Noname manuscript No.
(will be inserted by the editor)

Modeling Disjunctive Constraints with a Logarithmic Number of Binary Variables and Constraints

Juan Pablo Vielma · George L. Nemhauser

the date of receipt and acceptance should be inserted later

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 a finite number of specially structured polyhedra. Known mixed integer binary formulations for these constraints have a number of binary variables and extra constraints linear in the number of polyhedra. We give sufficient conditions for constructing formulations for these constraints with a number of binary variables and extra constraints logarithmic in the number of polyhedra. Using these conditions we introduce mixed integer binary formulations for SOS1 and SOS2 constraints that have a number of binary variables and extra constraints 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 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

Since the 1957 paper by Dantzig [15], the issue of modeling problems as mixed integer programs (MIPs) has been extensively studied. A study of the problems that can be modeled as MIPs began with Meyer [35–38] and was continued by Jeroslow and Lowe [21,23–25, 31].

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

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

An extended abstract of this paper appeared in [47]

J. P. Vielma · G. L. Nemhauser
H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, USA. E-mail: {jvielma, gnemhaus}@isye.gatech.edu

where $P_i = \{z \in \mathbb{R}^n : A^i z \leq b^i\}$ and I is a finite index set, can be modeled as a binary integer program. Jeroslow and Lowe [21,24,31] 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. That is, the directions of unboundedness of the polyheda given by $\{z \in \mathbb{R}^n : A^i z \leq 0\}$ for $i \in I$ are all equal. Using results from disjunctive programming [3,4,6,9,20,41] 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 [5,10,22], 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. 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 $|I|$. Another example of a model with a logarithmic number of variables is the work in [28], which also considers polytopes of the form $P_i := \{i\}$ to model different choices from an abstract set I. This work is used in [29] to model edge coloring problems by using $I = \{\text{possible colors}\}$.

Although (4) appears in the mathematical programming literature as early as [48], 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 [4] and in [19], we are not aware of any formulation with a logarithmic number of binary variables and extra constraints for the case in which each polyhedron P_i contains more than one point.

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. More constraints might provide a tighter LP relaxation and more variables might do the same by exploiting the favorable properties of projection [7]. 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 study the extension of the formulations from Sections 2 and 3 to a slightly different class of constraints and study the strength of these formulations. In Section 6 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 7 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 such that $|I|$ is a power of two, $\Delta^J := \{\lambda \in \mathbb{R}_+^{|J|} : \sum_{j\in J} \lambda_j \leq 1\}$ is the $|J|$-dimensional simplex in $\mathbb{R}^{|J|}$, $S_i \subset J$ for each $i \in I$ and

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

Furthermore, without loss of generality we assume that $\bigcup_{i\in I} S_i = J$. Since $Q(S_i)$ is a face of Δ^J we call Δ^J the *ground set* of the constraint. Except for Theorem 5, our results easily extend to the case in which the simplex is replaced by a box in $\mathbb{R}_+^{|J|}$, but the restriction to Δ^J greatly simplifies the presentation. We will study this extension in Section 5. We finally note that the requirement of $|I|$ being a power of two is without loss of generality as we can always add $2^{\lceil \log_2 |I| \rceil} - |I|$ polyhedra $Q(S_i)$ with $S_i = \emptyset$ to (5). We study the implications of this completion on formulation sizes in Section 3.

Disjunctive constraint (5) includes SOS1 and SOS2 constraints [8] 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 $I = J = \{1,\ldots,n\}$ 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 $I = \{1,\ldots,n\}$, $J = \{0,\ldots,n\}$ 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 [16,33], and some recent research has focused on branch-and-cut algorithms that do not use binary variables [18,26,27,34]. However, the incentive of being able to use state of the art MIP solvers (see for example the discussion in section 5 of [46]) makes binary models for these constraints very attractive [14,32,39,40].

We first review a formulation for (5) with a linear number of binary variables and 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

$$\lambda \in \Delta^J, \quad \lambda_j \leq \sum_{i\in I(j)} x_i \quad \forall j \in J, \quad \sum_{i\in I} x_i = 1, \quad x_i \in \{0,1\} \quad \forall i \in I \tag{7}$$

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 standard formulations for SOS1 and SOS2 constraints. (We consider the inequalities of ground set Δ^J as the original constraints and disregard the bounds on x.)

The following theorem shows that by using techniques from [19] we can obtain a formulation with $\log_2 |I|$ binary variables and $|I|$ extra constraints. Let $L(r) := \{1, \dots, \log_2 r\}$.

Theorem 1 *Let $B : I \to \{0,1\}^{\log_2 |I|}$ be any bijective function and $\sigma(B)$ be the support of vector B. Then*

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

is a valid formulation for (5).

Proof The formulation simply fixes λ_j to zero for all $j \notin S_i$ when x takes the value $B(i)$. □

The following example illustrates formulation (8) for SOS1 and SOS2 constraints.

Example 1 Let $J = \{1, \dots, 4\}$, $(\lambda_j)_{j=1}^4 \in \Delta^J$ be SOS1 constrained 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

$$\begin{aligned} \lambda \in \Delta^J, \quad x_1, x_2 \in \{0,1\}, \quad & \lambda_2 + \lambda_3 + \lambda_4 \le 2 - x_1 - x_2, \quad \lambda_1 + \lambda_3 + \lambda_4 \le 1 - x_1 + x_2, \\ & \lambda_1 + \lambda_2 + \lambda_4 \le 1 + x_1 - x_2, \quad \lambda_1 + \lambda_2 + \lambda_3 \le x_1 + x_2. \end{aligned}$$

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

$$\begin{aligned} \lambda \in \Delta^J, \quad x_1, x_2 \in \{0,1\}, \quad & \lambda_2 + \lambda_3 + \lambda_4 \le 2 - x_1 - x_2, \quad \lambda_0 + \lambda_3 + \lambda_4 \le 1 - x_1 + x_2, \\ & \lambda_0 + \lambda_1 + \lambda_4 \le 1 + x_1 - x_2, \quad \lambda_0 + \lambda_1 + \lambda_2 \le x_1 + x_2. \end{aligned}$$

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

Theorem 2 *Let $B : I \to \{0,1\}^{\log_2 |I|}$ be any bijective function. Then*

$$\lambda \in \Delta^J, \sum_{j \in J^+(l,B)} \lambda_j \le x_l, \sum_{j \in J^0(l,B)} \lambda_j \le (1 - x_l) \quad \forall j \in J, \quad x_l \in \{0,1\} \quad \forall l \in L(|I|), \tag{9}$$

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

Proof For SOS1 constraints we have $I = J = \{1, \dots, n\}$ and $S_j = \{j\}$ for each $i \in I$. This implies that $I(j) = \{j\}$ and hence $J^+(l,B) = \{j \in J : l \in \sigma(B(j))\}$ and $J^0(l,B) = \{j \in J : l \notin \sigma(B(j))\}$. Then, in formulation (9), we have that $\lambda_j = 0$ for all $x \neq B(j)$. □

The following example illustrates formulation (9) for SOS1 constraints.

Example 2 Let $J = \{1, \ldots, 4\}$, $(\lambda_j)_{j=1}^4 \in \Delta^J$ be SOS1 constrained. Formulation (9) for this case with $B = B^*$ from Example 1 is

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

We don't know how to give meaning to the binary variables in formulation (8) because fixing them individually has little effect on the λ variables. For example fixing $x_1 = 1$ and letting x_2 be free in either of the formulations of Example 1 has no effect on the λ variables. In contrast fixing $x_l = 1$ individually in (9) has the very precise effect fixing to zero all λ_j's for which $B(i)_l = 0$ for all i such that $j \in S_i$. Analogously, fixing $x_l = 0$ individually in (9) fixes to zero all λ_j's for which $B(i)_l = 1$ for all i such that $j \in S_i$. Fixing the binary variables then gives a way of enforcing $\lambda \in Q(S_i)$ by systematically fixing certain λ variables to zero.

Formulation (9) 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 (9) can be invalid for some choices of B. This is illustrated by the following example.

Example 3 Let $J = \{0, \ldots, 4\}$ and $(\lambda_j)_{j=0}^4 \in \Delta^J$ be SOS2 constrained. Formulation (9) 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 the feasible solution $\lambda_0 = 1/2$, $\lambda_2 = 1/2$, $\lambda_1 = \lambda_3 = \lambda_4 = 0$, $x_1 = x_2 = 1$ that does not comply with SOS2 constraints. However, the formulation can be made valid by adding constraints

$$\lambda_2 \le x_1 + x_2, \quad \lambda_2 \le 2 - x_1 - x_2. \tag{10}$$

For any B we can always correct formulation (9) 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) A function $B : \{1, \ldots, n\} \to \{0,1\}^{\log_2(n)}$ is *compatible* with an SOS2 constraint on $(\lambda_j)_{j=0}^n \in \mathbb{R}_+^{n+1}$ if it is bijective and for all $i \in \{1, \ldots, n-1\}$ the vectors $B(i)$ and $B(i+1)$ differ in at most one component.

Theorem 3 *If B is an SOS2 compatible function then* (9) *is valid for SOS2 constraints.*

Proof For SOS2 constraints we have that $I = \{1, \ldots, n\}$, $J = \{0, \ldots, n\}$ and $S_i = \{i-1, i\}$ for each $i \in I$. This implies that $I(0) = \{1\}$ and $I(n) = \{n\}$. Then, in a similar way to the proof of Theorem 2 for SOS1 constraints, we have that for $j \in \{0, n\}$ formulation (9) imposes $\lambda_j = 0$ for all $x \neq B(j)$.

In contrast, for $j \in J \setminus \{0, n\}$ we have $I(j) = \{j, j+1\}$ and hence $J^+(l, B) = \{j \in J : l \in \sigma(B(j)) \cap \sigma(B(j+1))\}$ and $J^0(l, B) = \{j \in J : l \notin \sigma(B(j))$ and $l \notin \sigma(B(j+1))\}$. Using the fact that B is SOS2 compatible we have that, in formulation (9), $\lambda_j = 0$ for all $x \notin \{B(j), B(j+1)\}$. $\square$

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

Example 3 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 (9) with $B=B^0$ for the same SOS2 constraints is

$$\lambda \in \Delta^J, \quad x_1, x_2 \in \{0,1\}$$
$$\lambda_0+\lambda_1 \leq x_1, \quad \lambda_3+\lambda_4 \leq (1-x_1) \tag{11}$$
$$\lambda_2 \leq x_2, \quad \lambda_0+\lambda_4 \leq (1-x_2). \tag{12}$$

Finally, the following lemma shows that an SOS2 compatible function can always be constructed.

Lemma 1 *For any $n \in \mathbb{Z}_+$ there exists a compatible function for SOS2 constraints on $(\lambda)_{j=0}^n$.*

Proof We construct an SOS2 compatible function $\bar{B}:\{1,\ldots,2^r\} \to \{0,1\}^r$ inductively on r. The case $r=1$ follows immediately. Now assume that we have an SOS2 compatible function $\bar{B}:\{1,\ldots,2^r\} \to \{0,1\}^r$. We define $\tilde{B}:\{1,\ldots,2^{r+1}\} \to \{0,1\}^{r+1}$ as

$$\tilde{B}(i)_l := \begin{cases} \bar{B}(i)_l & \text{if } i \leq 2^r \\ \bar{B}(2^{r+1}-i+1)_l & \text{o.w.} \end{cases} \quad \forall l \in \{1,\ldots,r\}, \quad \tilde{B}(i)_{r+1} := \begin{cases} 1 & \text{if } i \leq 2^r \\ 0 & \text{o.w.} \end{cases},$$

which is also SOS2 compatible. $\square$

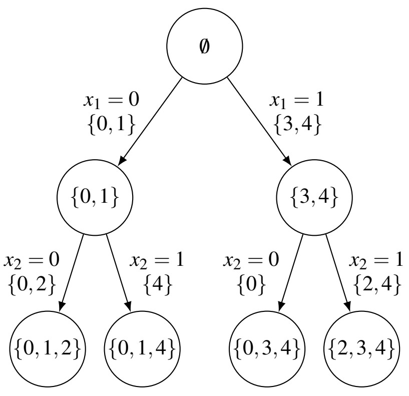
The function from the proof of Lemma 1 is not the only possible SOS2 compatible function. In fact, Definition 1 is equivalent to requiring $(B(i))_{i=1}^n$ to be a *reflected binary* or *Gray code* [49] and the construction from Lemma 1 corresponds to a version of this code that is usually called the *standard reflected Gray code*. Definition 1 is also equivalent to requiring $(B(i))_{i=1}^n$ to be a Hamiltonian path on the hypercube.

3 Branching and Logarithmic Size Formulations

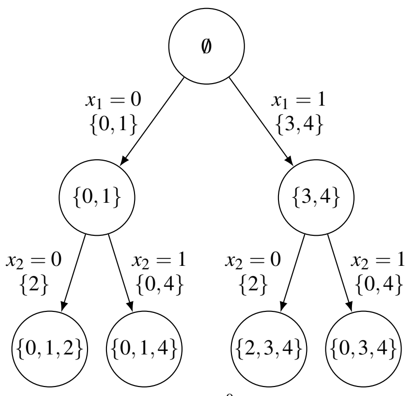
We have seen that fixing the binary variables of (9) provides a systematic procedure for enforcing $\lambda \in Q(S_i)$. In this section we exploit the relation between this procedure and specialized branching schemes to extend the formulation to a more general framework.

We can identify each vector in $\{0,1\}^{\log_2 |I|}$ with a leaf in a binary tree with $\log_2 |I|$ levels 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 $\log_2 |I|$ variables as the execution of a branching scheme on sets $Q(S_i)$. The formulations in Example 3 illustrate this idea.

In formulation (9) 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 [8] 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 level branching using linear inequalities. For B^0 the branch alternatives associated with x_2 are implemented by (12), which only include binary variable x_2. In contrast, for B^* one of the branching alternatives requires additional constraints (10) which involve both x_1 and x_2. The binary tree associated with the model for B^* and B^0 are shown in Figure 1, where the arc labels indicate the values taken by the binary variables and the indices of the λ variables which are fixed to zero because of this and the node labels indicate the indices of the λ variables that are set to zero because of the cumulative effect of the binary variable fixing. The main difference in the trees is that for $B = B^*$ the effect on the λ variables of fixing x_2 to a particular value depends on the value previously assigned to x_1 while for $B = B^0$ this effect is independent of the previous assignment to x_1.



(a) $B = B^*$.



(b) $B = B^0$.

Fig. 1 Two level binary trees for example 3.

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

$$\bigcup_{i\in I} Q(S_i) = \bigcap_{k=1}^{d} \big(Q(L_k) \cup Q(R_k)\big). \tag{13}$$

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

$$\lambda \in \Delta^J, \quad \sum_{j\notin L_k} \lambda_j \le x_k, \quad \sum_{j\notin R_k} \lambda_j \le (1 - x_k), \quad x_k \in \{0,1\} \quad \forall k \in L(|I|) \tag{14}$$

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

Formulation (9) with $B = B^0$ in Example 3 illustrates how an SOS2 compatible function induces an independent branching scheme for SOS2 constraints. In general, given an SOS2 compatible function $B : \{1,\dots,n\} \to \{0,1\}^{\log_2 n}$ the induced independent branching is given by $L_k = J \setminus J^+(k,B)$, $R_k = J \setminus J^0(l,B)$ for all $k \in \{1,\dots,n\}$.

Formulation (14) in Theorem 4 can be interpreted as a way of implementing a specialized branching scheme using binary variables. Similar techniques for implementing specialized branching schemes have been given in [1] and [42], but the resulting models require at least a linear number of binary variables. To the best of our knowledge the first independent branching schemes of logarithmic depth for the case in which polytopes $Q(S_i)$ contain more than one point are the ones for SOS1 constraints from Theorem 2 and for SOS2 constraints induced by an SOS2 compatible function.

Formulation (14) 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 (14) [4,5,10,22]. Fortunately, as the following theorem shows, the restriction to Δ^J makes (14) as tight as any other mixed integer formulation for (5).

Theorem 5 *Let P_λ and Q_λ be the projection onto the λ variables of the LP relaxation of formulation* (14) *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 $P_\lambda \subseteq Q_\lambda$.*

Proof Without loss of generality $\bigcup_{i\in I} S_i = J$ and hence for every $j \in J$ there is a $i \in I$ such that $j \in S_i$. Using this, it follows that $P_\lambda = \Delta^J = \operatorname{conv}\left(\bigcup_{i\in I} Q(S_i)\right)$. The relation with other mixed integer programming formulations follows directly from Theorem 3.1 of [24]. □

Theorem 5 might not be true if we do not use ground set Δ^J, but this restriction is not too severe as it includes a popular way of modeling piecewise linear functions. We explore this modeling in Section 4 and the potential loss of Theorem 5 when using a different ground set in Section 5.

We finally study the effect on formulation (14) of dropping the assumption that $|I|$ is a power of two. As mentioned in Section 2, if $|I|$ is not a power of two we can complete I to an index set of size $2^{\lceil \log_2 |I| \rceil}$ without changing (5). If we now construct a formulation that is of logarithmic size with respect to the completed index set we obtain a formulation that is still of logarithmic order with respect to the original index set. For instance, if I is not a power of two we can complete it and apply Theorem 1 to obtain a formulation with $\lceil \log_2 |I| \rceil < \log_2 |I| + 1$ binary variables and $2^{\lceil \log_2 |I| \rceil} < 2|I|$ extra constraints with respect to the original index set I. This is illustrated in the following example.

Example 4 Let $J = \{1,\dots,3\}$, $(\lambda_j)_{j=1}^3 \in \Delta^J$ be SOS1 constrained. In this case $I = \{1,\dots,3\}$ and $S_i = \{i\}$ for all $i \in I$. We can complete I so that $|I|$ is a power of two by letting $I = \{1,\dots,4\}$. We then set $S_4 = \emptyset$ to avoid adding new feasible solutions. Using $B = B^*$ from example 1 formulation (8) for the completed I is

$$\begin{aligned} \lambda \in \Delta^J, \quad x_1, x_2 \in \{0,1\}, \quad \lambda_2 + \lambda_3 &\le 2 - x_1 - x_2, & \lambda_1 + \lambda_3 &\le 1 - x_1 + x_2, \\ \lambda_1 + \lambda_2 &\le 1 + x_1 - x_2, & \lambda_1 + \lambda_2 + \lambda_3 &\le x_1 + x_2. \end{aligned}$$

We can alternatively think of this formulation as being obtained by setting $\lambda_4 = 0$ in the formulation for SOS1 constraints over $(\lambda_j)_{j=1}^4 \in \Delta^J$ given in Example 1.

Formulation (14) deals with the requirement that $|I|$ is a power of two somewhat differently. It is clear that (14) does not have this requirement explicitly as it only needs the existence of an independent branching scheme. Fortunately, if a family of constraints has an independent branching scheme when $|I|$ is a power of two we can easily construct an independent branching scheme for the cases in which $|I|$ is not a power of two. This is illustrated in the following example.

Example 5 Let $\{\overline{L}_k,\overline{R}_k\}_{k=1}^{\lceil \log_2 n\rceil}$ be an independent branching scheme for an SOS2 constraint on $(\lambda_j)_{j=0}^{\overline{n}} \in \Delta^{\overline{J}}$ for $\overline{n} := 2^{\lceil \log_2 n\rceil}$ and $\overline{J} = \{0,\dots,\overline{n}\}$. Then $\{L_k,R_k\}_{k=1}^{\lceil \log_2 n\rceil}$ defined by

$$L_k := \overline{L}_k \cap \{0,\dots,n\}, \quad R_k := \overline{R}_k \cap \{0,\dots,n\} \quad \forall k \in \{1,\dots,\lceil \log_2 n\rceil\} \tag{15}$$

is an independent branching scheme for an SOS2 constraint on $(\lambda_j)_{j=0}^{n} \in \Delta^J$ for $J = \{0,\dots,n\}$.

For example, for $n = 3$ and $\overline{n} = 4$, SOS2 compatible function B^0 from example 3 yields the independent branching scheme for SOS2 on $(\lambda_j)_{j=0}^{4} \in \Delta^{\overline{J}}$ given by $\overline{L}_1 := \{2,3,4\}$, $\overline{R}_1 := \{0,1,2\}$, $\overline{L}_2 := \{0,1,3,4\}$ and $\overline{R}_2 := \{1,2,3\}$. By restricting this scheme to $\{0,\dots,3\}$ we get the independent branching scheme for SOS2 on $(\lambda_j)_{j=0}^{3} \in \Delta^J$ given by $L_1 := \{2,3\}$, $R_1 := \{0,1,2\}$, $L_2 := \{0,1,3\}$ and $R_2 := \{1,2,3\}$. This scheme yields the following formulation of SOS2 on $(\lambda_j)_{j=0}^{3} \in \Delta^J$.

$$\begin{aligned}
&\lambda \in \Delta^J, \quad x_1,x_2 \in \{0,1\}\\
&\lambda_0+\lambda_1 \le x_1, \quad \lambda_3 \le (1-x_1)\\
&\lambda_2 \le x_2, \quad \lambda_0 \le (1-x_2).
\end{aligned}$$

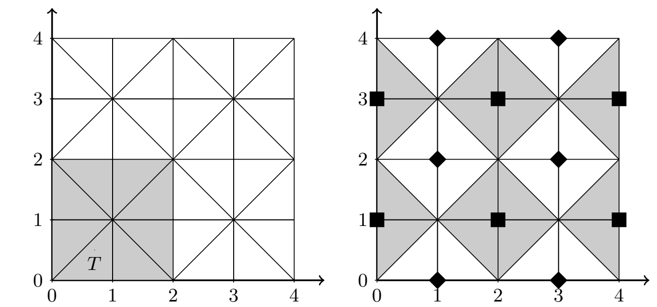
Note that this formulation can also be obtained by completing the constraint to $I = \{1,\dots,4\}$ by adding $S_4 = \emptyset$ and using formulation (9) for $B = B^0$ from example 3. We could show the validity of this procedure without referring to independent branching schemes by proving an analog to Theorem 3 for the case in which $|I|$ is not a power of two. A third alternative is to again think of this formulation as being obtained by setting $\lambda_4 = 0$ in the formulation for SOS2 constraints over $(\lambda_j)_{j=0}^{4} \in \Delta^J$ given in the continuation of Example 3.

4 Modeling Nonseparable Piecewise Linear Functions

In this section we use Theorem 4 to construct a model for non-separable piecewise linear functions of two variables that use a number of binary variables and extra constraints logarithmic in the number of linear pieces of the functions. We also extend this formulation to functions of n variables, in which case the formulation is slightly larger, but still asymptotically logarithmic for fixed n.

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 [26, 27, 30, 34, 44]. This approach has been extended to non-separable piecewise linear functions in [30, 34, 44, 50]. 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 [43] which is depicted in Figure 2(a) for $w = 4$. The $\mathbf{J}_1$ triangulation



(a) Example of “Union Jack” Triangulation
(b) Triangle selecting branching

Fig. 2 Triangulations

of $[0,w]^2$ for any even w is obtained by adding copies of the 8 triangles shaded gray in Figure 2(a). This yields a triangulation with $2w^2$ triangles.

We use this triangulation to approximate f with a piecewise 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 2(a), the vertices of the triangle labeled T are $S_T := \{(0,0),(1,0),(1,1)\}$. A valid model for $g(y)$ [30,34,44] is

$$\sum_{j\in J}\lambda_j = 1, \quad y=\sum_{j\in J} v_j\lambda_j, \quad g(y)=\sum_{j\in J} f(v_j)\lambda_j \tag{16a}$$

$$\lambda \in \bigcup_{i\in I} Q(S_i) \subset \Delta^J, \tag{16b}$$

where $J := \{0,\ldots,w\}^2$, $v_j = j$ for $j\in J$. This model becomes a traditional model for one variable piecewise linear functions when we restrict it to one coordinate of $[0,w]^2$ by setting $y_2 = 0$ and $\lambda_{(s,t)} = 0$ for all $0 \le s \le w$, $1 \le t \le w$.

To obtain a mixed integer formulation of (16) with a logarithmic number of binary variables and extra constraints it suffices to construct an independent binary branching scheme of logarithmic depth for (16b) and use formulation (14). Binary branching schemes for (16b) with a similar triangulation have been developed in [44] and [34], 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 [34,44] 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

$$\sum_{v_2=0}^{w}\sum_{v_1\in J_2^+(l,B,w)}\lambda_{(v_1,v_2)}\le x_l^1,\quad \sum_{v_2=0}^{w}\sum_{v_1\in J_2^0(l,B,w)}\lambda_{(v_1,v_2)}\le 1-x_l^1,$$
$$x_l^1\in\{0,1\}\quad \forall l\in L(w), \tag{17a}$$
$$\sum_{v_1=0}^{w}\sum_{v_2\in J_2^+(l,B,w)}\lambda_{(v_1,v_2)}\le x_l^2,\quad \sum_{v_1=0}^{w}\sum_{v_2\in J_2^0(l,B,w)}\lambda_{(v_1,v_2)}\le 1-x_l^2,$$
$$x_l^2\in\{0,1\}\quad \forall l\in L(w), \tag{17b}$$

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 (17a) and binary variables x_l^1 implement the independent SOS2 branching for the first coordinate and (17b) and binary variables x_l^2 do the same for the second one.

To implement the second phase we use the branching scheme depicted in Figure 2(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 (v_1,v_2) with v_1 even and v_2 odd in one branch (square vertices in the figure) and forbidding the vertices (v_1,v_2) with v_1 odd and v_2 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

$$\sum_{(v_1,v_2)\in L}\lambda_{(v_1,v_2)}\le y_0,\quad \sum_{(v_1,v_2)\in R}\lambda_{(v_1,v_2)}\le 1-y_0,\quad y_0\in\{0,1\}, \tag{18}$$

where $L=\{(v_1,v_2)\in J:\ v_1$ is even and v_2 is odd$\}$ and $R=\{(v_1,v_2)\in J:\ v_1$ is odd and v_2 is even$\}$. When w is a power of two the resulting formulation has exactly $\log_2\mathscr{T}$ binary variables and $2\log_2\mathscr{T}$ extra constraints where $\mathscr{T}$ is the number of triangles in the triangulation. We illustrate the formulation with the following example.

Example 6 Constraints (17)–(18) for $w=2$ are

$$\lambda_{(0,0)}+\lambda_{(0,1)}+\lambda_{(0,2)}\le x_{(1,1)},\quad \lambda_{(2,0)}+\lambda_{(2,1)}+\lambda_{(2,2)}\le 1-x_{(1,1)}$$
$$\lambda_{(0,0)}+\lambda_{(1,0)}+\lambda_{(2,0)}\le x_{(2,1)},\quad \lambda_{(0,2)}+\lambda_{(1,2)}+\lambda_{(2,2)}\le 1-x_{(2,1)}$$
$$\lambda_{(0,1)}+\lambda_{(2,1)}\le x_0,\quad \lambda_{(1,0)}+\lambda_{(1,2)}\le 1-x_0.$$

A portion of the associated branching scheme is shown in Figure 3. The shaded triangles inside the nodes indicates the triangles forbidden by the corresponding assignment of the binary variables.

The restriction to the first coordinate 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.

The extension of the formulation to functions of n variables is direct from the definition of the n-dimensional $\mathbf{J}_1$ triangulation [43]. For $D=[0,w]^n$ with w an even integer the vertex set of the triangulation is defined to be $\{0,\dots,w\}^n$ and the triangulation is composed by the finite family of simplices defined as follows. Let $N=\{1,\dots,n\}$, $\mathscr{V}^0=\{v\in\{0,\dots,w\}^n:$

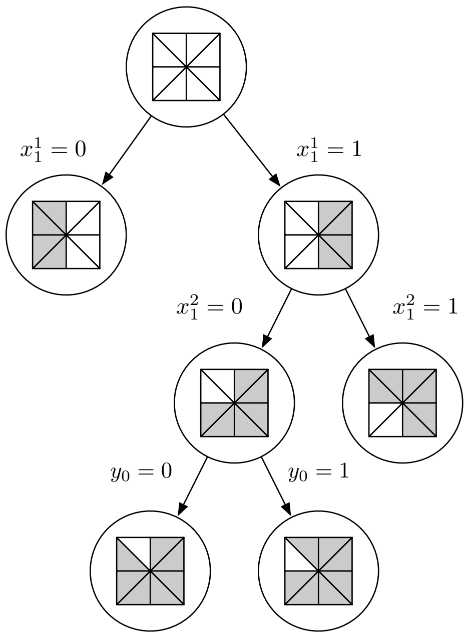


Fig. 3 Partial B&B tree for Example 6

v_i is odd, $\forall i \in N\}$, $\mathrm{Sym}(N)$ be the group of all permutations on N and e^i be the i-th unit vector of $\mathbb{R}^n$. For each $(v^0, \pi, s) \in \mathscr{V}^0 \times \mathrm{Sym}(N) \times \{-1,1\}^n$ we define $j_1(v^0, \pi, s)$ to be the simplex whose extreme points are $\{y^i\}_{i=0}^n$ where $y^i = y^{i-1} + s_{\pi(i)} e^{\pi(i)}$ for each $i \in N$. The $\mathbf{J}_1$ triangulation of $D = [0,w]^n$ is given by all the simplices $j_1(v^0, \pi, s)$. By letting $J = \{0, \dots, w\}^n$ and I be the set of triangles of the $\mathbf{J}_1$ of $D = [0,w]^n$ we have that (16) is a model for the piecewise linear approximation g of function $f : [0,w]^n \to \mathbb{R}$. For this case, to implement the independent branching scheme for (16b) we can use the fact that indices v^0 and s of the simplices determines the hypercube in which the simplex is contained and index π determines the selection of one of the $n!$ simplices contained in a given hypercube (For example for the triangulation in Figure 1(a), the simplices for $v^0 = (1,1)$ and $s = (-1,-1)$ are the two triangles contained in box $[0,1]^2$ and the triangle labeled T corresponds to the permutation $\pi(1) = 2$, $\pi(2) = 1$). Then to select the hypercube we can again apply independent SOS2 branching for each component which yields the constraints given by

$$\sum_{v \in \tilde{J}_2^+(l,B,w,k)} \lambda_v \le x_l^k, \qquad \sum_{v \in \tilde{J}_2^0(l,B,w,k)} \lambda_v \le 1 - x_l^k, \quad x_l^k \in \{0,1\} \quad \forall l \in L(w), \forall k \in N \tag{19}$$

where $\tilde{J}_2^+(l,B,w,k) = \{v \in J : v_k \in J_2^+(l,B,w)\}$ and $\tilde{J}_2^0(l,B,w,k) = \{v \in J : v_k \in J_2^0(l,B,w)\}$. To select a permutation π it suffices to select between $\pi^{-1}(r) < \pi^{-1}(s)$ or $\pi^{-1}(r) > \pi^{-1}(s)$ for each $r, s \in N$, $r < s$. If we select a permutation with $\pi^{-1}(r) < \pi^{-1}(s)$ we have that no vertex v of the resulting triangulation will have an odd v_r component and even v_s component. In contrast, if we select a permutation with $\pi^{-1}(s) < \pi^{-1}(r)$ we have that no vertex v of the resulting triangulation will have an even v_r component and odd v_s component. Hence to select a simplex if suffices to apply the triangle selection branching depicted in Figure 2(b) to each pair of indices $r,s \in N$, $r < s$ which yields the constraints given by

$$\sum_{v \in L(r,s)} \lambda_v \leq y_{(r,s)}, \quad \sum_{v \in R(r,s)} \lambda_v \leq 1 - y_{(r,s)}, \quad y_{(r,s)} \in \{0,1\} \quad \forall r,s \in N,\, r < s \tag{20}$$

where $L(r,s) = \{v \in J : \; v_r \text{ is even and } v_s \text{ is odd}\}$ and $R = \{v \in J : \; v_r \text{ is odd and } v_s \text{ is even}\}$. The resulting formulation has $L := n\lceil \log_2 w \rceil + n(n-1)/2$ binary variables (and twice as many extra constraints) and the $\mathbf{J}_1$ triangulation has $\mathscr{T} := w^n n!$ simplices. In contrast to the two dimensional case, it is not clear how to explicitly relate these two numbers even for the case when w is a power of two. However we can see that L grows asymptotically as $\log_2 \mathscr{T}$ only when n is fixed. More specifically, for fixed n we have $L \sim \log_2 \mathscr{T}$ (i.e. $\lim_{w\to\infty} L/\log_2 \mathscr{T} = 1$), but for fixed w we have $\log_2 \mathscr{T} \in o(L)$ (i.e. $\lim_{n\to\infty} \log_2 \mathscr{T}/L = 0$).

5 Extension of the Model to Ground Set $[0,1]^J$

We replace $\lambda \in \Delta^J$ in definition (6) of $Q(S_i)$ with the box constraint $\lambda \in [0,1]^J$ to obtain $\overline{Q}(S_i) = \{\lambda \in [0,1]^J : \lambda_j = 0 \forall j \notin S_i\}$. We have that an independent branching $\{L_k, R_k\}_{k=1}^d$ for (5) is also an independent branching for

$$\lambda \in \bigcup_{i \in I} \overline{Q}(S_i) \tag{21}$$

since

$$\bigcup_{i \in I} \overline{Q}(S_i) = \bigcap_{k=1}^{d} \left(\overline{Q}(L_k) \cup \overline{Q}(R_k)\right). \tag{22}$$

However, to preserve validity formulation (14) needs to be modified to

$$\lambda \in [0,1]^J \tag{23a}$$

$$\sum_{j \notin L_k} \lambda_j \leq |J \setminus L_k| x_k, \quad \sum_{j \notin R_k} \lambda_j \leq |J \setminus R_k| (1 - x_k), \quad x_k \in \{0,1\} \quad \forall k \in \{1,\ldots,d\}. \tag{23b}$$

This formulation still has d binary variables and $2d$ extra constraints, but Theorem 5 is no longer true for this formulation.

To understand the potential sources of weakness of formulation (23) we study how this formulation can be constructed from the standard disjunctive programming formulation of (21) in three steps, two of which have the potential for weakening the formulation. The first step is to use identity (22) to reduce the formulation of (21) to the formulation of

$$\lambda \in \overline{Q}(L_k) \cup \overline{Q}(R_k) \tag{24}$$

for each $k \in \{1,\ldots,d\}$. The second step is to eliminate the duplicated continuous variables of formulation (2) for (24) in the following way. Formulation (2) for (24) is given by

$$\lambda_j^{1,k}, \lambda^{2,k} \in \mathbb{R}_+^{|J|}, \; x_k \in \{0,1\} \tag{25a}$$

$$\lambda_j^{1,k} \leq (1 - x_k) \quad \forall j \in L_k, \quad \lambda_j^{1,k} \leq 0 \quad \forall j \notin L_k \tag{25b}$$

$$\lambda_j^{2,k} \leq x_k \quad \forall j \in R_k, \quad \lambda_j^{2,k} \leq 0 \quad \forall j \notin R_k \tag{25c}$$

$$\lambda = \lambda^{1,k} + \lambda^{2,k}. \tag{25d}$$

Using (25d) we can eliminate variables $\lambda^{1,k}, \lambda^{2,k}$ to obtain the formulation of (24) given by

$$\lambda \in [0,1]^J, \quad x_k \in \{0,1\} \tag{26a}$$
$$\lambda_j \le x_k \qquad \forall j \notin L_k \tag{26b}$$
$$\lambda_j \le (1-x_k) \qquad \forall j \notin R_k. \tag{26c}$$

The third and final step is to aggregate constraints (26b)–(26c) and combine the resulting formulation of (24) for all $k \in \{1,\dots,d\}$ to obtain (23).

With regard to the first step, we have that (22) shows how an independent branching scheme rewrites disjunctive constraint (5) from its *disjunctive normal form* (DNF) as the union of polyhedra (left hand side) to a conjunction of two-term polyhedral disjunctions (right hand side). It is well known that this rewrite can significantly reduce the tightness of mixed integer programming formulations [4]. More specifically, Theorem 3.1 of [24] tells us that if we directly formulate constraint (21) the best we can hope is for the projection onto the original λ variables of the LP relaxation of our formulation to be equal to $\operatorname{conv}(\bigcup_{i\in I}\overline{Q}(S_i))$. In contrast, if we construct a formulation for constraints (24) for each $k \in \{1,\dots,d\}$ and then combine them, the best we can hope is for the projection onto the original λ variables of the LP relaxation of our formulation to be equal to $\bigcap_{k=1}^d \operatorname{conv}(\overline{Q}(L_k)\cup\overline{Q}(R_k))$. Because the convex hull and intersection operations usually do not commute we only have

$$\operatorname{conv}\left(\bigcup_{i\in I}\overline{Q}(S_i)\right) \subset \bigcap_{k=1}^{d}\operatorname{conv}(\overline{Q}(L_k)\cup\overline{Q}(R_k)) \tag{27}$$

and we can expect strict containment resulting in the first formulation being stronger. This is illustrated in the following example.

Example 7 Let $J = \{0,\dots,4\}$ and $(\lambda_j)_{j=0}^4 \in \Delta^J$ be SOS2 constrained. We then have $S_1 = \{0,1\}$, $S_2 = \{1,2\}$, $S_3 = \{2,3\}$, $S_4 = \{3,4\}$ and using PORTA [12] we get that

$$\operatorname{conv}\left(\bigcup_{i=1}^{4}\overline{Q}(S_i)\right) = \Big\{(\lambda_j)_{j=0}^4 \in [0,1]^5 : \lambda_1+\lambda_4 \le 1, \quad \lambda_1+\lambda_3 \le 1, \quad \lambda_0+\lambda_3 \le 1,$$
$$\lambda_0+\lambda_2+\lambda_4 \le 1\Big\}. \tag{28}$$

If we let $d=1$, $L_1 = \{2,3,4\}$, $R_1 = \{0,1,2\}$, $L_2 = \{0,1,3,4\}$ and $R_2 = \{1,2,3\}$ we have $\bigcup_{i=1}^4 \overline{Q}(S_i) = \big(\overline{Q}(L_1)\cup\overline{Q}(R_1)\big) \cap \big(\overline{Q}(L_2)\cup\overline{Q}(R_2)\big)$. Again using PORTA we get that

$$\operatorname{conv}\big(\overline{Q}(L_1)\cup\overline{Q}(R_1)\big) = \Big\{(\lambda_j)_{j=0}^4 \in [0,1]^5 : \lambda_2 \le 1, \quad \lambda_1+\lambda_4 \le 1, \quad \lambda_1+\lambda_3 \le 1,$$
$$\lambda_0+\lambda_4 \le 1, \quad \lambda_0+\lambda_3 \le 1\}$$

and

$$\operatorname{conv}\big(\overline{Q}(L_2)\cup\overline{Q}(R_2)\big) = \Big\{(\lambda_j)_{j=0}^4 \in [0,1]^5 : \lambda_3 \le 1, \quad \lambda_1 \le 1, \quad \lambda_2+\lambda_4 \le 1,$$
$$\lambda_0+\lambda_2 \le 1, \quad \lambda_3+\lambda_4-\lambda_0-\lambda_1 \le 1\}.$$

Clearly $(1/2,1/2,1/2,1/2,1/2) \in \operatorname{conv}(Q(L_1)\cup Q(R_1)) \cap \operatorname{conv}(Q(L_2)\cup Q(R_2))$, but from (28) we have $(1/2,1/2,1/2,1/2,1/2) \notin \operatorname{conv}(\bigcup_{i=1}^4 Q(S_i))$. Hence

$$\operatorname{conv}\left(\bigcup_{i=0}^{4}\overline{Q}(S_i)\right) \subsetneq \bigcap_{k=1}^{2}\operatorname{conv}(\overline{Q}(L_k)\cup\overline{Q}(R_k)).$$

This source of weakness could be avoided by applying techniques from [4] at the expense of increasing the number of continuous variables.

With respect to the second step, it is well known that eliminating the multiple copies of the continuous variables in formulation (2) can result in a weaker formulation [5,10, 22]. Fortunately, as the following theorem shows, for constraints of the form (5) or (21) eliminating the multiple copies of the continuous variables does not make the formulations weaker.

Theorem 6 *Let P_λ be the projection onto the λ variables of the LP relaxation of formulation* (7) *for* (5) *and let $\overline{P}_\lambda$ the projection onto the λ variables of the LP relaxation of the formulation of* (21) *given by*

$$\lambda \in [0,1]^J, \quad \lambda_j \le \sum_{i \in I(j)} x_i \quad \forall j \in J, \quad \sum_{i \in I} x_i = 1, \quad x_i \in \{0,1\} \quad \forall i \in I. \tag{29}$$

Then $P_\lambda = \operatorname{conv}\left(\bigcup_{i\in I} Q(S_i)\right)$ and $\overline{P}_\lambda = \operatorname{conv}\left(\bigcup_{i\in I} \overline{Q}(S_i)\right)$. In particular the projections onto the λ variables of the LP relaxations of formulations (25) *and* (26) *are equal to* $\operatorname{conv}(\overline{Q}(L_k) \cup \overline{Q}(R_k))$.

Proof For P_λ the result follows directly from Theorem 5. For $\overline{P}_\lambda$ the result follows directly from Section 3.1 of [22] because $\bigcup_{i\in I} \overline{Q}(S_i)$ is the union of multidimensional intervals as defined in that section. □

Theorem 6 shows that the traditional formulations for SOS1 and SOS2 constraints are as tight as possible, which could explain their success. In addition, Theorem 6 shows that the second step does not weaken the formulation as we get the following corollary.

Corollary 1 *The projection onto the λ variables of the LP relaxation of the formulation given by* (26) *for all $k \in \{1,\dots,d\}$ is* $\bigcap_{k=1}^{d} \operatorname{conv}(\overline{Q}(L_k) \cup \overline{Q}(R_k))$.

Finally, with respect to the third step, it is well known that a weaker integer programming formulations can result from aggregating constraints. As expected it is also easy to construct examples where formulation (26) is stronger than formulation (23) (the example for the strict containment in (27) also works here). Of course, this source of weakness can be avoided by simply choosing formulation (26) instead of (23) at the expense of increasing the number of constraints from $2d$ to at most $|J|d$.

6 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 so called *incremental* and *multiple choice* models. The incremental model for one variable functions appears as early as [16,17, 33], was extended to functions of several variables in [50] and it has been recently shown to have favorable integrality and tightness properties [14,39,40]. We denote this model by Inc. The multiple choice model appears in [2,14,31] and also has favorable integrality and tightness properties. We denote this model by MC. 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 [42] 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 [34, p. 573]. This branching scheme is defined for any triangulation and its depth is equal to the number of vertices in the triangulation. In particular for piecewise linear functions of one variable with k intervals or segments its depth is $k+1$ and for piecewise linear functions on a $k \times k$ grid it is $(k+1)^2$. 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. We refer the reader to [45] for a more detailed study and evaluation of mixed integer formulations for piecewise linear functions. In addition to the mixed integer programming formulations we tested the traditional SOS2 formulation of univariate piecewise linear functions which does not include binary variables. We implemented this formulation using CPLEX's built in support for SOS2 constraints and we denote it by SOS2. All models were generated using Ilog Concert Technology and solved using CPLEX 11 on a dual 2.4GHz Xeon Linux workstation with 2GB of RAM. Furthermore, all tests were run with a time limit of 10000 seconds.

We note that Log, Inc, MC, LB1 and LB2 are mixed integer programming problems that do not include SOS2 constraints such as the ones supported by CPLEX. Hence, when CPLEX solves these formulations the only type of branching that occurs is due to the fixing of binary variables to zero or one. For Log, LB1 and LB2 this binary branching induces a specialized branching schemes that fixes some λ variables to zero, but CPLEX does not directly fix λ variables to zero. In contrast, formulation SOS2 does not contain any binary variables and to solve it CPLEX executes the traditional SOS2 branching of [8] by directly fixing λ variables to zero.

The first set of experiments correspond to piecewise linear functions of one variable for which we used the transportation models from [46]. 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. We use concave functions because they are widely used in practice and because using them results in NP-hard problems [26] that are challenging for our experiments. For each instance and number of segments we generated 20 objective functions to obtain a total of 100 instances for each number of segments. We excluded LB2 as LB1 performed consistently better. Table 1 shows the minimum, average, maximum and standard deviation of the solve times in seconds for 4, 8, 16 and 32 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 (win or tie). MC is the best model for 4 and 8 segments and Log is clearly the best model for 16 and 32 segments.

The next set of experiments correspond to piecewise linear functions of two variables for which we selected a series of two commodity transportation problems with 5 supply nodes and 2 demand nodes. These instances were constructed by combining two 5×2 transportation problems generated in a manner similar to the instances used in [46]. The supplies, demands and individual commodity arc capacities for each commodity were obtained from two different transportation problems and the joint arc capacities were set to $3/4$ of the sum of the corresponding individual arc capacities. We considered an objective function of the form $\sum_{e \in E} f_e(x_e^1, x_e^2)$ where E is the common set of 10 arcs of the transportation prob-

stat	Log	LB1	MC	Inc	SOS2
min	0	0	0	0	0
avg	**2**	**3**	**1**	**3**	**2**
max	12	16	8	15	8
std	2	3	2	3	1
wins	25	1	46	2	27
fail	0	0	0	0	0

(a) 4 segments.

stat	Log	LB1	MC	Inc	SOS2
min	1	3	1	5	1
avg	**12**	**26**	**10**	**47**	**16**
max	84	116	39	160	202
std	11	17	7	31	23
wins	34	0	43	0	23
fail	0	0	0	0	0

(b) 8 segments.

stat	Log	LB1	MC	Inc	SOS2
min	0	7	2	23	2
avg	**24**	**124**	**97**	**284**	**109**
max	96	376	730	1250	1030
std	18	78	122	201	167
wins	95	0	3	0	2
fail	0	0	0	0	0

(c) 16 segments.

stat	Log	LB1	MC	Inc	SOS2
min	2	117	23	214	10
avg	**43**	**569**	**2246**	**889**	**925**
max	194	2665	10000	3943	10000
std	39	476	3208	662	1900
wins	98	0	0	0	2
fail	0	0	9	0	2

(d) 32 segments.

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

lems and $f_e(x_e^1, x_e^2)$ is a piecewise linear function of the flows x_e^i in arc e of commodity i for $i = 1,2$. Each component $f_e(x_e^1, x_e^2)$ for arc e with individual arc capacities u_e^i for commodity $i = 1,2$ was constructed as follows. We begin by triangulating $[0,u_e^1] \times [0,u_e^2]$ as described in Section 4 with a $K \times K$ segment grid. Using this triangulation we then obtained $f_e(x_e^1, x_e^2)$ by interpolating $g\left(\left\|\left(x_e^1, x_e^2\right)\right\|\right)$ where $\|\cdot\|$ is the euclidean norm and $g : \left[0, \left\|\left(u_e^1, u_e^2\right)\right\|\right] \to \mathbb{R}$ is a continuous concave piecewise linear function which was randomly generated independently for each arc in a similar way to the one variable functions of the previous set of experiments. The idea of this function is to use the sub-linearity of the euclidean norm to consider discounts for sending the two commodities in the same arc and concave function g to consider economies of scale. We note that although g is concave its interpolation is not always concave due to the known fact that multivariate interpolation on a predefined triangulation is not always shape preserving [11]. We selected 5 combinations of different pairs of the original transportation problems and for each one of these we generated 20 objective functions for a total of 100 instances for each K. For these instances we excluded SOS2 and LB1 as they are only defined for univariate functions. Table 2 shows the statistics for this set of instances. In the two variable case, Log is best for all sizes and the advantage becomes overwhelming for the largest instances.

It is clear that one of the advantages of Log is that it is smaller than the other 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. Given its computational advantages, we anticipate that Log will become a valuable tool in practice.

stat	Log	LB2	MC	Inc
min	0	1	1	3
avg	**3**	**6**	**6**	**32**
max	9	22	17	127
std	2	4	3	26
wins	87	9	5	0
fail	0	0	0	0

(a) 4×4 grid.

stat	Log	LB2	MC	Inc
min	2	37	31	100
avg	**13**	**196**	**398**	**769**
max	33	804	5328	6543
std	5	129	584	1111
wins	100	0	0	0
fail	0	0	0	31

(b) 8×8 grid.

stat	Log	LB2	MC	Inc
min	27	3116	2853	772
avg	**56**	**9825**	**9266**	**4857**
max	118	10000	10000	10000
std	19	866	1678	3429
wins	100	0	0	0
fail	0	94	77	20

(c) 16×16 grid.

Table 2 Solve times for two variable functions on a 4×4, 8×8 and 16×16 grids [s].

7 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 more general sufficient conditions for the existence of formulations with a logarithmic number of binary variables and extra constraints. For example, 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. We do not know of an independent branching scheme for this constraint, 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 [16,33]

$$\sum_{j=1}^{n} x_j \le K; \quad \lambda_j \in [0,1], \quad \lambda_j \le x_j, \quad x_j \in \{0,1\} \quad \forall j \in J. \tag{30}$$

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 $n \le 2\log_2\left(\binom{n}{K}\right)$ to conclude that (30) has $O(\log_2(|I|))$ binary variables and extra constraints.

Another question concerns the case in which I is not a power of two. Theoretically, this does not pose a problem because we can complete I or adapt the independent branching scheme. However, preliminary tests in [45] showed that the computational effectiveness of independent branching schemes can be significantly reduced if I is not a power of two. This is a common problem with binary encoded formulations, that can be mitigated by the use of techniques developed in [13].

Acknowledgements 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. The authors would also like to thank two anonymous referees for some very thoughtful comments.

References

1. Appleget, J.A., Wood, R.K.: Explicit-constraint branching for solving mixed-integer programs. In: M. Laguna, J.L. González (eds.) Computing tools for modeling, optimization, and simulation: interfaces in computer science and operations research, *Operations research/computer science interfaces series*, vol. 12, pp. 245–261. Kluwer (2000)
2. Balakrishnan, A., Graves, S.C.: A composite algorithm for a concave-cost network flow problem. Networks **19**, 175–202 (1989)
3. Balas, E.: Disjunctive programming. Ann. Discrete Math. **5**, 3–51 (1979)
4. Balas, E.: Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. SIAM J. Algebraic Discrete Methods **6**, 466–486 (1985)
5. Balas, E.: On the convex-hull of the union of certain polyhedra. Oper. Res. Lett. **7**, 279–283 (1988)
6. Balas, E.: Disjunctive programming: Properties of the convex hull of feasible points. Discrete Appl. Math. **89**, 3–44 (1998)
7. Balas, E.: Projection, lifting and extended formulation in integer and combinatorial optimization. Ann. Oper. Res. **140**, 125–161 (2005)
8. 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: J. Lawrence (ed.) OR 69: Proceedings of the fifth international conference on operational research, pp. 447–454. Tavistock Publications (1970)
9. Blair, C.: 2 rules for deducing valid inequalities for 0-1 problems. SIAM J. Appl. Math. **31**, 614–617 (1976)
10. Blair, C.: Representation for multiple right-hand sides. Math. Program. **49**, 1–5 (1990)
11. Carnicer, J.M., Floater, M.S.: Piecewise linear interpolants to lagrange and hermite convex scattered data. Numer. Algorithms **13**, 345–364 (1996)
12. Christof, T., Loebel, A.: PORTA – POlyhedron Representation Transformation Algorithm, version 1.3. Available at `http://www.iwr.uni-heidelberg.de/groups/comopt/software/PORTA/`
13. Coppersmith, D., Lee, J.: Parsimonious binary-encoding in integer programming. Discrete Optim. **2**, 190–200 (2005)
14. Croxton, K.L., Gendron, B., Magnanti, T.L.: A comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems. Manage. Sci. **49**, 1268–1273 (2003)
15. Dantzig, G.B.: Discrete-variable extremum problems. Oper. Res. **5**, 266–277 (1957)
16. Dantzig, G.B.: On the significance of solving linear-programming problems with some integer variables. Econometrica **28**, 30–44 (1960)
17. Dantzig, G.B.: Linear Programming and Extensions. Princeton University Press (1963)
18. de Farias Jr., I.R., Johnson, E.L., Nemhauser, G.L.: Branch-and-cut for combinatorial optimization problems without auxiliary binary variables. Knowl. Eng. Rev. **16**, 25–39 (2001)
19. Ibaraki, T.: Integer programming formulation of combinatorial optimization problems. Discrete Math. **16**, 39–52 (1976)
20. Jeroslow, R.G.: Cutting plane theory: disjunctive methods. Ann. Discrete Math. **1**, 293–330 (1977)
21. Jeroslow, R.G.: Representability in mixed integer programming 1: characterization results. Discrete Appl. Math. **17**, 223–243 (1987)
22. Jeroslow, R.G.: A simplification for some disjunctive formulations. Eur. J. Oper. Res. **36**, 116–121 (1988)
23. Jeroslow, R.G.: Representability of functions. Discrete Appl. Math. **23**, 125–137 (1989)
24. Jeroslow, R.G., Lowe, J.K.: Modeling with integer variables. Math. Program. Study **22**, 167–184 (1984)
25. Jeroslow, R.G., Lowe, J.K.: Experimental results on the new techniques for integer programming formulations. J. Oper. Res. Soc. **36**, 393–403 (1985)
26. Keha, A.B., de Farias, I.R., Nemhauser, G.L.: Models for representing piecewise linear cost functions. Oper. Res. Lett. **32**, 44–48 (2004)

27. Keha, A.B., de Farias, I.R., Nemhauser, G.L.: A branch-and-cut algorithm without binary variables for nonconvex piecewise linear optimization. Oper. Res. **54**, 847–858 (2006)
28. Lee, J.: All-different polytopes. J. Comb. Optim. **6**, 335–352 (2002)
29. Lee, J., Margot, F.: On a binary-encoded ilp coloring formulation. INFORMS J. Comput. **19**, 406–415 (2007)
30. Lee, J., Wilson, D.: Polyhedral methods for piecewise-linear functions I: the lambda method. Discrete Appl. Math. **108**, 269–285 (2001)
31. Lowe, J.K.: Modelling with integer variables. Ph.D. thesis, Georgia Institute of Technology (1984)
32. Magnanti, T.L., Stratila, D.: Separable concave optimization approximately equals piecewise linear optimization. In: D. Bienstock, G.L. Nemhauser (eds.) IPCO, *Lecture Notes in Computer Science*, vol. 3064, pp. 234–243. Springer (2004)
33. Markowitz, H.M., Manne, A.S.: On the solution of discrete programming-problems. Econometrica **25**, 84–110 (1957)
34. Martin, A., Moller, M., Moritz, S.: Mixed integer models for the stationary case of gas network optimization. Math. Program. **105**, 563–582 (2006)
35. Meyer, R.R.: On the existence of optimal solutions to integer and mixed-integer programming problems. Math. Program. **7**, 223–235 (1974)
36. Meyer, R.R.: Integer and mixed-integer programming models - general properties. J. Optim. Theory Appl. **16**, 191–206 (1975)
37. Meyer, R.R.: Mixed integer minimization models for piecewise-linear functions of a single variable. Discrete Math. **16**, 163–171 (1976)
38. Meyer, R.R.: A theoretical and computational comparison of equivalent mixed-integer formulations. Nav. Res. Logist. **28**, 115–131 (1981)
39. Padberg, M.: Approximating separable nonlinear functions via mixed zero-one programs. Oper. Res. Lett. **27**, 1–5 (2000)
40. Sherali, H.D.: On mixed-integer zero-one representations for separable lower-semicontinuous piecewise-linear functions. Oper. Res. Lett. **28**, 155–160 (2001)
41. Sherali, H.D., Shetty, C.M.: Optimization with Disjunctive Constraints, *Lecture Notes in Economics and Mathematical Systems*, vol. 181. Springer-Verlag (1980)
42. Shields, R.: personal communication (2007)
43. Todd, M.J.: Union jack triangulations. In: S. Karamardian (ed.) Fixed Points: algorithms and applications, pp. 315–336. Academic Press (1977)
44. Tomlin, J.A.: A suggested extension of special ordered sets to non-separable non-convex programming problems. In: P. Hansen (ed.) Studies on Graphs and Discrete Programming, *Annals of Discrete Mathematics*, vol. 11, pp. 359–370. North Holland (1981)
45. Vielma, J.P., Ahmed, S., Nemhauser, G.L.: Mixed-integer models for nonseparable piecewise linear optimization: Unifying framework and extensions. Oper. Res. (To Appear) (2009)
46. Vielma, J.P., Keha, A.B., Nemhauser, G.L.: Nonconvex, lower semicontinuous piecewise linear optimization. Discrete Optim. **5**, 467–488 (2008)
47. Vielma, J.P., Nemhauser, G.L.: Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. In: A. Lodi, A. Panconesi, G. Rinaldi (eds.) IPCO, *Lecture Notes in Computer Science*, vol. 5035, pp. 199–213. Springer (2008)
48. Watters, L.J.: Reduction of integer polynomial programming problems to zero-one linear programming problems. Oper. Res. **15**, 1171–1174 (1967)
49. Wilf., H.S.: Combinatorial algorithms–an update, *CBMS-NSF regional conference series in applied mathematics*, vol. 55. Society for Industrial and Applied Mathematics (1989)
50. Wilson, D.: Polyhedral methods for piecewise-linear functions. Ph.D. thesis, University of Kentucky (1998)