Modeling Disjunctive Constraints with a Logarithmic Number of Binary Variables and Constraints[⋆] [⋆⋆]

Juan Pablo Vielma and George L. Nemhauser

H. Milton Stewart School of Industrial and Systems Engineering,
Georgia Institute of Technology, Atlanta, GA, USA.
{jvielma, gnemhaus}@isye.gatech.edu

Abstract. Many combinatorial constraints over continuous variables such as SOS1 and SOS2 constraints can be interpreted as disjunctive constraints that restrict the variables to lie in the union of m specially structured polyhedra. Known mixed integer binary formulations for these constraints have a number of binary variables and extra constraints that is linear in m. We give sufficient conditions for constructing formulations for these constraints with a number of binary variables and extra constraints that is logarithmic in m. Using these conditions we introduce the first mixed integer binary formulations for SOS1 and SOS2 constraints that use a number of binary variables and extra constraints that is logarithmic in the number of continuous variables. We also introduce the first mixed integer binary formulations for piecewise linear functions of one and two variables that use a number of binary variables and extra constraints that is logarithmic in the number of linear pieces of the functions. We prove that the new formulations for piecewise linear functions have favorable tightness properties and present computational results showing that they can significantly outperform other mixed integer binary formulations.

1 Introduction

An important question in the area of mixed integer programming (MIP) is characterizing when a disjunctive constraint of the form

$$z \in \bigcup_{i \in I} P_i \subset \mathbb{R}^n, \tag{1}$$

where $P_i = \{z \in \mathbb{R}^n : A^i z \leq b^i\}$, can be modeled as a binary integer program. Jeroslow and Lowe ([1–3]) showed that a necessary and sufficient condition is for $\{P_i\}_{i \in I}$ to be a finite family of polyhedra with a common recession cone.

[⋆] This research has been supported by NSF grant CMMI-0522485, AFOSR grant FA9550-07-1-0177 and Exxon Mobil Upstream Research Company.

[⋆⋆] The authors would like to thank Daniel Espinoza for pointing out the relation between SOS2 compatible functions and Gray codes.

Using results from disjunctive programming ([4–9]) they showed that, in this case, constraint (1) can be simply modeled as

$$A^i z^i \leq x_i b^i \quad \forall i \in I, \quad z = \sum_{i \in I} z^i, \quad \sum_{i \in I} x_i = 1, \quad x_i \in \{0,1\} \quad \forall i \in I. \tag{2}$$

The possibility of reducing the number of continuous variables in these models has been studied in [10–12], but the number of binary variables and extra constraints needed to model (1) has received little attention. However, it has been observed that a careful construction can yield a much smaller model than a naive approach. Perhaps the simplest example comes from the equivalence between general integer and binary integer programming (see for example page 12 of [13]). The requirement $x \in [0,u] \cap \mathbb{Z}$ can be written in the form (1) by letting $P_i := \{i\}$ for all i in $I := [0,u] \cap \mathbb{Z}$ which, after some algebraic simplifications, yields a representation of the form (2) given by

$$z = \sum_{i \in I} i\, x_i, \quad \sum_{i \in I} x_i = 1, \quad x_i \in \{0,1\} \quad \forall i \in I. \tag{3}$$

This formulation has a number of binary variables that is linear in $|I|$ and can be replaced by

$$z = \sum_{i=0}^{\lfloor \log_2 u \rfloor} 2^i\, x_i, \quad z \leq u, \quad x_i \in \{0,1\} \quad \forall i \in \{0, \ldots, \lfloor \log_2 u \rfloor\}. \tag{4}$$

In contrast to (3), (4) has a number of binary variables that is logarithmic in $|I|$. Although (4) appears in the mathematical programming literature as early as [14], and the possibility of modeling with a logarithmic number of binary variables and a linear number of constraints is studied in the theory of disjunctive programming (see for example [5]) and in [15], we are not aware of any other non-trivial formulations with a logarithmic number of binary variables and extra constraints.

The main objective of this work is to show that some well known classes of constraints of the form (1) can be modeled with a logarithmic number of binary variables and extra constraints. Although modeling with fewer binary variables and constraints might seem advantageous, a smaller formulation is not necessarily a better formulation (see for example section I.1.5 of [16]). More constraints might provide a tighter LP relaxation and more variables might do the same by exploiting the favorable properties of projection (see for example [17]). For this reason, we will also show that under some conditions our new formulations are as tight as any other mixed integer formulation, and we empirically show that they can provide a significant computational advantage.

The paper is organized as follows. In Section 2 we study the modeling of a class of hard combinatorial constraints. In particular we introduce the first formulations for SOS1 and SOS2 constraints that use only a logarithmic number of binary variables and extra constraints. In Section 3 we relate the modeling with a logarithmic number of binary variables to branching and we introduce sufficient conditions for these models to exist. We then show that for a broad class of problems the new formulations are as tight as any other mixed integer programming formulation. In Section 4 we use the sufficient conditions to present a new formulation for non-separable piecewise linear functions of one and two variables that uses only a logarithmic number of binary variables and extra constraints. In Section 5 we show that the new models for piecewise linear functions of one and two variables can perform significantly better than the standard binary models. Section 6 gives some conclusions.

2 Modeling a Class of Hard Combinatorial Constraints

In this section we study a class of constraints of the form (1) in which the polyhedra P_i have the simple structure of only allowing some subsets of variables to be non-zero. Specifically, we study constraints over a vector of continuous variables λ indexed by a finite set J that are of the form

$$\lambda \in \bigcup_{i\in I} Q(S_i) \subset \Delta^J, \tag{5}$$

where I is a finite set, $\Delta^J := \{\lambda \in \mathbb{R}^J_+ : \sum_{j\in J} \lambda_j \leq 1\}$ is the $|J|$-dimensional simplex in $\mathbb{R}^J$, $S_i \subset J$ for each $i \in I$ and

$$Q(S_i) = \left\{\lambda \in \Delta^J : \lambda_j \leq 0 \,\forall\, j \notin S_i\right\}. \tag{6}$$

Furthermore, without loss of generality we assume that $\bigcup_{i\in I} S_i = J$. Except for Theorem 3, our results easily extend to the case in which the simplex Δ^J is replaced by a box in $\mathbb{R}^J_+$, but the restriction to Δ^J greatly simplifies the presentation.

Disjunctive constraint (5) includes SOS1 and SOS2 constraints [18] over continuous variables in Δ^J. SOS1 constraints on $\lambda \in \mathbb{R}^n_+$ allow at most one of the λ variables to be non-zero which can be modeled by letting $I = J = \{1, \dots, n\}$ and $S_i = \{i\}$ for each $i \in I$. SOS2 constraints on $(\lambda_j)_{j=0}^n \in \mathbb{R}^{n+1}_+$ allow at most two λ variables to be non-zero and have the extra requirement that if two variables are non-zero their indices must be adjacent. This can be modeled by letting $I = \{1, \dots, n\}$, $J = \{0, \dots, n\}$ and $S_i = \{i-1, i\}$ for each $i \in I$.

Mixed integer binary models for SOS1 and SOS2 constraints have been known for many years (see for example [19, 20]), and some recent research has focused on branch-and-cut algorithms that do not use binary variables [21–24]. However, the incentive of being able to use state of the art MIP solvers (see for example the discussion in section 5 of [25]) makes binary models for these constraints very attractive (see for example [26–29]).

We first review a formulation for (5) with a linear number of binary variables and then we give a formulation with a logarithmic number of binary variables and a linear number of extra constraints. We then study how to obtain a formulation with a logarithmic number of variables and a logarithmic number of extra constraints and show that this can be achieved for SOS1 and SOS2 constraints.

The most direct way of formulating (5) as an integer programming problem is by assigning a binary variable for each set $Q(S_i)$ and using formulation (2). After some algebraic simplifications this yields the formulation of (5) given by

$$\lambda \in \Delta^J, \quad \lambda_j \leq \sum_{i \in I(j)} x_i \quad \forall j \in J, \quad \sum_{i \in I} x_i = 1, \quad x_i \in \{0,1\} \quad \forall i \in I$$

where $I(j) = \{i \in I : j \in S_i\}$. This gives a formulation with $|I|$ binary variables and $|J| + 1$ extra constraints and yields the standard formulations for SOS1 and SOS2 constraints. (We consider Δ^J as the original constraints and disregard the bounds on x.)

The following proposition shows that by using techniques from [15] we can obtain a formulation with $\lceil \log_2 |I| \rceil$ binary variables and $|I|$ extra constraints.

Proposition 1. *Let $B : I \to \{0,1\}^{\lceil \log_2 |I| \rceil}$ be any injective function. Then*

$$\lambda \in \Delta^J, \sum_{j \notin S_i} \lambda_j \leq \sum_{l \notin \sigma(B(i))} x_l + \sum_{l \in \sigma(B(i))} (1 - x_l) \quad \forall i \in I,$$
$$x_l \in \{0,1\} \quad \forall l \in L(|I|) \quad (7)$$

where $\sigma(B)$ is the support of vector B and $L(r) := \{1, \ldots, \lceil \log_2 r \rceil\}$, is a valid formulation for (5).

For SOS1 constraints, for which $|I(j)| = 1$ for all $j \in J$, we can obtain the following alternative formulation of (5) which has $\lceil \log_2 |I| \rceil$ binary variables and $2\lceil \log_2 |I| \rceil$ extra constraints.

Proposition 2. *Let $B : I \to \{0,1\}^{\lceil \log_2 |I| \rceil}$ be any injective function. Then*

$$\lambda \in \Delta^J, \sum_{j \in J^+(l,B)} \lambda_j \leq x_l, \sum_{j \in J^0(l,B)} \lambda_j \leq (1 - x_l), \, x_l \in \{0,1\} \, \forall l \in L(|I|), \quad (8)$$

where $J^+(l,B) = \{j \in J : \forall i \in I(j) \quad l \in \sigma(B(i))\}$ and $J^0(l,B) = \{j \in J : \forall i \in I(j) \quad l \notin \sigma(B(i))\}$, is a valid formulation for SOS1 constraints.

The following example illustrates formulation (8) for SOS1 constraints.

Example 1 Let $J = \{1, \ldots, 4\}$ and $(\lambda_j)_{j=1}^4 \in \Delta^J$ be constrained to be SOS1 and let $B^*(1) = (1,1)^T$, $B^*(2) = (1,0)^T$, $B^*(3) = (0,1)^T$ and $B^*(4) = (0,0)^T$. Formulation (8) for this case with $B = B^*$ is

$$\lambda \in \Delta^J, \quad x_1, x_2 \in \{0,1\}, \quad \lambda_1 + \lambda_2 \leq x_1, \quad \lambda_3 + \lambda_4 \leq 1 - x_1, \quad \lambda_1 + \lambda_3 \leq x_2,$$
$$\lambda_2 + \lambda_4 \leq 1 - x_2.$$

Formulation (8) is valid for SOS1 constraints independent of the choice of B. In contrast, for SOS2 constraints, where $|I(j)| = 2$ for some $j \in J$, formulation (8) can be invalid for some choices of B. This is illustrated by the following example.

Example 2
Let $J = \{0, \ldots, 4\}$ and $(\lambda_j)_{j=0}^4 \in \Delta^J$ be constrained to be SOS2. Formulation (8) for this case with $B = B^*$ is

$$\lambda \in \Delta^J, \quad x_1, x_2 \in \{0,1\}, \quad \lambda_0 + \lambda_1 \le x_1, \quad \lambda_3 + \lambda_4 \le 1 - x_1, \quad \lambda_0 \le x_2, \quad \lambda_4 \le 1 - x_2$$

which has $\lambda_0 = 1/2$, $\lambda_2 = 1/2$, $\lambda_1 = \lambda_3 = \lambda_4 = 0$, $x_1 = x_2 = 1$ as a feasible solution that does not comply with SOS2 constraints. However, the formulation can be made valid by adding constraints

$$\lambda_2 \le x_1 + x_2, \quad \lambda_2 \le 2 - x_1 - x_2. \tag{9}$$

For any B we can always correct formulation (8) for SOS2 constraints by adding a number of extra linear inequalities, but with a careful selection of B the validity of the model can be preserved without the need for additional constraints.

Definition 1 (SOS2 Compatible Function). *We say that an injective function $B : \{1, \ldots, n\} \to \{0,1\}^{\lceil \log_2(n) \rceil}$ is* compatible *with an SOS2 constraint on $(\lambda_j)_{j=0}^n \in \mathbb{R}_+^{n+1}$ if for all $i \in \{1, \ldots, n-1\}$ the vectors $B(i)$ and $B(i+1)$ differ in at most one component.*

Theorem 1. *If B is an SOS2 compatible function then* (8) *is valid for SOS2 constraints.*

The following example illustrates how an SOS2 compatible function yields a valid formulation.

Example 2 continued
Let $B^0(1) = (1,0)^T$, $B^0(2) = (1,1)^T$, $B^0(3) = (0,1)^T$ and $B^0(4) = (0,0)^T$. Formulation (8) with $B = B^0$ for the same SOS2 constraints is

$$\lambda \in \Delta^J, \quad x_1, x_2 \in \{0,1\}$$

$$\lambda_0 + \lambda_1 \le x_1, \quad \lambda_3 + \lambda_4 \le (1 - x_1) \tag{10}$$

$$\lambda_2 \le x_2, \quad \lambda_0 + \lambda_4 \le (1 - x_2). \tag{11}$$

An SOS2 compatible function can always be constructed and for each $n \in \mathbb{Z}_+$ there are several SOS2 compatible functions. In fact, definition 1 is equivalent to requiring $(B(i))_{i=1}^n$ to be a *reflected binary* or *Gray code* (see for example [30]).

3 Branching and Modeling with a Logarithmic Number of Binary Variables and Constraints

The way in which formulation (8) is constructed does not provide a clear interpretation of the binary variables used, which makes it hard to extend to other combinatorial constraints. In this section we develop a more general scheme which is related to specialized branching schemes.

We can identify each vector in $\{0,1\}^{\lceil \log_2 |I| \rceil}$ with a leaf in a binary tree with $\lceil \log_2 |I| \rceil$ levels in a way such that each component corresponds to a level and the value of that component indicates the selected branch in that level. Then, using function B we can identify each set $Q(S_i)$ with a leaf in the binary tree and we can interpret each of the $\lceil \log_2 |I| \rceil$ variables as the execution of a branching scheme on sets $Q(S_i)$. The formulations in Example 2 illustrates this idea.

In formulation (8) with $B = B^0$ the branching scheme associated with x_1 sets $\lambda_0 = \lambda_1 = 0$ when $x_1 = 0$ and $\lambda_3 = \lambda_4 = 0$ when $x_1 = 1$, which is equivalent to the traditional SOS2 constraint branching of [18] whose dichotomy is fixing to zero variables to the "left of" (smaller than) a certain index in one branch and to the "right" (greater) in the other. In contrast, the scheme associated with x_2 sets $\lambda_2 = 0$ when $x_2 = 0$ and $\lambda_0 = \lambda_4 = 0$ when $x_2 = 1$, which is different from the traditional branching as its dichotomy can be interpreted as fixing variables in the "center" and on the "sides" respectively. If we use function B^* instead we recover the traditional branching. The drawback of the B^* scheme is that the second level branching cannot be implemented independently of the first one using linear inequalities. For B^0 the branch alternatives associated with x_2 are implemented by (11), which only include binary variable x_2. In contrast, for B^* one of the branching alternatives requires additional constraints (9) which involve both x_1 and x_2.

This example illustrates that a sufficient condition for modeling (5) with a logarithmic number of binary variables and extra constraints is to have a binary branching scheme for $\lambda \in \bigcup_{i \in I} Q(S_i)$ with a logarithmic number of dichotomies and for which each dichotomy can be implemented independently. This condition is formalized in the following definition.

Definition 2. (Independent Branching Scheme) *$\{L_k, R_k\}_{k=1}^{d}$ with $L_k, R_k \subset J$ is an independent branching scheme of depth d for disjunctive constraint* (5) *if $\bigcup_{i \in I} Q(S_i) = \bigcap_{k=1}^{d} (Q(L_k) \cup Q(R_k))$.*

This definition can then be used in the following theorem and immediately gives a sufficient condition for modeling with a logarithmic number of variables and constraints.

Theorem 2. *Let $\{Q(S_i)\}_{i \in I}$ be a finite family of polyhedra of the form* (6) *and let $\{L_k, R_k\}_{k=1}^{\lceil \log_2(|I|) \rceil}$ be an independent branching scheme for $\lambda \in \bigcup_{i \in I} Q(S_i)$.*

Then

$$\lambda \in \Delta^J, \quad \sum_{j \notin L_k} \lambda_j \leq x_k, \quad \sum_{j \notin R_k} \lambda_j \leq (1 - x_k),$$
$$x_k \in \{0,1\} \quad \forall k \in \{1, \ldots, \lceil \log_2(|I|) \rceil\} \qquad (12)$$

is a valid formulation for (5) *with* $\lceil \log_2(|I|) \rceil$ *binary variables and* $2\lceil \log_2(|I|) \rceil$ *extra constraints.*

Formulation (8) with $B = B^0$ in Example 2 illustrates how an SOS2 compatible function induces an independent branching scheme for SOS2 constraints. In general, given an SOS2 compatible function $B : \{1, \ldots, n\} \to \{0,1\}^{\lceil \log_2(n) \rceil}$ the induced independent branching is given by $L_k = J \setminus J^+(k, B)$, $R_k = J \setminus J^0(l, B)$ for all $k \in \{1, \ldots, n\}$.

Formulation (12) in Proposition 2 can be interpreted as a way of implementing a specialized branching scheme using binary variables. Similar techniques for implementing specialized branching schemes have been previously introduced for example, in [31] and [32], but the resulting models require at least a linear number of binary variables. To the best of our knowledge the first non-trivial independent branching schemes of logarithmic depth are the ones for SOS1 constraints from Proposition 2 and for SOS2 constraints induced by an SOS2 compatible function.

Formulation (12) can be obtained by algebraic simplifications from formulation (2) of (5) rewritten as the conjunction of two-term polyhedral disjunctions. Both the simplifications and the rewrite can result in a significant reduction in the tightness of the linear programming relaxation of (12) (see for example [5, 10–12]). Fortunately, as the following propositions shows, the restriction to Δ^J makes (12) as tight as any other mixed integer formulation for (5).

Theorem 3. *Let* P_λ *and* Q_λ *be the projection onto the* λ *variables of the LP relaxation of formulation* (12) *and of any other mixed integer programming formulation of* (5) *respectively. Then* $P_\lambda = \operatorname{conv}\left(\bigcup_{i \in I} Q(S_i)\right)$ *and hence* $P_\lambda \subseteq Q_\lambda$.

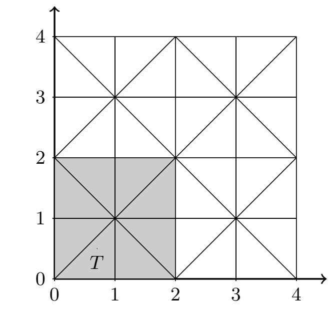
Theorem 3 might no longer be true if we do not restrict to Δ^J, but this restriction is not too severe as it includes a popular way of modeling piecewise linear functions. We explore this further in the following section.

4 Modeling Nonseparable Piecewise Linear Functions of Two Variables

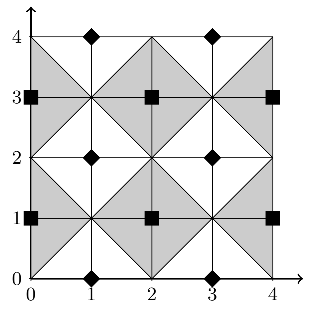
In this section we use Theorem 2 to construct a model for non-separable piecewise linear functions of two variables that use a number of binary variables and extra constraints that is logarithmic in the number of linear pieces of the functions. Although some non-separable functions can be separated there are many practical reasons to avoid this separation (see for example the discussion on page 569 of [24]).

Imposing SOS2 constraints on $(\lambda_j)_{j=0}^n \in \Delta^J$ with $J = \{0, \dots, n\}$ is a popular way of modeling a one variable piecewise-linear function which is linear in n different intervals (see for example [22, 23]). This approach has been extended to non-separable piecewise linear functions in [33, 24, 34, 35]. For functions of two variables this approach can be described as follows.

We assume that for an even integer w we have a continuous function $f : [0, w]^2 \to \mathbb{R}$ which we want to approximate by a piecewise linear function. A common approach is to partition $[0, w]^2$ into a number of triangles and approximate f with a piecewise linear function that is linear in each triangle. One possible triangulation of $[0, w]^2$ is the $\mathbf{J}_1$ or "Union Jack" triangulation (see for example [36]) which is depicted in Figure 1(a) for $w = 4$. The $\mathbf{J}_1$ triangulation of $[0, w]^2$ for any even integer w is simply obtained by adding copies of the 8 triangles shaded gray in Figure 1(a). This yields a triangulation with a total of $2w^2$ triangles. We use this triangulation to approximate f with a piecewise



(a) Example of "Union Jack" Triangulation



(b) Triangle selecting branching

Fig. 1. Triangulations

linear function that we denote by g. Let I be the set of all the triangles of the $\mathbf{J}_1$ triangulation of $[0, w]^2$ and let S_i be the vertices of triangle i. For example, in Figure 1(a), the vertices of the triangle labeled T are $S_T := \{(0, 0), (1, 0), (1, 1)\}$. A valid model for $g(y)$ (see for example [33, 24, 34]) is

$$\sum_{j \in J} \lambda_j = 1, \quad y = \sum_{j \in J} v_j \lambda_j, \quad g(y) = \sum_{j \in J} f(v_j) \lambda_j \tag{13a}$$

$$\lambda \in \bigcup_{i \in I} Q(S_i) \subset \Delta^J, \tag{13b}$$

where $J := \{0, \dots, w\}^2$, $v_j = j$ for $j \in J$. This model becomes a traditional model for one variable piecewise linear functions (see for example [22, 23]) when we restrict it to one coordinate of $[0, w]^2$.

To obtain a mixed integer formulation of (13) with a logarithmic number of binary variables and extra constraints it suffices to construct an independent binary branching scheme of logarithmic depth for (13b) and use formulation (12). Binary branching schemes for (13b) with a similar triangulation have been developed in [34] and [24], but they are either not independent or have too many dichotomies. We adapt some of the ideas of these branching schemes to develop an independent branching scheme for the two-dimensional $\mathbf{J}_1$ triangulation. Our independent branching scheme will basically select a triangle by forbidding the use of vertices in J. We divide this selection into two phases. We first select the square in the grid induced by the triangulation and we then select one of the two triangles inside this square.

To implement the first branching phase we use the observation made in [24, 34] that selecting a square can be achieved by applying SOS2 branching to each component. To make this type of branching independent it then suffices to use the independent SOS2 branching induced by an SOS2 compatible function. This results in the set of constraints

$$\sum_{s=0}^{w}\sum_{r\in J_2^+(l,B,w)}\lambda_{(r,s)} \le x_{(1,l)},\quad \sum_{s=0}^{w}\sum_{r\in J_2^0(l,B,w)}\lambda_{(r,s)} \le 1-x_{(1,l)},$$
$$x_{(1,l)}\in\{0,1\}\quad \forall l\in L(w), \tag{14a}$$
$$\sum_{r=0}^{w}\sum_{s\in J_2^+(l,B,w)}\lambda_{(r,s)} \le x_{(2,l)},\quad \sum_{r=0}^{w}\sum_{s\in J_2^0(l,B,w)}\lambda_{(r,s)} \le 1-x_{(2,l)},$$
$$x_{(2,l)}\in\{0,1\}\quad \forall l\in L(w), \tag{14b}$$

where B is an SOS2 compatible function and $J_2^+(l,B,w)$, $J_2^0(l,B,w)$ are the specializations of $J^+(l,B)$, $J^0(l,B)$ for SOS2 constraints on $(\lambda_j)_{j=0}^{w}$. Constraints (14a) implement the independent SOS2 branching for the first coordinate and (14b) do the same for the second one.

To implement the second phase we use the branching scheme depicted in Figure 1(b) for the case $w = 4$. The dichotomy of this scheme is to select the triangles colored white in one branch and the ones colored gray in the other. For general w, this translates to forbidding the vertices (r, s) with r even and s odd in one branch (square vertices in the figure) and forbidding the vertices (r, s) with r odd and s even in the other (diamond vertices in the figure). This branching scheme selects exactly one triangle of every square in each branch and induces the set of constraints

$$\sum_{(r,s)\in L}\lambda_{(r,s)} \le x_0,\quad \sum_{(r,s)\in R}\lambda_{(r,s)} \le 1-x_0,\quad x_0\in\{0,1\}, \tag{15}$$

where $L = \{(r,s) \in J : r \text{ is even and } s \text{ is odd}\}$ and $R = \{(r,s) \in J : r \text{ is odd and } s \text{ is even}\}$. This formulation is illustrated by the following example.

Example 3 Constraints (14)–(15) for $w = 2$ are

$$\begin{aligned} \lambda_{(0,0)} + \lambda_{(0,1)} + \lambda_{(0,2)} \leq x_{(1,1)}, &\quad \lambda_{(2,0)} + \lambda_{(2,1)} + \lambda_{(2,2)} \leq 1 - x_{(1,1)} \\ \lambda_{(0,0)} + \lambda_{(1,0)} + \lambda_{(2,0)} \leq x_{(2,1)}, &\quad \lambda_{(0,2)} + \lambda_{(1,2)} + \lambda_{(2,2)} \leq 1 - x_{(2,1)} \\ \lambda_{(0,1)} + \lambda_{(2,1)} \leq x_0, &\quad \lambda_{(1,0)} + \lambda_{(1,2)} \leq 1 - x_0. \end{aligned}$$

A portion of the associated branching scheme is shown in Figure 2. The shaded triangles inside the nodes indicates the triangles forbidden by the corresponding assignment of the binary variables. The restriction to the first coordinate

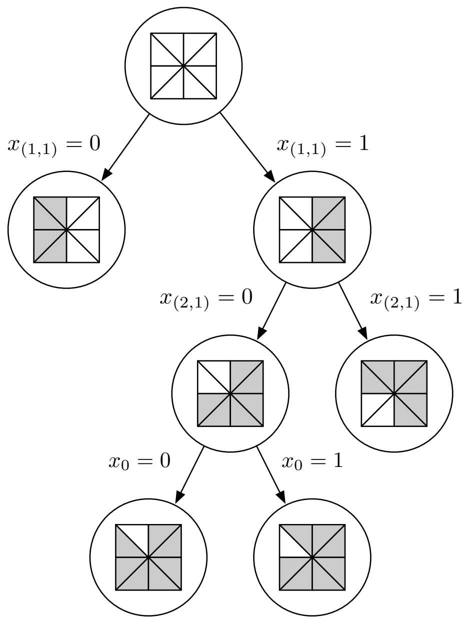


Fig. 2. Partial B&B tree from Example 3

of $[0, w]^2$ yields a logarithmic formulation for piecewise linear functions of one variable that only uses one of the SOS2 branchings and does not use the triangle selecting branching. Furthermore, under some mild assumptions, the model can be extended to non-uniform grids by selecting different values of v_j and to functions of 3 variables as well.

5 Computational Results

In this section we computationally test the logarithmic models for piecewise linear functions of one and two variables against some other existing models. For a set of transportation problems with piecewise linear cost functions, the logarithmic models provide a significant advantage in almost all of our experiments.

We denote the model for piecewise linear functions of one and two variables from section 4 by (Log). From the traditional models we selected the one usually denoted as the incremental model. This model for one variable functions appears as early as [19, 37, 20] and it has been recently shown to have favorable integrality and tightness properties [26, 28, 29]. The incremental model was extended to functions of several variables in [35]. We denote this model by (Inc). We also include two models that are based on independent branching schemes of linear depth. The first model is based on the independent branching scheme for SOS2 constraints on $(\lambda_j)_{j=0}^n$ given by $L_k = \{k, \ldots, n\}$, $R_k = \{0, \ldots, k\}$ for every $k \in \{1, \ldots, n-1\}$. This formulation has been independently developed in [32] and is currently defined only for functions of one variable. We denote this model by (LB1). The second model is based on an independent branching defined in [24, p. 573]. This branching scheme is defined for any triangulation and it has one branch for every vertex in the triangulation. In particular for piecewise linear functions of one variable with k intervals or segments the scheme has $k+1$ branches and for piecewise linear functions on a $k \times k$ grid it has $(k+1)^2$ branches. We denote the model by (LB2). We also tested some other piecewise linear models, but do not report results for them since they did not significantly improve the worst results reported here. All models were generated using Ilog Concert Technology and solved using CPLEX 9 on a dual 2.4GHz Linux workstation with 2GB of RAM. Furthermore, all tests were run with a time limit of 10000 seconds.

The first set of experiments correspond to piecewise linear functions of one variable for which we used the transportation models from [25]. We selected the instances with 10 supply and 10 demand nodes and for each of the 5 available instances we generated several randomly generated objective functions. We generated a separable piecewise linear objective function given by the sum of concave non-decreasing piecewise linear functions of the flow in each arc. For each instance and number of segments we generated 20 objective functions to obtain a total of 100 instances for each number of segments. Tables 1(a), 1(b) and 1(c) show the minimum, average, maximum and standard deviation of the solve times in seconds for 4, 8 and 16 segments. The tables also shows the number of times the solves failed because the time limit was reached and the number of times each formulation had the fastest solve time. As a final test for the one variable functions we tested the 3 best models on 100 instances with functions with 32 segments. Table 1(d) presents the statistics for these instances. For 16 and 32 segments we excluded the "wins" row as (Log) had the fastest solve times for every instance. The next set of experiments correspond to piecewise linear functions of two variables and we again used the 10×10 transportation models from [25]. In this case we took two copies of the same transportation model for each instance. We used an objective function which is the sum over all the arcs in the original transportation problem of non-separable two variable piecewise linear functions of the flows in the two copies of the arc. For each arc we generated the corresponding two variable piecewise linear function by triangulating a domain of the form $[0, w]^2$ as described in section 4 with a 8×8 segment grid to obtain a total of 128 triangles with 81 vertices. We then interpolated on this grid the functions of the two flows x_e, $x_{e'}$ given by $\sqrt{(a_1 x_e + b_1)(a_2 x_{e'} + b_2)}$ for $a_1, b_1, a_2, b_2 \in \mathbb{R}_+$ randomly generated independently for each arc. In addition, we eliminated the supply constraints from the two copies of the transportation problem to make the instances easier to solve. These problems were not created with a realistic application in mind and are just a simple extension of the problems in the previous set designed to include two variable non-separable functions. We again generated 20 objective functions for each of the original instances for a total of 100 instances. We excluded formulation (LB1) in this second set of tests as it is only valid for functions of one variable. Table 2(a) shows the statistics for this set of instances. As a final experiment we generated a new set of problems using a 16×16 grid for the interpolation obtaining a total of 512 triangles and 289 vertices. For these instances we only used formulations (Log) and (LB2). Table 2(b) shows the statistics for this last set of instances. It is clear that one of the advantages of the (Log) formulation is that it is smaller than the other

stat	(Log)	(LB1)	(LB2)	(Inc)
min	0	0	1	1
avg	2	2	4	3
max	9	9	27	16
std	1	1	3	2
fails	0	0	0	0
wins	72	27	0	1

(a) 4 segments.

stat	(Log)	(LB1)	(LB2)	(Inc)
min	1	0	1	0
avg	8	19	88	44
max	44	162	1171	245
std	8	19	147	36
fails	0	0	0	0
wins	98	1	1	0

(b) 8 segments.

stat	(Log)	(LB1)	(LB2)	(Inc)
min	1	13	15	46
avg	19	127	3561	374
max	83	652	10000	1859
std	17	105	3912	338
fails	0	0	21	0

(c) 16 segments.

stat	(Log)	(LB1)	(Inc)
min	3	113	182
avg	33	880	1445
max	174	10000	8580
std	33	1289	1327
fails	0	1	0

(d) 32 segments.

Table 1. Solve times for one variable functions [s].

stat	(Log)	(LB2)	(Inc)
min	1	3	95
avg	11	78	3521
max	102	967	10000
std	15	140	3648
fails	0	0	19
wins	99	1	0

(a) 8×8 grid.

stat	(Log)	(LB2)
min	5	22
avg	374	2910
max	10000	10000
std	1057	3444
fails	1	11
wins	98	2

(b) 16×16 grid.

Table 2. Solve times for two variable functions on a 8×8 and 16×16 grids [s].

formulations while retaining favorable tightness properties. In addition, formulation (Log) effectively transforms CPLEX's binary variable branching into a specialized branching scheme for piecewise linear functions. This allows formulation (Log) to combine the favorable properties of specialized branching schemes and the technology in CPLEX's variable branching. This last property is what probably allows (LB1) and (LB2) to outperform (Inc) too. In this regard we would like to emphasise the fact that all models tested are pure mixed integer programming problems (i.e. they do not include high level SOS2 constraints). Although CPLEX allows SOS2 high level descriptions and can use specialized SOS2 branching schemes that do not use binary variables the performance of these features for CPLEX 9 was inferior to most binary models we tested (including all for which results are presented here). Preliminary tests with CPLEX 11 show that these features have been considerably improved, which could make them competitive with the binary models. It is clear that formulation (Log) is superior to all of the others and that its advantage increases as the number of segments grows.

6 Conclusions

We have introduced a technique for modeling hard combinatorial problems with a mixed 0-1 integer programing formulation that uses a logarithmic number of binary variable and extra constraints. It is based on the concept of independent branching which is closely related to specialized branching schemes for combinatorial optimization. Using this technique we have introduced the first binary formulations for SOS1 and SOS2 constraints and for one and two variable piecewise linear functions that use a logarithmic number of binary variables and extra constraints. Finally, we have illustrated the usefulness of these new formulations by showing that for one and two variable piecewise linear functions they provide a significant computational advantage.

There are still a number of unanswered questions concerning necessary and sufficient conditions for the existence of formulations with a logarithmic number of binary variables and extra constraints. Simple examples show that it may not always be possible to obtain such a model. Moreover, if we allow the formulation to have a number of binary variables and extra constraints whose asymptotic growth is logarithmic our sufficient conditions do not seem to be necessary. Consider cardinality constraints that restrict at most K components of $\lambda \in [0,1]^n$ to be non-zero. This constraint does not satisfy the sufficient conditions but it does have a formulation with a number of variables and constraints of logarithmic order. We can write cardinality constraints in the form (5) by letting $J = \{1, \ldots, n\}$, $I = \{1, \ldots, m\}$ for $m = \binom{n}{K}$ and $\{S_j\}_{j=1}^m$ be the family of all subsets of J such that $|S_i| = K$. The traditional formulation for cardinality constraints is [19, 20]

$$\sum_{j=1}^{n} x_j \leq K; \quad \lambda_j \in [0,1], \quad \lambda_j \leq x_j, \quad x_j \in \{0,1\} \quad \forall j \in J. \tag{16}$$

Let n be an even number. By choosing $K = n/2$, which is the non-trivial cardinality constraint with the largest number of sets S_i, we can use the fact that for $K = n/2$ we have $n \leq 2\log_2\left(\binom{n}{K}\right)$ to conclude that (16) has $O(\log_2(|I|))$ binary variables and extra constraints.

References

1. Jeroslow, R.G.: Representability in mixed integer programming 1: characterization results. Discrete Applied Mathematics **17** (1987) 223–243
2. Jeroslow, R.G., Lowe, J.K.: Modeling with integer variables. Mathematical Programming Study **22** (1984) 167–184
3. Lowe, J.K.: Modelling with Integer Variables. PhD thesis, Georgia Institute of Technology (1984)
4. Balas, E.: Disjunctive programming. Annals of Discrete Mathematics **5** (1979) 3–51
5. Balas, E.: Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. SIAM Journal on Algebraic and Discrete Methods **6** (1985) 466–486
6. Balas, E.: Disjunctive programming: Properties of the convex hull of feasible points. Discrete Applied Mathematics **89** (1998) 3–44
7. Blair, C.: 2 rules for deducing valid inequalities for 0-1 problems. SIAM Journal on Applied Mathematics **31** (1976) 614–617
8. Jeroslow, R.G.: Cutting plane theory: disjunctive methods. Annals of Discrete Mathematics **1** (1977) 293–330
9. Sherali, H.D., Shetty, C.M.: Optimization with Disjunctive Constraints. Volume 181 of Lecture Notes in Economics and Mathematical Systems. Springer-Verlag (1980)
10. Balas, E.: On the convex-hull of the union of certain polyhedra. Operations Research Letters **7** (1988) 279–283
11. Blair, C.: Representation for multiple right-hand sides. Mathematical Programming **49** (1990) 1–5
12. Jeroslow, R.G.: A simplification for some disjunctive formulations. European Journal of Operational Research **36** (1988) 116–121
13. Garfinkel, R.S., Nemhauser, G.L.: Integer Programming. John Wiley & Sons, Inc. (1972)
14. Watters, L.J.: Reduction of integer polynomial programming problems to zero-one linear programming problems. Operations Research **15** (1967) 1171–1174
15. Ibaraki, T.: Integer programming formulation of combinatorial optimization problems. Discrete Mathematics **16** (1976) 39–52
16. Nemhauser, G.L., Wolsey, L.A.: Integer and combinatorial optimization. Wiley-Interscience (1988)
17. Balas, E.: Projection, lifting and extended formulation in integer and combinatorial optimization. Annals of Operations Research **140** (2005) 125–161
18. Beale, E.M.L., Tomlin, J.A.: Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. In Lawrence, J., ed.: OR 69: Proceedings of the fifth international conference on operational research, Tavistock Publications (1970) 447–454
19. Dantzig, G.B.: On the significance of solving linear-programming problems with some integer variables. Econometrica **28** (1960) 30–44

20. Markowitz, H.M., Manne, A.S.: On the solution of discrete programming-problems. Econometrica **25** (1957) 84–110
21. de Farias Jr., I.R., Johnson, E.L., Nemhauser, G.L.: Branch-and-cut for combinatorial optimization problems without auxiliary binary variables. The Knowledge Engineering Review **16** (2001) 25–39
22. Keha, A.B., de Farias, I.R., Nemhauser, G.L.: Models for representing piecewise linear cost functions. Operations Research Letters **32** (2004) 44–48
23. Keha, A.B., de Farias, I.R., Nemhauser, G.L.: A branch-and-cut algorithm without binary variables for nonconvex piecewise linear optimization. Operations Research **54** (2006) 847–858
24. Martin, A., Moller, M., Moritz, S.: Mixed integer models for the stationary case of gas network optimization. Mathematical Programming **105** (2006) 563–582
25. Vielma, J.P., Keha, A.B., Nemhauser, G.L.: Nonconvex, lower semicontinuous piecewise linear optimization. Discrete Optimization **5** (2008) 467–488
26. Croxton, K.L., Gendron, B., Magnanti, T.L.: A comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems. Management Science **49** (2003) 1268–1273
27. Magnanti, T.L., Stratila, D.: Separable concave optimization approximately equals piecewise linear optimization. In Bienstock, D., Nemhauser, G.L., eds.: IPCO. Volume 3064 of Lecture Notes in Computer Science., Springer (2004) 234–243
28. Padberg, M.: Approximating separable nonlinear functions via mixed zero-one programs. Operations Research Letters **27** (2000) 1–5
29. Sherali, H.D.: On mixed-integer zero-one representations for separable lower-semicontinuous piecewise-linear functions. Operations Research Letters **28** (2001) 155–160
30. Wilf., H.S.: Combinatorial algorithms–an update. Volume 55 of CBMS-NSF regional conference series in applied mathematics. Society for Industrial and Applied Mathematics (1989)
31. Appleget, J.A., Wood, R.K.: Explicit-constraint branching for solving mixed-integer programs. In Laguna, M., González Velarde, J.L., eds.: Computing tools for modeling, optimization, and simulation: interfaces in computer science and operations research. Volume 12 of Operations research / computer science interfaces series. Kluwer Academic (2000) 245–261
32. Shields, R. personal communication (2007)
33. Lee, J., Wilson, D.: Polyhedral methods for piecewise-linear functions I: the lambda method. Discrete Applied Mathematics **108** (2001) 269–285
34. Tomlin, J.: A suggested extension of special ordered sets to non-separable non-convex programming problems. In Hansen, P., ed.: Studies on Graphs and Discrete Programming. Volume 11 of Annals of Discrete Mathematics., North Holland (1981) 359–370
35. Wilson, D.: Polyhedral methods for piecewise-linear functions. PhD thesis, University of Kentucky (1998)
36. Todd, M.J.: Union jack triangulations. In Karamardian, S., ed.: Fixed Points: algorithms and applications. Academic Press (1977) 315–336
37. Dantzig, G.B.: Linear Programming and Extensions. Princeton University Press (1963)