instances of X_2, $\mathrm{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$, *Log*, and *DLog*. Five individual 10×10 transportation problems as described above were created and then 20 different objective functions were formed for each individual transportation problem. A total of 100 problem instances where generated for the cases $q = 13, 17, 25, 33$. The number of binary variables, continuous variables, and constraints for a single transportation problem are given in Table 3.2. All problems were formulated in AMPL and solved using using ILOG CPLEX 10.0 on a Sun V440 workstation with 16 GB of RAM and four 1.6 GHz CPU's running Solaris 10.

$q = 13$	X_2	$\mathrm{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$	*Log*	*DLog*
Bin. Var.	400	400	400	400
Cont. Var.	2400	1300	1300	2400
Const.	520	920	920	920

$q = 17$	X_2	$\mathrm{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$	*Log*	*DLog*
Bin. Var.	400	400	400	400
Cont. Var.	3200	1700	1700	3200
Const.	520	920	920	920

$q = 25$	X_2	$\mathrm{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$	*Log*	*DLog*
Bin. Var.	500	500	500	500
Cont. Var.	4800	2500	2500	4800
Const.	620	1120	1120	1120

$q = 33$	X_2	$\mathrm{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$	*Log*	*DLog*
Bin. Var.	500	500	500	500
Cont. Var.	6400	3300	3300	6400
Const.	620	1120	1120	1120

Table 3.2: Size of a single transportation problem.

The average time (in CPU seconds) of the 100 transportation problem tested are displayed in Table 3.3 for all cases. (A complete list of all computational result are found in the appendix.).

The first observation we make is that X_2, $\mathrm{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$, and *Log*, outperformed *DLog*, on average, for each transportation problem and for every instance of q with the exception of *Log* for

$q = 13$	Average Time (CPU sec.)			
Problem	X_2	$\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$	*Log*	*DLog*
Trans A	350.80	254.10	708.70	701.27
Trans B	25.85	19.75	25.03	47.93
Trans C	17.18	20.53	21.33	34.51
Trans D	122.75	140.22	152.83	231.37
Trans E	15.69	15.70	19.08	32.33
Average	106.45	90.06	185.39	209.48

$q = 17$	Average Time (CPU sec.)		
Problem	X_2	$\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)/Log$	*DLog*
Trans A	295.94	223.14	485.80
Trans B	43.02	23.11	73.89
Trans C	42.61	28.15	73.96
Trans D	238.84	217.48	436.50
Trans E	24.51	17.43	46.66
Average	128.99	101.86	223.36

$q = 25$	Average Time (CPU sec.)			
Problem	X_2	$\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$	*Log*	*DLog*
Trans A	1788.90	1417.96	2520.20	2668.94
Trans B	83.39	56.58	74.90	139.83
Trans C	50.90	50.87	56.35	95.15
Trans D	551.44	541.48	394.02	1031.96
Trans E	49.60	51.78	58.71	99.55
Average	504.85	423.74	620.83	807.09

$q = 33$	Average Time (CPU sec.)		
Problem	X_2	$\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)/Log$	*DLog*
Trans A	948.24	1623.58	3129.62
Trans B	49.09	47.34	122.73
Trans C	58.93	55.97	156.67
Trans D	451.16	387.06	1406.26
Trans E	45.46	50.05	129.70
Average	310.58	432.80	988.99

Table 3.3: Average computational times for $q = 13, 17, 25, 33$.

Transportation Problem A when $q = 13$. For this instance, *Log* was only, on average, about 1% slower than *DLog*. Next, formulation X_2 outpreformed *DLog*, which was expected, because both formulations have the same number of variables but *DLog* requires $1 + 2\lceil \log_2(q-1) \rceil$ inequality constraints for each piecewise-linear function while X_2 only needs $1 + \lceil \log_2(q-1) \rceil$ equality constraints. In addition, formulations $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ and *Log* did better on average, beside the exception already mentioned, than *DLog* which is expected because all three formulations have the same number of constraints, but $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ and *Log* required q continuous variables for each piecewise-linear function while *DLog* needs $2(q-1)$. The additional variables for *DLog* result in longer computation times. Also, we note that the computational trials of [6] comparing *Log* and *DLog* demonstrated that *Log* outperformed *DLog*, on average, for all values of q tested and our trials demonstate a similar result.

The most interesting results occurred between X_2, $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$, and *Log*. First consider the cases $q = 13, 25$. Recall that formulation X_2 requires $1 + \lceil \log_2(q-1) \rceil$ constraints and $2(q-1)$ continuous variables for each piecewise-linear function while $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ and *Log* need $1 + 2\lceil \log_2(q-1) \rceil$ constraints but only q continuous variables. In addition, from Theorem 3, $Log \subset \text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ when $\log_2(q-1) < \lceil \log_2(q-1) \rceil$ which is the case for $q = 13, 25$. The differences between these two formulations offers insight into the computational results obtained. To begin with, X_2, on average, did better then *Log* in 7 out of the 10 transportation problems tested and this could be attributed to the fact that X_2 has more variables but fewer constraints than *Log*.

The formulation $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$, on average, outperformed *Log* in 9 out of the 10 transportation problems tests. This is to be expected because each formulation had the same number of constraints and variables, but $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ had one less binary realization of $\boldsymbol{u}$ than *Log* for each piecewise-linear function. Last, formulations X_2 and $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ demonstrated similar results since X_2 has fewer constraints while $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ has fewer variables. As a result, $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ did better, on average, than X_2 in only 6 out of 10 transportation problems tested. But, the results between these two problem were similar which suggest that more testing could be done to assess the validity of one formulation over the other.

Next, when $q = 17, 33$, recall that $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)$ and *Log* are the same formulation and, thus have the same computational times. We observe that $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)/Log$ is faster, on average, than X_2 in 8 out of 10 transportation problems which could have resulted from X_2 having $2(q-1)$ continuous variables for each piecewise-linear function while $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)/Log$ only requires q and X_2 using $1 + \lceil \log_2(q-1) \rceil$ equality constraints for each function while $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)/Log$ needs $1+2\lceil \log_2(q-1) \rceil$ inequality constraints. In this instance, it might be preferable to reduce the number of variables while increasing the number of constraints by using the formulation $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)/Log$ instead of X_2. We will mention that overall computational times were similar between X_2 and $\text{Proj}_{(\boldsymbol{x},\boldsymbol{u})}(X_2)/Log$, thus more testing could be done to assess the validity of formulations with fewer constraints and more variables versus formulations with more constraints and less variables.

3.6 Conclusions

This chapter presented a new approach for modeling disjunctions of polytopes using a logarithmic number of binary variables. Our form permits reductions in the problem via substitutions of continuous variables through suitable projections. Specifically, disjunctions modeling SOS-2 restrictions are formed and then reduced in size via projections. Such reductions result in models having only a logarithmic number of binary variables and constraints to represent the SOS-2 restrictions. We proved that our formulations are as least as tight as other leading SOS-2 representations in the literature. In special instances, our models are contained within these other representations. All formulations modeling SOS-2 restrictions are easily adapted to represent piecewise-linear functions. Such functions are useful in larger transportation networks with piecewise-linear objective functions.

Computational results demonstrated the merits of our models compared to other leading representations for piecewise-linear functions using a logarithmic number of constraints and binary variables. Specifically, our models, on average, outperformed the other formulations tested when the number of segments for each piecewise-liner functions was not a power of two. When the number of segments was a power of two, one of our formulations reduces to another found in the literature and the other model compared favorable to this representation and outperformed another from the literature. In the former instance, the model from the literature had more constraints, but a fewer number of continuous variables than our formulation. More computational testing could be done to assess the benefits of representations with fewer constraints versus models with fewer continuous variables.

Bibliography

[1] Adams, W.P. and Henry, S.M., "Base-2 Expansions for Linearizing Products of Functions of Discrete Variables," *Operations Research*, to appear, manuscript 2010.

[2] Balas, E., "Disjunctive Programming," *Annals of Discrete Mathematics*, Vol. 5, pp 3–51, 1979.

[3] Balas, E. "Disjunctive Programming and a Hierarchy of Relaxations for Discrete Optimization Problems," *SIAM Journal on Algebraic and Discrete Methods*, Vol. 6, No. 3, pp 466–486, 1985.

[4] Keha, A.B., Farias, I.R., and Nemhauser, G.L., "A Brand-and-Cut Algorithm Without Binary Variables for Nonconvex Piecewise linear Optimization," *Operations Research*, Vol. 54, No. 5, 847–858, 2006.

[5] Vielma, J.P. and Nemhauser, G.L., "Modeling Disjunctive Constraints with a Logarithmic Number of Binary Variables and Constraints," *Mathematical Programming, Series A*, DOI 10.1007/s10107-009-0295-4.

[6] Vielma, J.P., Ahmed, S., and Nemhauser, G., "Mixed-Integer Models for Nonseparable Piecewise-Linear Optimization: Unifying Framework and Extensions," *Operations Research*, Vol. 58, No. 2, 303–315, 2010.

Chapter 4

Ideal Representations of Lexicographic Orderings and Base-2 Expansions of Integer Variables

4.1 Introduction

Consider a discrete variable x having $\ell \leq x \leq u$, where ℓ and u denote integer lower and upper bounds on x respectively, so that there exist $n = u - \ell + 1$ permissible realizations. There are various ways to represent x in terms of binary variables. One approach is to define $n-1$ binary variables λ_j, $j = 1, \ldots, n-1$, and to enforce the set

$$P_1(x, \boldsymbol{\lambda}) \equiv \left\{ (x, \boldsymbol{\lambda}) \in \mathbb{R} \times \{0,1\}^{n-1} : x = \ell + \sum_{j=1}^{n-1} \lambda_j \right\}. \tag{4.1}$$

In this way, x will equal ℓ plus the number of variables λ_j fixed to 1. An alternative approach that employs these same variables is based on the set

$$P_2(x, \boldsymbol{\lambda}) \equiv \left\{ (x, \boldsymbol{\lambda}) \in \mathbb{R} \times \{0,1\}^{n-1} : x = \ell + \sum_{j=1}^{n-1} j\lambda_j,\ \sum_{j=1}^{n-1} \lambda_j \leq 1 \right\}. \tag{4.2}$$

These restrictions ensure that x will equal ℓ when all λ_j equal 0, and will equal $\ell + j$ when some single λ_j is equal to 1. A third approach constructs a base-2 expansion of x by enforcing the set

$$P_3(x, \boldsymbol{\lambda}) \equiv \left\{ (x, \boldsymbol{\lambda}) \in \mathbb{R} \times \{0,1\}^{\lceil \log_2 n \rceil} : x = \ell + \sum_{j=1}^{\lceil \log_2 n \rceil} 2^{\lceil \log_2 n \rceil - j} \lambda_j,\ x \leq u \right\}. \tag{4.3}$$

The expression $\sum_{j=1}^{\lceil \log_2 n \rceil} 2^{\lceil \log_2 n \rceil - j} \lambda_j$ for $\boldsymbol{\lambda} \in \{0,1\}^{\lceil \log_2 n \rceil}$ can realize any integer value between 0 and $2^{\lceil \log_2 n \rceil} - 1$ so that the identity in (4.3) enforces that x is an integer satisfying $\ell \leq x \leq \ell + 2^{\lceil \log_2 n \rceil} - 1$. The inequality $x \leq u$ serves to upper bound x at the value u, but is not needed when $\lceil \log_2 n \rceil = \log_2 n$, as $n = u - \ell + 1$ by definition.

An immediate distinction between the sets $P_1(x, \boldsymbol{\lambda})$ and $P_2(x, \boldsymbol{\lambda})$ of (4.1) and (4.2), respectively, and $P_3(x, \boldsymbol{\lambda})$ of (4.3) is that the first two use $n - 1$ binary variables, while the last uses only $\lceil \log_2 n \rceil$. These three sets, with minor variations, are prevalent throughout the operations research literature, appearing in such works as [2, 5, 7, 13, 15, 18, 20, 22]. In particular, the expansion of (4.3) arises in various contexts. The "all different polytope" of [10] is defined in terms of $m \times n$ matrices, where the rows of each matrix serve as base-2 expansions. Here, it is desired to select m elements, order important, from amongst a collection of 2^n, with row i of a matrix denoting the base-2 expansion of the *i-th* element selected. Each matrix represents a different overall selection of m elements, and the rows are restricted to be distinct so that each element is selected at most once. Motivated by this work, the paper [4] studies cases where the number of elements within the collection need not be a power of 2. Given an n-dimensional hypercube, and an integer $k \leq 2^n$, a task of this latter paper is to select k vertices so that a minimal number of linear inequalities is needed to define the convex hull. Special "cropping inequalities" are employed. By numbering the vertices of the hypercube in terms of their base-2 expansions, the lexicographic orderings allow us to select the set of vertices numbered 0 through $k - 1$ using m_0 readily-defined minimal cover inequalities, in addition to the trivial bounding inequalities, where m_0 represents the number of entries of value 0 in the binary expansion of $k - 1$. Extensions to the all different polytope are found in [11, 12] relative to an edge coloring problem on a graph, where row i of an $m \times n$ matrix represents the base-2 expansion of that color number assigned to edge i. In this way, m equals the number of edges and $n = \lceil \log_2 c \rceil$, where c is the number of colors. This problem differs from [10] in that two rows of the matrix need not be distinct if the corresponding edges do not share a vertex.

Within an optimization setting, the idea behind (4.1), (4.2), and (4.3) is to convert integer programs to binary problems, which are sometimes simpler to solve. It is well-known that a critical concern in approaching discrete optimization problems is the strength of the continuous relaxation. Generally speaking, tighter relaxations are preferable. Thus, it is prudent to consider the respective strengths of (4.1)–(4.3) when constructing a conversion.

Let $\overline{P}_1(x, \boldsymbol{\lambda})$, $\overline{P}_2(x, \boldsymbol{\lambda})$, and $\overline{P}_3(x, \boldsymbol{\lambda})$ denote the continuous relaxations of the sets $P_1(x, \boldsymbol{\lambda})$, $P_2(x, \boldsymbol{\lambda})$, and $P_3(x, \boldsymbol{\lambda})$, respectively, obtained by replacing the $\boldsymbol{\lambda} \in \{0,1\}^{n-1}$ restrictions with $\boldsymbol{\lambda} \in [0,1]^{n-1}$ within $P_1(x, \boldsymbol{\lambda})$, the $\boldsymbol{\lambda} \in \{0,1\}^{n-1}$ restrictions with $\boldsymbol{\lambda} \geq \mathbf{0}$ within $P_2(x, \boldsymbol{\lambda})$, and the $\boldsymbol{\lambda} \in \{0,1\}^{\lceil \log_2 n \rceil}$ restrictions with $\boldsymbol{\lambda} \in [0,1]^{\lceil \log_2 n \rceil}$ within $P_3(x, \boldsymbol{\lambda})$. It is simple to show that the sets $P_1(x, \boldsymbol{\lambda})$ and $P_2(x, \boldsymbol{\lambda})$ are *ideal* in that $\text{conv}(P_1(x, \boldsymbol{\lambda})) = \overline{P}_1(x, \boldsymbol{\lambda})$ and $\text{conv}(P_2(x, \boldsymbol{\lambda})) = \overline{P}_2(x, \boldsymbol{\lambda})$, where $\text{conv}(\bullet)$ denotes the convex hull of the set $\bullet$. It is also known [1] that $P_3(x, \boldsymbol{\lambda})$ is ideal when $\lceil \log_2 n \rceil = \log_2 n$, but [1] gives an example showing $\text{conv}(P_3(x, \boldsymbol{\lambda})) \subset \overline{P}_3(x, \boldsymbol{\lambda})$ for a specific instance having $\lceil \log_2 n \rceil > \log_2 n$. An emphasis of this chapter is to show that (4.3) can be made ideal by appending at most $\lceil \log_2 n \rceil - 1$ minimal cover inequalities. For the simple special case in which $\lceil \log_2 n \rceil = \log_2 n$ so that the convex hull is known, no new inequalities are needed.

The convex hull argument for $P_3(x, \boldsymbol{\lambda})$ is motivated by a lexicographic ordering on vectors of binary variables. In the spirit of [8], given a nonzero vector $\boldsymbol{\alpha} \in \{0,1\}^m$ for some positive integer m, Section 2 explains that the convex hull of the set of vectors in $\{0,1\}^m$ that is lexicographically less than or equal to $\boldsymbol{\alpha}$ can be characterized in terms of m_0 minimal cover inequalities in m variables, where m_0 denotes the number of entries of value 0 in $\boldsymbol{\alpha}$. This section then relates lexicographic orderings and base-2 expansions to obtain $\text{conv}(P_3(x, \boldsymbol{\lambda}))$.

Extensions of convex hull forms of lexicographic orderings and their applications to knapsack polytopes are presented in Sections 3 and 4, respectively. Section 3 generalizes our results on lexicographic orderings in two ways. First, it shows that the convex hull of all binary vectors lexicographically greater than or equal to a vector $\boldsymbol{\alpha} \in \{0,1\}^m$ can be obtained in an analogous fashion as to when $\boldsymbol{\alpha}$ serves as an upper bound. Second, it combines these lower and upper bounding results to provide an explicit description for the convex hull of binary vectors that are restricted to lexicographically lie between two binary vectors. This description employs both minimal cover and set covering inequalities and is somewhat unexpected, as the intersection of two integral polytopes is not necessarily an integral polytope. Section 4 then considers specially-structured 0-1 knapsack problems whose constraint coefficients are weakly super-decreasing. Earlier work of [9], using results of [19], showed that the set of minimal cover inequalities for a 0-1 knapsack problem defines the convex hull of solutions if and only if the problem has weakly super-decreasing (equivalently, weakly super-increasing) coefficients. The convex hull proof was later simplified by [6]. Our lexicographic orderings naturally extend to weakly super-decreasing coefficients, and so we are able to generalize the result of [6, 9] to 0-1 knapsack polytopes having weakly super-decreasing coefficients where both lower and upper bounds are enforced on the knapsack constraint. (Approximately two months after the completion of this chapter, the article [3] appeared on *Optimization Online*; this article provides an alternate proof of the convex hull form for two-sided knapsack sets having weakly super-increasing coefficients. This work, motivated by a problem of efficiently representing the convex hull of an arbitrary set of vertices of the unit hypercube, uses a different method of proof that relies on an inductive argument based on an extended formulation obtained via disjunctive programming.)

The chapter continues by presenting some preliminary computational experience in Section 5 to demonstrate the usefulness of incorporating $\text{conv}(P_3(x, \boldsymbol{\lambda}))$ within a base-2 expansion of integer variables, and ends with a conclusions section.

4.2 Minimal Cover Description of Bounded Integer Variables

As mentioned earlier, our modeling of the convex hull of the set $P_3(x, \boldsymbol{\lambda})$ of (4.3) is based on a minimal cover description of a lexicographic ordering of binary vectors. This description relates and combines published works, but differs in that the motivation stems from the representation of integer variables. The paper [8] gives a thorough study of the facial structure of the convex hull of the set of binary vectors that is lexicographically upper bounded by a given such vector. Though motivated from a different perspective than [8], the minimal cover inequalities we obtain include all the nontrivial facets. Our approach extends, in Section 3, this work of [8] to binary vectors that are lexicographically bounded from *both* below and above by including set covering restrictions. Section 4 uses our lexicographic results to extend the convex hull representations of [6, 9] for special knapsack problems having weakly super-decreasing coefficients to include instances having lower and upper bounds on the structural constraint. In the process, this latter section relates [8] to such problems.

Recall that a vector $\boldsymbol{y}$ is lexicographically nonpositive, denoted $\boldsymbol{y} \preceq \mathbf{0}$, if either $\boldsymbol{y} = \mathbf{0}$ or the first nonzero entry is negative. Also recall that, given two vectors $\boldsymbol{y}^1$ and $\boldsymbol{y}^2$, the vector $\boldsymbol{y}^1$ is lexicographically less than or equal to the vector $\boldsymbol{y}^2$, denoted $\boldsymbol{y}^1 \preceq \boldsymbol{y}^2$, if $\boldsymbol{y}^1 - \boldsymbol{y}^2 \preceq \mathbf{0}$. Now, consider a vector $\boldsymbol{\alpha} \in \{0,1\}^m$ for some integer $m \geq 2$ and the set of $\boldsymbol{y} \in \{0,1\}^m$ satisfying $\boldsymbol{y} \preceq \boldsymbol{\alpha}$. It turns out that m_0 minimal cover inequalities in $\boldsymbol{y}$, together with the restrictions $\boldsymbol{y} \in [0,1]^m$, are sufficient to define the convex hull where, consistent with earlier use, m_0 represents the number of entries of value 0 in $\boldsymbol{\alpha}$. For convenience, we henceforth denote the *i-th* entries of $\boldsymbol{\alpha}$ and $\boldsymbol{y}$ by α_i and y_i respectively, and assume without loss of generality that $\alpha_1 = 1$ since otherwise all $\boldsymbol{y}$ satisfying $\boldsymbol{y} \preceq \boldsymbol{\alpha}$ must have $y_1 = 0$, and similarly assume that $\alpha_m = 0$ since otherwise y_m can realize a value of either 0 or 1 without restriction.

To motivate the convex hull description, based on the vector $\boldsymbol{\alpha}$, partition the set $M \equiv \{1,\ldots,m\}$ into the two subsets $M_0 = \{i \in M : \alpha_i = 0\}$ and $M_1 = \{i \in M : \alpha_i = 1\}$, noting from above that $1 \in M_1$ and $m \in M_0$. Let $m_0 = |M_0|$ (as used earlier) and $m_1 = |M_1|$ denote the cardinalities of the sets M_0 and M_1, respectively, so that $m_0 + m_1 = m$. The approach follows from the property of lexicographic orderings that a vector $\boldsymbol{y} \in \{0,1\}^m$ is such that $\boldsymbol{y} \preceq \boldsymbol{\alpha}$ if and only if, for each $i \in M_0$ such that $y_i = 1$ (if any), there exists some $j \in M_1$, $j < i$, with $y_j = 0$. This observation is summarized below.

Observation

Given a vector $\boldsymbol{\alpha} \in \{0,1\}^m$ and the sets M, M_0, and M_1 defined in terms of m and $\boldsymbol{\alpha}$ as above, a binary vector $\boldsymbol{y}$ is such that $\boldsymbol{y} \preceq \boldsymbol{\alpha}$ if and only if the following m_0 inequalities are satisfied:

$$y_i \leq \sum_{\substack{j \in M_1 \\ j < i}} (1 - y_j) \;\; \forall\, i \in M_0.$$

Based on this observation, the set

$$S \equiv \left\{ \boldsymbol{y} \in \{0,1\}^m : y_i \leq \sum_{\substack{j \in M_1 \\ j < i}} (1 - y_j) \;\; \forall\, i \in M_0 \right\} \tag{4.4}$$

characterizes those vectors $\boldsymbol{y} \in \{0,1\}^m$ having $\boldsymbol{y} \preceq \boldsymbol{\alpha}$. Let $\overline{S}$ denote the continuous relaxation of S

obtained by replacing the $\boldsymbol{y} \in \{0,1\}^m$ restrictions of (4.4) with $\boldsymbol{y} \in [0,1]^m$.

As noted by [8], and later by [6], the inequalities of (4.4) possess the "interval matrix" or "consecutive ones" property (see [16, page 544, Definition 2.2] or earlier work by [21]). Note that each variable y_i having $i \in M_0$ appears in a single inequality while each variable y_i having $i \in M_1$ appears in those inequalities corresponding to $j \in M_0$ for which $j > i$. As a consequence, $\text{conv}(S) = \overline{S}$.

This minimal cover description of the convex hull of S allows us to write $\text{conv}(P_3(x, \boldsymbol{\lambda}))$. The connection between lexicographic orderings and base-2 expansions is the following. Given any two vectors $\boldsymbol{\alpha}, \boldsymbol{y} \in \{0,1\}^m$ for $m \geq 1$, we have

$$\boldsymbol{y} \preceq \boldsymbol{\alpha} \Longleftrightarrow \sum_{j=1}^{m} \gamma_j y_j \leq \sum_{j=1}^{m} \gamma_j \alpha_j, \tag{4.5}$$

where

$$\gamma_j = 2^{m-j} \quad \forall\, j = 1, \ldots, m. \tag{4.6}$$

To apply this connection, observe that the variable x of $P_3(x, \boldsymbol{\lambda})$ is allowed to realize any of the $n = u - \ell + 1$ integral values between ℓ and u. Consequently, by letting $m = \lceil \log_2 n \rceil$ in (4.3) and computing $\boldsymbol{\alpha} \in \{0,1\}^m$ so that $\sum_{j=1}^{m} 2^{m-j} \alpha_j = u - \ell$, the set $P_3(x, \boldsymbol{\lambda})$ of (4.3) can be rewritten as

$$P_3(x, \boldsymbol{y}) = \left\{ (x, \boldsymbol{y}) \in \mathbb{R} \times \{0,1\}^m : x = \ell + \sum_{j=1}^{m} 2^{m-j} y_j,\ \boldsymbol{y} \preceq \boldsymbol{\alpha} \right\}, \tag{4.7}$$

where we have substituted $\boldsymbol{y}$ for $\boldsymbol{\lambda}$ in (4.3) and replaced $x \leq u$ with $\boldsymbol{y} \preceq \boldsymbol{\alpha}$. But then

$$\text{conv}(P_3(x, \boldsymbol{y})) = \left\{ (x, \boldsymbol{y}) \in \mathbb{R} \times [0,1]^m : x = \ell + \sum_{j=1}^{m} 2^{m-j} y_j,\ \boldsymbol{y} \in \overline{S} \right\}. \tag{4.8}$$

Note in this construction that if $\lceil \log_2 n \rceil = \log_2 n$ in $P_3(x, \boldsymbol{\lambda})$ of (4.3), then $\boldsymbol{\alpha}$ would be a vector of ones, and no inequalities would be present in $\overline{S}$ of (4.8). This is to be expected since, as mentioned in Section 1, $\text{conv}(P_3(x, \boldsymbol{\lambda})) = \overline{P}_3(x, \boldsymbol{\lambda})$ for such values of n.

The example below demonstrates the construction of $\text{conv}(P_3(x, \boldsymbol{\lambda}))$ when $\lceil \log_2 n \rceil > \log_2 n$.

<u>Example</u>

Consider an integer variable x having $1 \leq x \leq 91$. Then $n = 91$ and $\lceil \log_2 91 \rceil = 7$ so that $P_3(x, \boldsymbol{\lambda})$

of (4.3), with $\boldsymbol{\lambda}$ replaced by $\boldsymbol{y}$, takes the form

$$P_3(x, \boldsymbol{y}) = \left\{ \begin{array}{l} (x, \boldsymbol{y}) \in \mathbb{R} \times \{0,1\}^7 : x \leq 91, \\ \qquad\qquad x = 1 + 64y_1 + 32y_2 + 16y_3 + 8y_4 + 4y_5 + 2y_6 + y_7 \end{array} \right\}.$$

We have $m = \lceil \log_2 91 \rceil = 7$ and $\boldsymbol{\alpha}^T = (1, 0, 1, 1, 0, 1, 0)$ because $u - \ell = 90 = 64 + 16 + 8 + 2$. Consequently, the $m_0 = 3$ minimal cover inequalities $y_1 + y_2 \leq 1$, $y_1 + y_3 + y_4 + y_5 \leq 3$, and $y_1 + y_3 + y_4 + y_6 + y_7 \leq 4$ describe $\overline{S}$, so that conv$(P_3(x, \boldsymbol{y}))$ of (4.8) is given by

$$\operatorname{conv}(P_3(x, \boldsymbol{y})) = \left\{ \begin{array}{l} (x, \boldsymbol{y}) \in \mathbb{R} \times [0,1]^7 : \\ \qquad x = 1 + 64y_1 + 32y_2 + 16y_3 + 8y_4 + 4y_5 + 2y_6 + y_7, \\ \qquad y_1 + y_2 \leq 1, \ y_1 + y_3 + y_4 + y_5 \leq 3, \\ \qquad y_1 + y_3 + y_4 + y_6 + y_7 \leq 4 \end{array} \right\}.$$

Observe from above that the point $(x, y_1, y_2, y_3, y_4, y_5, y_6, y_7) = (91, 1, \frac{13}{16}, 0, 0, 0, 0, 0)$ is feasible to the continuous relaxation of $P_3(x, \boldsymbol{y})$, but violates the inequality $y_1 + y_2 \leq 1$ of conv$(P_3(x, \boldsymbol{y}))$.

Before proceeding to the next section, we note that the Observation, together with the consequence that conv$(S) = \overline{S}$, directly relates to [4]. Given an n-dimensional hypercube and any $k \in \{1, \ldots, 2^n\}$, the set $\overline{S}$ explicitly defines the convex hull of that collection of k vertices whose base-2 expansions are less than or equal to $k-1$ by defining $\boldsymbol{\alpha} \in \{0,1\}^n$ so that $\sum_{j=1}^{n} 2^{n-j} \alpha_j = k-1$.

4.3 Lexicographic Extensions

Given an $\boldsymbol{\alpha} \in \{0,1\}^m$, Section 2 provided the convex hull of the set of vectors $\boldsymbol{y} \in \{0,1\}^m$ satisfying $\boldsymbol{y} \preceq \boldsymbol{\alpha}$. This result can be extended in two ways.

First, given such an $\boldsymbol{\alpha}$, a similar argument can be used to generate the convex hull of the set of vectors $\boldsymbol{y} \in \{0,1\}^m$ satisfying $\boldsymbol{y} \succeq \boldsymbol{\alpha}$ by the relationship

$$\boldsymbol{y} \succeq \boldsymbol{\alpha} \Longleftrightarrow \mathbf{1} - \boldsymbol{y} \preceq \mathbf{1} - \boldsymbol{\alpha}.$$

Using the same definitions of M_0 and M_1, we have that the set covering inequalities of

$$Q \equiv \left\{ \boldsymbol{y} \in \{0,1\}^m : 1 - y_i \leq \sum_{\substack{j \in M_0 \\ j < i}} y_j \;\; \forall\, i \in M_1 \right\}$$

characterize the vectors $\boldsymbol{y} \in \{0,1\}^m$ satisfying $\boldsymbol{y} \succeq \boldsymbol{\alpha}$, and that $\overline{Q}$, the continuous relaxation of the set Q obtained by relaxing $\boldsymbol{y} \in \{0,1\}^m$ to $\boldsymbol{y} \in [0,1]^m$, gives $\text{conv}(Q) = \overline{Q}$. Here, to potentially reduce problem size, we can assume without loss of generality that $\alpha_1 = 0$ since otherwise y_1 must equal 1, and that $\alpha_m = 1$ since otherwise y_m would be unrestricted to realize value of either 0 or 1.

The second extension is, given two vectors $\boldsymbol{\alpha}^1, \boldsymbol{\alpha}^2 \in \{0,1\}^m$ with $\boldsymbol{\alpha}^1 \preceq \boldsymbol{\alpha}^2$, the construction of the convex hull of the set

$$T \equiv \left\{ \boldsymbol{y} \in \{0,1\}^m : \boldsymbol{\alpha}^1 \preceq \boldsymbol{y} \preceq \boldsymbol{\alpha}^2 \right\}, \tag{4.9}$$

or equivalently, from Q and S, the convex hull of the set

$$T = \left\{ \boldsymbol{y} \in \{0,1\}^m : 1 - y_i \leq \sum_{\substack{j \in M_0^1 \\ j < i}} y_j \;\; \forall\, i \in M_1^1, \;\; y_i \leq \sum_{\substack{j \in M_1^2 \\ j < i}} (1 - y_j) \;\; \forall\, i \in M_0^2 \right\}, \tag{4.10}$$

where $M \equiv \{1, \ldots, m\}$ is partitioned in terms of $\boldsymbol{\alpha}^1$ into the subsets $M_0^1 = \{i \in M : \alpha_i^1 = 0\}$ and $M_1^1 = \{i \in M : \alpha_i^1 = 1\}$, and again partitioned in terms of $\boldsymbol{\alpha}^2$ into the subsets $M_0^2 = \{i \in M : \alpha_i^2 = 0\}$ and $M_1^2 = \{i \in M : \alpha_i^2 = 1\}$. Here, α_i^1 and α_i^2 denote, respectively, the *i-th* entries of $\boldsymbol{\alpha}^1$ and $\boldsymbol{\alpha}^2$ for each $i \in M$, and we assume without loss of generality that $\alpha_1^1 = 0$ and $\alpha_1^2 = 1$ so that $1 \in M_0^1 \cap M_1^2$.

The result below establishes that the continuous relaxation of T yields the convex hull. This description is unexpected, as the intersection of two integral polytopes does not generally yield an integral polytope.

Theorem 1

Let T be the set of binary vectors defined by (4.10) and let $\overline{T}$ denote the continuous relaxation of T obtained by relaxing the $\boldsymbol{y} \in \{0,1\}^m$ restrictions to $\boldsymbol{y} \in [0,1]^m$. Then $\text{conv}(T) = \overline{T}$.

Proof

Let $\boldsymbol{\alpha}^1, \boldsymbol{\alpha}^2 \in \{0,1\}^m$ with $\boldsymbol{\alpha}^1 \preceq \boldsymbol{\alpha}^2$ and $\alpha_1^1 = 0$ and $\alpha_1^2 = 1$, and let $\hat{\boldsymbol{y}}$ be any extreme point of $\overline{T}$. It is sufficient to show that $\hat{\boldsymbol{y}} \in \{0,1\}^m$. Toward this end, define $S_1 \subseteq M_1^1$ and $S_2 \subseteq M_0^2$ as the index sets of left-hand and right-hand inequalities respectively of (4.10) that are satisfied with equality at $\hat{\boldsymbol{y}}$. If either $S_1 = \emptyset$ or $S_2 = \emptyset$, then $\hat{\boldsymbol{y}} \in \{0,1\}^m$, as each set individually, with the restrictions $\boldsymbol{y} \in [0,1]^m$, has binary extreme points. Otherwise, express the inequalities holding at equality in matrix form, with the left-hand restrictions followed by the right-hand, and using an ordering of the variables according to the set in which each index is contained. The four possibilities are $i \in S_1 \backslash S_2$, $i \in S_2 \backslash S_1$, $i \in S_1 \cap S_2$, and $i \notin S_1 \cup S_2$, which give the four sets of columns in the following system of $|S_1|+|S_2|$ equations in m variables, where some variables y_i with $i \notin S_1 \cup S_2$ may have all coefficients of 0. Observe that $1 \notin S_1 \cup S_2$.

$$\begin{bmatrix} \bar{I}_1 & \bar{A} & \bar{I}_2 & \bar{B} \\ \tilde{A} & \tilde{I}_1 & \tilde{I}_2 & \tilde{B} \end{bmatrix} \begin{bmatrix} \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \mathbf{1} \\ \boldsymbol{d} \end{bmatrix}. \tag{4.11}$$

Here, $\bar{I}_1$ and $\bar{I}_2$ are obtained through a possible reordering and partitioning of the columns of the $|S_1| \times |S_1|$ identity matrix to reflect the variables y_i for which $i \in S_1$. Similarly, $\tilde{I}_1$ and $\tilde{I}_2$ are obtained through a possible reordering and partitioning of the columns of the $|S_2| \times |S_2|$ identity matrix to represent the variables y_i for which $i \in S_2$. The matrices $\bar{A}$, $\tilde{A}$, $\bar{B}$, and $\tilde{B}$ have all binary entries, and have the additional property that whenever a 1 appears in some row, the value 1 is repeated down the column through the last row of the matrix. The notation $\mathbf{1}$ denotes the column vector of size $|S_1|$ having all entries of 1 and $\boldsymbol{d}$ denotes an integral column vector of size $|S_2|$ representing the constants in the associated right-hand side restrictions of (4.10).

The proof proceeds in two parts. First, it shows that each variable y_q with $q \in S_1 \cup S_2$ either can be identified as having $\hat{y}_q$ binary, or the system (4.11) can be equivalently rewritten so that the associated column in $\bar{A}$, $\tilde{A}$, or $\tilde{I}_2$ consists of all zeros. The rewriting consists of removing select redundant constraints, while preserving all variables. Second, it shows that upon substituting those elements $\hat{y}_j$ of $\hat{\boldsymbol{y}}$ into (4.11) for which $\hat{y}_j$ has been identified at a binary value, the reduced system over the remaining entries of $\boldsymbol{y}$ has all binary extreme points.

Begin by defining F_0 and F_1, respectively, to be the index sets of those $\hat{y}_j$ that are currently identified as being at value 0 and 1, initialized as $F_0 = F_1 = \emptyset$. Now, consider any entry of value 1 (if it exists) within $\bar{A}$ appearing in some column corresponding to a variable y_q. Then, by definition, $q \in S_2 \cap M_0^1$ and there exists a $p \in S_1$ such that $q < p$. We thus have that

$$1 - y_p = \sum_{\substack{j \in M_0^1 \\ j < p}} y_j \quad \text{and} \quad y_q = \sum_{\substack{j \in M_1^2 \\ j < q}} (1 - y_j), \tag{4.12}$$

with the left equation yielding the selected coefficient 1 in $\bar{A}$. Substitute y_q from the right equation of (4.12) into the left (as $q \in M_0^1$ with $q < p$) to obtain, using $1 \in M_0^1 \cap M_1^2$, that

$$0 = \sum_{\substack{j \in M_0^1 \\ 1 < j < p, j \neq q}} y_j + \sum_{\substack{j \in M_1^2 \\ 1 < j < q}} (1 - y_j) + y_p.$$

Then $\boldsymbol{y} \in [0,1]^m$ identifies $\hat{y}_j = 0$ for all $j \in M_0^1$ with $1 < j < p, j \neq q$; $\hat{y}_j = 1$ for all $j \in M_1^2$ with $1 < j < q$, and $\hat{y}_p = 0$. Modify F_0 and F_1 to reflect these variables recognized to be binary in $\hat{\boldsymbol{y}}$. For these binary values, each of the two equations in (4.12) reduces to $y_1 + y_q = 1$. Remove the left equation of (4.12) from (4.11) but maintain the right. This replacement reduces the number of rows indexed by S_1 by one. Repeat this approach until no entries of value 1 remain within a column of $\bar{A}$ corresponding to a variable y_q for which $q \notin F_0 \cup F_1$.

In a similar manner, consider any entry of value 1 (if it exists) within $\tilde{A}$ appearing in some column corresponding to a variable y_q where $q \notin F_0 \cup F_1$. Then, by definition, $q \in S_1 \cap M_1^2$ and there exists a $p \in S_2$ such that $q < p$. We thus have that

$$1 - y_q = \sum_{\substack{j \in M_0^1 \\ j < q}} y_j \quad \text{and} \quad y_p = \sum_{\substack{j \in M_1^2 \\ j < p}} (1 - y_j), \tag{4.13}$$

with the right equation yielding the selected coefficient 1 in $\tilde{A}$. Substitute y_q from the right equation of (4.13) into the left (as $q \in M_1^2$ with $q < p$) to obtain, using $1 \in M_0^1 \cap M_1^2$, that

$$0 = \sum_{\substack{j \in M_0^1 \\ 1 < j < q}} y_j + \sum_{\substack{j \in M_1^2 \\ 1 < j < p, j \neq q}} (1 - y_j) + (1 - y_p).$$

Then $\boldsymbol{y} \in [0,1]^m$ identifies $\hat{y}_j = 0$ for all $j \in M_0^1$ with $1 < j < q$; $\hat{y}_j = 1$ for all $j \in M_1^2$ with $1 < j < p, j \neq q$, and $\hat{y}_p = 1$. Modify F_0 and F_1 to reflect these variables recognized to be binary in $\hat{\boldsymbol{y}}$. For these binary values, each of the two equations in (4.13) reduces to $y_1 + y_q = 1$. Remove the right equation of (4.13) from (4.11) but maintain the left. This replacement reduces the number of rows indexed by S_2 by one. Repeat this approach until no entries of value 1 remain within a column of $\tilde{A}$ corresponding to a variable y_q for which $q \notin F_0 \cup F_1$.

Next, suppose there exists a $q \in (S_1 \cap S_2) \backslash (F_0 \cup F_1)$. The two restrictions in (4.11) having a coefficient of 1 for y_q are

$$1 - y_q = \sum_{\substack{j \in M_0^1 \\ j<q}} y_j \quad \text{and} \quad y_q = \sum_{\substack{j \in M_1^2 \\ j<q}} (1 - y_j), \tag{4.14}$$

with the right equation yielding the coefficient 1 in $\tilde{I}_2$. Substitute y_q from the right equation of (4.14) into the left to obtain, using $1 \in M_0^1 \cap M_1^2$, that

$$0 = \sum_{\substack{j \in M_0^1 \\ 1<j<q}} y_j + \sum_{\substack{j \in M_1^2 \\ 1<j<q}} (1 - y_j).$$

Then $\boldsymbol{y} \in [0,1]^m$ enforces that $\hat{y}_j = 0$ for all $j \in M_0^1$ with $1 < j < q$ and $\hat{y}_j = 1$ for all $j \in M_1^2$ with $1 < j < q$. Modify F_0 and F_1 accordingly. For these binary values of $\hat{y}_j$, each of the two equations in (4.14) reduces to $y_1 + y_q = 1$. Remove the right equation of (4.14) from (4.11) but maintain the left. Repeat until no entries of value 1 remain within a column of $\tilde{I}_2$ corresponding to a variable y_q for which $q \notin F_0 \cup F_1$.

The second part of the proof substitutes $\hat{y}_j = 0$ for all $j \in F_0$ and $\hat{y}_j = 1$ for all $j \in F_1$ within the reduced version of (4.11). This substitution rewrites (4.11) in the form below, where a prime is used to denote the potential changes in the corresponding matrices, and where $\mathbf{0}$ is used to denote appropriately-dimensioned matrices of zeros.

$$\begin{bmatrix} \bar{I}_1' & \mathbf{0} & \bar{I}_2' & \bar{B}' \\ \mathbf{0} & \tilde{I}_1' & \mathbf{0} & \tilde{B}' \end{bmatrix} \begin{bmatrix} \boldsymbol{y}' \end{bmatrix} = \begin{bmatrix} \mathbf{1}' \\ \boldsymbol{d}' \end{bmatrix}. \tag{4.15}$$

The matrices $\bar{B}'$ and $\tilde{B}'$ have all binary entries and preserve the property of $\bar{B}$ and $\tilde{B}$ respectively in that whenever a 1 appears in some row, the value 1 is repeated down the column through the last row. Reorder the second family of equations so that the rows of $\tilde{B}'$ appear in reverse order. Then every column of the resulting coefficient matrix for the adjusted $\boldsymbol{y}'$ possesses the consecutive ones property. Thus, since $\mathbf{1}'$ and $\boldsymbol{d}'$ are integral, the extreme point(s) to (4.15) must be integral [16, 21].

As $\boldsymbol{y} \in [0,1]^m$, $\boldsymbol{y}'$ must be binary, making $\hat{\boldsymbol{y}}$ binary. The proof is complete. □

Theorem 1, together with an extension of (4.5), suggests an alternative formulation to (4.8) of $\text{conv}(P_3(x, \boldsymbol{y}))$, as well as a generalization of a result of [8]. The relationship of (4.5) between lexicographic orderings and base-2 expansions extends to those cases where a vector $\boldsymbol{y} \in \{0,1\}^m$ is lexicographically bounded between vectors $\boldsymbol{\alpha}^1, \boldsymbol{\alpha}^2 \in \{0,1\}^m$ as $\boldsymbol{\alpha}^1 \preceq \boldsymbol{y} \preceq \boldsymbol{\alpha}^2$. Given $\boldsymbol{\alpha}^1$, $\boldsymbol{\alpha}^2, \boldsymbol{y} \in \{0,1\}^m$ for $m \geq 1$, we have

$$\boldsymbol{\alpha}^1 \preceq \boldsymbol{y} \preceq \boldsymbol{\alpha}^2 \Longleftrightarrow \sum_{j=1}^{m} \gamma_j \alpha_j^1 \leq \sum_{j=1}^{m} \gamma_j y_j \leq \sum_{j=1}^{m} \gamma_j \alpha_j^2, \tag{4.16}$$

where the coefficients γ_j are as defined in (4.6). Thus, by computing $\boldsymbol{\alpha}^1 \in \{0,1\}^m$ so that $\sum_{j=1}^{m} 2^{m-j} \alpha_j^1 = \ell$ and $\boldsymbol{\alpha}^2 \in \{0,1\}^m$ so that $\sum_{j=1}^{m} 2^{m-j} \alpha_j^2 = u$, the set $P_3(x, \boldsymbol{y})$ of (4.7) can be expressed as

$$P_3(x, \boldsymbol{y}) \equiv \left\{ (x, \boldsymbol{y}) \in \mathbb{R} \times \{0,1\}^m : x = \sum_{j=1}^{m} 2^{m-j} y_j,\ \boldsymbol{\alpha}^1 \preceq \boldsymbol{y} \preceq \boldsymbol{\alpha}^2 \right\}. \tag{4.17}$$

Theorem 1 allows us to substitute the inequalities of (4.10) for $\boldsymbol{\alpha}^1 \preceq \boldsymbol{y} \preceq \boldsymbol{\alpha}^2$ in (4.17), and the continuous relaxation will give the convex hull of $P_3(x, \boldsymbol{y})$. Nonetheless, this strategy is not recommended in this case, as T of (4.10) will contain at least as many inequalities as $\overline{S}$ of (4.8), and can possibly contain additional variables in $\boldsymbol{y}$. The constraint count follows from the property of base-2 addition that for any $\boldsymbol{a}, \boldsymbol{b} \in \{0,1\}^m$, we have

$$W(\boldsymbol{a}) + W(\boldsymbol{b}) \geq W(\boldsymbol{a} \oplus \boldsymbol{b}), \tag{4.18}$$

where $W(\bullet)$ denotes the Hamming weight of $\bullet$ (the number of entries of value 1 within $\bullet$) and where $\boldsymbol{a} \oplus \boldsymbol{b}$ denotes the base-2 addition of $\boldsymbol{a}$ and $\boldsymbol{b}$. For x having $\ell \leq x \leq u$, let $\boldsymbol{a}$ and $\boldsymbol{b}$ be the base-2 expansions of $u - \ell$ and ℓ respectively, to obtain that the number of inequalities defining $\overline{S}$ in (4.8) is $(m - W(\boldsymbol{a}))$, and the number of inequalities in (4.17) is $W(\boldsymbol{b}) + (m - W(\boldsymbol{a} \oplus \boldsymbol{b}))$, so (4.18) gives us that the former number is bounded above by the latter. An additional $\lceil \log_2(u+1) \rceil - \lceil \log_2(u-\ell+1) \rceil$ variables beyond that of (4.8) are needed in $\boldsymbol{y}$ of (4.17).

As a final note of this section, observe how Theorem 1, (4.16), and (4.17) provide a generalization of work in [8]. Identity (4.16) uses Theorem 1 to obtain the convex hull of the set of vectors that is lexicographically bounded between two vectors while (4.17) relates this bounding to integer variables. In contrast, [8] considers only upper bounds.

4.4 0-1 Knapsack Polytopes

The strategy used to obtain the convex hull of the set $P_3(x, \boldsymbol{y})$ of (4.17) via the inequalities in T of (4.10) is applicable to a special family of 0-1 knapsack polytopes. The key ingredient is that, given vectors $\boldsymbol{\alpha}^1$, $\boldsymbol{\alpha}^2$, $\boldsymbol{y} \in \{0,1\}^m$ for $m \geq 1$, the if-and-only-if implications of (4.16) are applicable to a more general family of coefficients γ_j than described in (4.6). In fact, (4.16) will hold true for coefficients $\gamma_j > 0$ that are "weakly super-decreasing" in that

$$\gamma_j \geq \sum_{i=j+1}^{m} \gamma_i \text{ for each } j = 1, \ldots, m-1. \tag{4.19}$$

Here, it is possible that two vectors in $\{0,1\}^m$ yield the same value $\sum_{j=1}^m \gamma_j \alpha_j^1$, and we assume that $\boldsymbol{\alpha}^1$ is taken as the lexicographically smaller. Similarly, it is possible that two vectors in $\{0,1\}^m$ yield the same value $\sum_{j=1}^m \gamma_j \alpha_j^2$, and we assume that $\boldsymbol{\alpha}^2$ is taken as the lexicographically larger.

Now, consider 0-1 knapsack polytopes of the form

$$KP(\boldsymbol{y}) \equiv \left\{ \boldsymbol{y} \in \{0,1\}^m : \kappa_1 \leq \sum_{j=1}^{m} \gamma_j y_j \leq \kappa_2 \right\}, \tag{4.20}$$

where κ_1 and κ_2 are scalars satisfying $\kappa_1 \leq \kappa_2$, and where the coefficients $\gamma_j > 0$ are weakly super-decreasing. It is a simple task [14, page 300] via a greedy algorithm to compute a vector $\boldsymbol{\alpha}^2 \in \{0,1\}^m$ yielding the largest scalar $\kappa_2' \leq \kappa_2$ such that $\sum_{j=1}^m \gamma_j \alpha_j^2 = \kappa_2'$. A similar greedy algorithm computes a vector $\boldsymbol{\alpha}^1 \in \{0,1\}^m$ yielding the smallest scalar $\kappa_1' \geq \kappa_1$ such that $\sum_{j=1}^m \gamma_j \alpha_j^1 = \kappa_1'$. As alluded to above, we select the lexicographically larger vector if two $\boldsymbol{\alpha}^2$ provide κ_2', and we select the lexicographically smaller vector if two $\boldsymbol{\alpha}^1$ provide κ_1'. We assume without loss of generality that $\kappa_1 \leq \sum_{j=2}^m \gamma_j$ and $\kappa_2 \geq \gamma_1$ so that $\alpha_1^1 = 0$ and $\alpha_1^2 = 1$ since otherwise the value of y_1 would be fixed. (We can also assume that $\gamma_m^1 = 0$ *and* $\gamma_m^2 = 1$ are not both true since otherwise y_m would not appear within the inequalities of (4.10), having the convex hull representation allow $0 \leq y_m \leq 1$.)

The generalization of (4.16) to handle coefficients $\gamma_j > 0$ satisfying (4.19) allows us to rewrite (4.20) as

$$KP(\boldsymbol{y}) = \left\{\boldsymbol{y} \in \{0,1\}^m : \boldsymbol{\alpha}^1 \preceq \boldsymbol{y} \preceq \boldsymbol{\alpha}^2\right\},$$

where $\boldsymbol{\alpha}^1$ and $\boldsymbol{\alpha}^2$ are as described above. This version of $KP(\boldsymbol{y})$ is of the form (4.9) so that it can be rewritten as (4.10) where, as before, $M \equiv \{1,\ldots,m\}$, $M_0^1 = \{i \in M : \alpha_i^1 = 0\}$, $M_1^1 = \{i \in M : \alpha_i^1 = 1\}$, $M_0^2 = \{i \in M : \alpha_i^2 = 0\}$, and $M_1^2 = \{i \in M : \alpha_i^2 = 1\}$. Then Theorem 1 gives us that the set $\overline{T}$ defines the convex hull of the set $KP(\boldsymbol{y})$ of (4.20).

There is an important distinction between using lexicographic orderings to represent a bounded integer variable x having $\ell \leq x \leq u$ and to use these same orderings to model $KP(\boldsymbol{y})$ in (4.20). As mentioned earlier for the case of $\ell \leq x \leq u$, it is preferable to scale x so that the lower bound is 0. In this manner, the lower bounding vector $\boldsymbol{\alpha}^1$ of (4.17) would have $\boldsymbol{\alpha}^1 = \mathbf{0}$ so that a potentially reduced number of inequalities and binary variables can be used. Such a scaling is not, in general, possible when the two-sided knapsack constraint of (4.20) is found within some optimization problem, as the specific binary realizations of $\boldsymbol{y}$ can affect both the objective function and remaining constraints.

This convex hull representation of $KP(\boldsymbol{y})$ relates to earlier work in [6, 8, 9]. The paper [9] forms the minimal cover inequalities defining S of (4.4) associated with those special cases of (4.20) for which $\kappa_1 = 0$, and uses results of [19] to show that the continuous relaxation $\overline{S}$ of S defines the convex hull. The work of [6] simplifies the arguments of [9] by using the interval-matrix property of (4.4). The paper [6] also gives a novel application of weakly super-increasing knapsack problems in the context of discretizing continuous power variables within Signal-to-Interference Ratio restrictions for radio network design. Neither of these papers, however, consider binary expansions of integer variables to obtain $\text{conv}(P_3(x,\boldsymbol{y}))$. Moreover, for the knapsack polytopes, we obtain a more general result by providing the convex hull for those cases having nonzero κ_1 values. Upon observing the relationship between lexicographic orderings and weakly super-decreasing coefficients as provided in (4.16) for (4.19), the paper [8] can be used to model weakly super-decreasing knapsack problems having $\kappa_1 = 0$.

4.5 Computational Experience

Given an optimization problem containing bounded integer variables, the question arises as to whether the explicit algebraic characterization of $\text{conv}(P_3(x,\boldsymbol{y}))$ afforded by (4.8) can improve computational efficiency. Specifically, when a base-2 expansion of a bounded integer variable is performed, do the inequality restrictions of (4.4) included within (4.8) in the form of $\overline{S}$ assist in reducing the number of nodes explored within an enumerative strategy beyond that of simply enforcing $x \leq u$? The work of [17] convincingly argues that binary expansions of bounded integer variables should not be used in practice unless special techniques are employed. The purpose of our computations is not to assess the merits of binary expansions, but rather to investigate the usefulness of including these type inequalities within such representations.

The inequalities defining $\overline{S}$ do not serve to tighten the continuous relaxation of an integer programming problem, but they can help expedite an enumerative strategy. Given an integer variable x satisfying $\ell \leq x \leq u$, the sets $P_3(x, \boldsymbol{y})$ with $\boldsymbol{y}$ substituted for $\boldsymbol{\lambda}$ in (4.3) and $\text{conv}(P_3(x, \boldsymbol{y}))$ of (4.8) both allow x to satisfy $\ell \leq x \leq u$, so that the relaxation relative to x is the same. However, given a fixed value of x, the additional inequalities of (4.8) can restrict the permissible realizations of $\boldsymbol{y}$, thereby curtailing a binary search over $\boldsymbol{y}$. Consider, for example, some x having $u - \ell = 2^{m-1}$ for an integer $m \geq 2$. Then the continuous relaxation of (4.3) with $\boldsymbol{y}$ substituted for $\boldsymbol{\lambda}$ and with $m = \lceil \log_2 n \rceil$ in (4.3), since $m = \lceil \log_2(2^{m-1} + 1) \rceil = \lceil \log_2(u - \ell + 1) \rceil = \lceil \log_2 n \rceil$ as $n = u - \ell + 1$, enforces

$$x = \ell + \sum_{j=1}^{m} 2^{m-j} y_j, \; x \leq u, \; \boldsymbol{y} \in [0, 1]^m. \tag{4.21}$$

The associated vector $\boldsymbol{\alpha}$ has a 1 in the first position and zeros elsewhere so the inequalities defining $\overline{S}$ of (4.8) replace the restriction $x \leq u$ of (4.21) with the $m - 1$ constraints

$$y_1 + y_j \leq 1 \;\; \forall \, j = 2, \ldots, m. \tag{4.22}$$

Now, if $x = u$, all y_j can be fractional within (4.21); for example (4.21) permits $y_j = \frac{2^{m-1}}{2^m - 1}$ for all $j = 1, \ldots, m$. But the inclusion of (4.22) forces $y_1 = 1$ and $y_j = 0$ for $j = 2, \ldots, m$ in this case. Alternatively, for any $j \neq 1$, fixing $y_j = 1$ restricts $y_1 = 0$ within (4.22) but not within (4.21).

We conduct our tests on a bounded, mixed-integer knapsack problem of the form:

$$
\begin{aligned}
\text{MIKP : Minimize} \quad & \sum_{i=1}^{p} x_i + (pU)r & \\
\text{subject to} \quad & \sum_{i=1}^{p} (2x_i) + r = pU - 1 & \\
& 0 \le x_i \le U & \forall\, i = 1, \dots, p \\
& x_i \text{ integer} & \forall\, i = 1, \dots, p \\
& r \ge 0. &
\end{aligned}
$$

Here, there are p integer variables x_i, all having lower bounds of 0 and upper bounds given by the same positive integer U. There is a single continuous variable r that serves as a nonnegative slack on the knapsack constraint. This problem was chosen because it is challenging for enumerative schemes, not permitting $r = 0$ when pU is even.

Given that pU is even, optimal solutions $(\boldsymbol{x}^*, r^*)$ to MIKP and its continuous relaxation, say $\overline{\text{MIKP}}$, are readily available. Relative to MIKP, when p is even, set any $\frac{p}{2} - 1$ variables x_i^* to U, set a single x_i^* to $U - 1$, set the remaining x_i^* to 0, and fix $r^* = 1$. When p is odd so that U must be even, set any $p - 1$ variables x_i^* to $\frac{U}{2}$, set the remaining variable x_i^* to $\frac{U}{2} - 1$, and fix $r^* = 1$. In either case, the objective function value is $\frac{3}{2}pU - 1$. For $\overline{\text{MIKP}}$, an optimal solution $(\boldsymbol{x}', r')$ has $x_i' = \frac{pU-1}{2p}$ for all i, $r' = 0$, and objective value $\frac{1}{2}(pU - 1)$.

We reformulate MIKP as a mixed-binary program using the set $P_3(x, \boldsymbol{\lambda})$ of (4.3) with $\boldsymbol{y}$ substituted for $\boldsymbol{\lambda}$ in the following manner. For each integer variable x_i, we associate a distinct $\boldsymbol{y}^i \in \{0,1\}^{\lceil \log_2(U+1) \rceil}$, since $n = U+1$ within (4.3), and let y_j^i denote entry j of $\boldsymbol{y}^i$. Then we perform a base-2 expansion on each x_i with $\ell = 0$ to obtain the following, where BKP1 represents our first mixed-binary knapsack problem formulation:

$$
\begin{aligned}
\text{BKP1 : Minimize} \quad & \sum_{i=1}^{p} x_i + (pU)r \\
\text{subject to} \quad & \sum_{i=1}^{p} (2x_i) + r = pU - 1 \\
& x_i = \sum_{j=1}^{\lceil \log_2(U+1) \rceil} 2^{\lceil \log_2(U+1) \rceil - j} y_j^i \qquad \forall\, i = 1, \ldots, p \\
& x_i \leq U \qquad \forall\, i = 1, \ldots, p \qquad (4.23) \\
& \boldsymbol{y}^i \in \{0,1\}^m \qquad \forall\, i = 1, \ldots, p \\
& r \geq 0.
\end{aligned}
$$

The second mixed-binary form of MIKP, denoted BKP2, is obtained by replacing the p inequalities of (4.23) with the pm_0 inequalities of (4.4), where there are m_0 such inequalities for each x_i. Here, we let $m = \lceil \log_2(U + 1) \rceil$ so that, as before, the vector $\boldsymbol{\alpha} \in \{0,1\}^m$ is defined in terms of the scalar $u - \ell = U$ to satisfy $\sum_{j=1}^{m} 2^{m-j} \alpha_j = U$. Then we again have $M \equiv \{1, \ldots, m\}$, $M_0 = \{i \in M : \alpha_i = 0\}$, $M_1 = \{i \in M : \alpha_i = 1\}$, and $m_0 = |M_0|$ with $m_1 = |M_1|$. Consequently, BKP2 is BKP1 with (4.23) replaced by

$$
y_j^i \leq \sum_{\substack{k \in M_1 \\ k < j}} (1 - y_k^i) \;\; \forall\, (i,j),\; i = 1, \ldots, p,\; j \in M_0. \qquad (4.24)
$$

The computational tests were designed to compare the relative merits of BKP1 and BKP2. Instances with various values for U and p were submitted to ILOG CPLEX 11.0 on a Sun V440 workstation with 16 GB of RAM and four 1.6 GHz CPU's running Solaris 10. The *presolve* option in CPLEX was turned off to more accurately assess the utility of (4.24).

Fifty-one different test problems were attempted, with the parameter U successively increased. The results are summarized in Table 1. The table is arranged in seven columns, with the first providing the problem number, the second and third stating the input parameters U and p respectively, the fourth and fifth respectively giving the CPU execution times in seconds and the number of nodes explored in the binary search for BKP1, and the last two giving this same information for BKP2. All problems were assigned a time limit of 10000 CPU seconds, with a $-$ used to denote that this limit was exceeded. For comparative purposes, when Problem MIKP was directly submitted to CPLEX, every instance was solved within .01 CPU seconds.

Table 1 reinforces the findings of [17], that binary expansions of bounded integer variables should not be used in practice unless special techniques are employed. It also suggests that BKP2 is preferable to BKP1 for most problem instances. For 46 of the 51 problems tested (20, 23, 29, 30, and 45 being the exceptions), BKP2 performed at least as well as BKP1 in terms of both CPU times and the number of nodes explored. The extra constraints (4.24) present in BKP2 typically allowed for a reduction in the number of nodes explored. For the 31 test cases that both formulations solved to optimality and expended at least one CPU second, BKP2 reduced the effort required by BKP1 by, on average, 59% in terms of CPU execution times and 60% in terms of the number of nodes explored. As would be expected, instances involving larger numbers of variables p could be solved when smaller upper bounds U were used.

Interestingly, the values of U led to different forms of contraints of the type (4.24), reflecting different computational results. We considered no U for which $U + 1$ is a power of 2 since no constraints would be present in (4.24), and BKP2 would reduce to BKP1. For those cases in which U is a power of 2, inequalities (4.24) are of the type (4.22), which tend to restrict branching in the enumerative tree. However, when $U + 2$ is a power of 2, then inequalities (4.24) become

$$y_m^i \leq \sum_{k=1}^{m-1} (1 - y_k^i) \ \ \forall\, i = 1, \ldots, p,$$

which are relatively weak. This variation in the strength of cuts might help explain why Problems 36 and 37 with $U = 30$ required longer running times using BKP2 than Problems 40 and 41, respectively, for this same form when $U = 32$. The same logic holds for Problems 45 and 48 with BKP2.

These results are preliminary, but tend to suggest that the additional cuts (4.24) of BKP2 are useful when conducting base-2 expansions of integer variables. Depending on the problem of concern, a strategic implementation may serve to expedite an enumerative search and lead to more efficient solution strategies.

Problem	Parameters		BKP1		BKP2	
Number	U	p	Time	Nodes	Time	Nodes
1	4	4	0.01	43	0.01	33
2	4	8	0.42	4377	0.13	1100
3	4	12	28.00	269167	2.16	19948
4	4	16	531.64	4444014	26.22	223212
5	4	20	–	–	545.55	41337
6	6	4	0.02	116	0.01	30
7	6	8	3.72	37691	0.78	9597
8	6	12	170.67	1472287	132.44	1178809
9	6	16	–	–	7862.82	48148528
10	8	4	0.06	541	0.02	151
11	8	8	19.39	199539	6.95	56926
12	8	10	6915.05	65745383	99.18	717965
13	8	12	9358.48	82868619	670.32	5695074
14	10	4	0.10	1002	0.02	206
15	10	8	74.97	759959	6.74	58498
16	10	10	2116.99	20067301	96.78	746016
17	10	12	–	–	3140.55	22424458
18	13	4	0.12	1450	0.05	528
19	13	8	928.03	9055811	400.08	4815216
20	13	10	6374.92	59452931	–	–
21	14	4	0.15	1696	0.04	517
22	14	7	55.17	537771	15.87	188863
23	14	8	293.77	2983979	285.24	3218810
24	14	10	5566.94	46418657	1874.48	19637668
25	16	4	0.34	3996	0.19	1801
26	16	6	15.30	151067	8.81	70574
27	16	8	531.25	4836820	387.15	3020196
28	16	10	–	–	–	–
29	20	4	0.20	2679	0.30	3695
30	20	6	24.81	299121	56.25	625597
31	20	8	3314.68	39278278	2418.43	23714696
32	25	4	0.96	11283	0.53	6770
33	25	6	190.15	2097275	28.28	329260
34	25	8	–	–	8868.03	92084251
35	30	4	1.44	17090	0.31	4066
36	30	5	15.57	180195	8.96	114146
37	30	6	807.36	8833755	151.82	1508086
38	30	7	7593.46	76698740	1872.03	21405569
39	32	4	1.86	23378	0.49	5666
40	32	5	35.48	472612	4.36	47655
41	32	6	455.27	4054616	85.23	709202
42	32	7	8418.81	101731770	2689.05	25110337
43	62	4	9.54	111537	2.19	29674
44	62	5	196.33	2413475	75.97	949121
45	62	6	4660.84	44019164	5039.48	46042359
46	64	4	23.41	241704	2.99	34543
47	64	5	943.29	11402642	97.05	975373
48	64	6	–	–	3875.00	31454104
49	128	4	74.12	794381	42.71	429435
50	128	5	8578.16	101187915	1975.86	17545861
51	256	4	617.30	6162856	234.17	2006200

Table 4.1: Computational comparisons.

4.6 Conclusions

Whereas base-2 expansions of bounded integer variables are classical in discrete optimization, and whereas tight polyhedral outer-approximations of 0-1 linear programs are widely-recognized as being a key ingredient for improved solution techniques, these two concepts have not been combined to motivate convex hull (ideal) representations of such expansions. Given an integer variable, this chapter provides an ideal form in the original variable space by explicitly describing the additional inequalities needed to capture the convex hull. The representation requires at most $\lceil \log_2 n \rceil - 1$ minimal-cover type inequalities, where n is the number of permissible realizations of the discrete variable.

Our arguments are based on a lexicographic ordering of binary vectors. Given such a vector, the convex hull of the set of binary vectors that is lexicographically less than or equal to this chosen vector is presented using minimal cover inequalities. This convex hull form is then related to base-2 expansions, allowing us to model bounded integer variables. A similar description gives the convex hull for the set of vectors that is lexicographically greater than or equal to a given binary vector. We combine these results to characterize the convex hull of the set of binary vectors that is lexicographically bounded between two binary vectors, using a combination of minimal cover and set covering restrictions. This characterization leads to the ideal representation of binary knapsack polytopes having weakly super-decreasing coefficients, where the knapsack constraint can have both lower and upper bounds, and serves to extend earlier work that addresses only upper bounds.

Preliminary computations suggest that the additional inequalities needed to describe the convex hull forms of base-2 expansions are useful within an enumerative setting. That is, these additional inequalities tend to reduce the number of nodes explored in a binary search tree compared to a standard base-2 expansion, and thereby lessen the overall effort. Computational results on mixed-integer knapsack problems exhibited, on average, a 59% reduction in CPU times and a 60% reduction in the number of nodes explored. The utility of these cuts appears to depend on the number of permissible realizations of the integer variable. On the one extreme, when the number of realizations is a power of 2, no inequalities are generated since the convex hull is already available. On the other extreme, significant strength is obtained when the number of realizations is one greater than a power of 2. More extensive tests are needed to determine the types and structures of problems for which these cuts will be most effective.

Bibliography

[1] Adams, W.P. and Henry, S.M., "Base-2 Expansions for Linearizing Products of Functions of Discrete Variables," *Operations Research*, to appear, (manuscript 2011).

[2] Adams, W.P. and Sherali, H.D., "A Hierarchy of Relaxations Leading to the Convex Hull Representation for General Discrete Optimization Problems," *Annals of Operations Research*, Vol. 140, No. 1, pp 21–47, 2005.

[3] Angulo, G., Ahmed, S., and Dey, S.S., "Forbidding Extreme Points from the 0-1 Hypercube," *Optimization Online*, 2012.

[4] Coppersmith, L. and Lee, J., "Parsimonious Binary-Encoding in Integer Programming," *Discrete Optimization*, Vol. 2, No. 3, pp 190–200, 2005.

[5] Dantzig, G.B., "On the Significance of Solving Linear Programming Problems with Some Integer Variables," *Econometrica*, Vol. 28, No. 1, pp 30–44, 1960.

[6] D'Andreagiovanni, F., "Pure 0-1 Programming Approaches to Wireless Network Design," Ph.D. Dissertation, University of Rome, 2010.

[7] Garfinkel, R.S. and Nemhauser, G.L., *Integer Programming*, Wiley, New York, NY, 1972.

[8] Gillmann, R. and Kaibel, V., "Revlex-initial 0/1-polytopes," *Journal of Combinatorial Theory Series A*, Vol. 113, No. 5, pp 799–821, 2006.

[9] Laurent, M. and Sassano, A., "A Characterization of Knapsacks with the Max-Flow-Min-Cut Property," *Operations Research Letters*, Vol. 11, No. 2, pp 105–110, 1992.

[10] Lee, J., "All-Different Polytopes," *Journal of Combinatorial Optimization*, Vol. 6, No. 3, pp 335–352, 2002.

[11] Lee, J. and Margot, F., "More on a Binary-Encoded Coloring Formulation," *Lecture Notes in Computer Science*, Vol. 3064, pp 271–282, 2004.

[12] Lee, J. and Margot, F., "On a Binary-Encoded ILP Coloring Formulation," *INFORMS Journal on Computing*, Vol. 19, No. 3, pp 406–415, 2007.

[13] McMillan, Jr., C., *Mathematical Programming: An Introduction to the Design and Application of Optimal Decision Machines*, Wiley, New York, NY, 1970.

[14] Menezes, A.J., van Oorschot, P.C., and Vanstone, S.A., *Handbook of Applied Cryptography*, CRC Press, Boca Raton, FL, 1997.

[15] Murty, K.G., *Linear and Combinatorial Programming*, Wiley, New York, NY, 1976.

[16] Nemhauser, G.L. and Wolsey, L.A., *Integer and Combinatorial Optimization*, John Wiley & Sons, New York, NY, 1999.

[17] Owen, J.H. and Mehrotra, S., “On the Value of Binary Expansions for General Mixed-Integer Linear Programs,” *Operations Research*, Vol. 50, No. 5, pp 810–819, 2002.

[18] Papadimitriou, C.H. and Steiglitz, K., *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[19] Seymour, P.D., “The Matroids with the Max-Flow-Min-Cut Property,” *Journal of Combinatorial Theory Series B*, Vol. 23, pp 189–222, 1977.

[20] Taha, H., *Integer Programming: Theory, Applications, and Computations*, Academic Press, New York, NY, 1975.

[21] Veinnott, Jr., A.F. and Wagner, H.M., “Optimal Capacity Scheduling – I,” *Operations Research*, Vol. 10, No. 4, pp 518–532, 1962.

[22] Zionts, S., *Linear and Integer Programming*, Prentice-Hall, Englewood Cliffs, NJ, 1974.