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

Branch-and-Cut for Complementarity-Constrained Optimization

I.R. de Farias JR. · E. Kozyreff · M. Zhao

Received: date / Accepted: date
December 10, 2012

Abstract We report and analyze the results of our computational testing of branch-and-cut for the *complementarity-constrained optimization problem* (CCOP). Besides the MIP cuts commonly present in commercial optimization software, we used inequalities that explore complementarity constraints. To do so, we generalized two families of cuts proposed earlier by de Farias, Johnson, and Nemhauser that had never been tested computationally. Our test problems consisted of linear, binary, and general integer programs with complementarity constraints. Our results on the use of complementarity cuts within a major commercial optimization solver show that they are of critical importance to tackling difficult CCOP instances, typically reducing the computational time required to solve them tremendously.

Keywords complementarity · multiple-choice · special ordered set · mixed-integer programming · knapsack set · polyhedral method · branch-and-cut

1 Introduction

Given a set of variables, the constraint that at most one of them is allowed to be nonzero in a feasible solution is one of the most recurrent in the field of mathematical optimization. It is usually called *complementarity* or *multiple-choice*, and the set of variables is called a *special ordered set of type 1* (SOS1 for short. The multiple-choice constraint is also commonly called SOS1). When the variables are binary, complementarity is often called *generalized upper bound* (GUB). Complementarity constraints are present in an enormous number of applications. Two examples are

I.R. de Farias JR. · E. Kozyreff
Texas Tech
E-mail: {ismael.de-farias, ernee.kozyreff}@ttu.edu

M. Zhao
SAS
E-mail: ming.zhao@sas.com

location of stores in shopping malls [3] and nurse scheduling [13]. We refer to the book by Williams [30] and references therein for a list of applications. See also Cottle et al. [4].

Due to its importance, all main commercial optimization solvers contain special facilities for SOS1. Yet, despite major improvements in recent years, typically they are capable of solving only toy instances of the *complementarity-constrained optimization problem* (CCOP) to proven optimality. One feature that is missing is the use of cutting planes that specifically take SOS1 into account (henceforth called SOS1 cuts) in branch-and-cut (B&C). To justify the absence of SOS1 cuts, it is often argued that complementarity can be formulated by using binary variables [7] (note that this is only true when the variables in the SOS1 are bounded [22,23]), in which case mixed-integer programming (MIP) cutting planes, such as Gomory or MIR, can be used to tackle CCOP through B&C.

We investigate the importance of SOS1 cuts for solving CCOP. Specifically, we investigate the efficiency of two families of SOS1 cuts proposed earlier by de Farias et al. [10], which were never tested computationally, and which we generalize here. We show that, contrary to general belief, the use of SOS1 cuts can reduce the time required to solve CCOP tremendously, even when MIP cuts are used. As a matter of fact, we show that SOS1 cuts are also important when CCOP is an integer program (IP, i.e. all variables are integer).

We note that recently Zhao and de Farias [32] and de Farias et al. [11] investigated the use of *special ordered sets of type 2 (SOS2) cuts* on piecewise linear optimization. There, as in here for SOS1 on CCOP, the use of SOS2 cuts enhanced considerably our ability to solve difficult instances of the piecewise linear optimization problem to proven optimality through B&C.

1.1 Notation, Cutting Plane Approach, and Assumptions

Let m be a positive integer and $M = \{1, \ldots, m\}$. For each $i \in M$, let n_i be a positive integer and $N_i = \{1, \ldots, n_i\}$. We denote as ij the ordered pair (i, j) and any set with one element by the element itself. Let $u_{ij} > 0 \; \forall j \in N_i$, $i \in M$. CCOP is:

$$
\begin{aligned}
maximize \quad & \sum_{i \in M} \sum_{j \in N_i} c_{ij} x_{ij} \\
s.t. \quad & Ax \leq \beta && (1) \\
& \{x_{i1}, \ldots, x_{in_i}\} \text{ is SOS1 }, \; i \in M && (2) \\
& x_{ij} \in [0, u_{ij}], \qquad j \in N_i, i \in M. && (3)
\end{aligned}
$$

We denote $d = \sum_{i \in M} n_i$, i.e. d is the number of variables. Also, $I = \cup_{i \in M} (i \times N_i)$, i.e. I is the set of indices of x. For $T \subseteq I$, $M_T = \{i \in M : ij \in T \text{ for some } j \in N_i\}$. Finally, if $\{a_i\}$ is a sequence of real numbers and $R = \emptyset$, $\sum_{i \in R} a_i = 0$.

Our cutting plane approach for tackling CCOP through B&C follows that of Crowder et al. [5]. Specifically, we use as cuts inequalities valid for the convex hull

of the set defined by one of the inequalities of (1), the complementarity constraints (2), and the bound constraints (3). In other words, our polyhedron of interest is the *SOS1 knapsack polytope* $P = conv(S)$, where

$$S = \left\{ x \in \Re^d : x \text{ satisfies } (2), (3), \text{ and } (4) \right\},$$

with

$$\sum_{i \in M} \sum_{j \in N_i} a_{ij} x_{ij} \leq b. \tag{4}$$

Here, (4) is one of the inequalities of (1), which is determined by the separation algorithm (to be discussed in Section 3) of the B&C scheme.

To the best of our knowledge, the inequality description of P was first studied by de Farias et al. [10]. There, among other results, two families of facet-defining inequalities were given. In this paper we generalize these inequalities and use them within B&C to solve difficult instances of CCOP. We evaluate their impact on the performance of B&C.

We also analyzed the impact of formulation choice ("usual" MIP [25], LOG formulation [28,29], or SOS1 approach [2]), to be reported in a follow-up paper. Because the results for the "usual" MIP formulation were far superior to LOG and the SOS1 approach, here we report results for the "usual" MIP formulation only.

We assume that:

Assumption 1. $n_i \geq 2$ for some $i \in M$
Assumption 2. $\sum_{i \in M} max \left\{ a_{i1} u_{i1}, \ldots, a_{in_i} u_{in_i} \right\} > b$
Assumption 3. $b > 0$ and $a_{ij} \geq 0 \; \forall ij \in I$
Assumption 4. x_{ij}, $ij \in I$, is scaled so that $a_{ij} \leq b$ and $u_{ij} = 1$
Assumption 5. $a_{i1} \geq a_{i2} \geq \cdots \geq a_{in_i} \; \forall i \in M$.

If Assumptions 1 and 2 do not hold, the problem is trivial, so there is no loss of generality in them. Assumption 4 can be made without loss of generality once Assumption 3 is made. There is loss of generality in Assumption 3. However, even in the presence of Assumption 3, CCOP can be interesting and difficult [3,13,30]. There is no loss of generality in Assumption 5. Once Assumptions 3, 4, and 5 are made, Assumption 2 can be rewritten as $\sum_{i \in M} a_{i1} > b$.

1.2 Previous Work

As already mentioned, P was first studied by de Farias et al. [10]. Some of the results given there are reviewed in Section 2. We note that the *complementarity knapsack problem* was introduced by Ibaraki et al. [19], see also Ibaraki [17].

To the best of our knowledge, the first cutting plane results on CCOP were given by Ibaraki [18]. There, it is shown how to derive from the tableaux of the linear programming (LP) relaxation (LPR) of CCOP, obtained by dropping (2) from the constraint set of CCOP, an inequality valid for the feasible set of CCOP, which cuts off the basic solution of the tableaux in case it does not satisfy (2).

Another early result on the inequality description of the convex hull of the feasible set of CCOP was given by Jeroslow [20]. There, a simple procedure is given to derive a full inequality description of it.

Recently, Nguyen et al. [26] extended reformulation-linearization [27] to the following special case of the feasible set of CCOP:

$$Ax + By + Cs = b \tag{5}$$
$$\{y_i, s_i\} \text{ is SOS1 } \forall i \tag{6}$$
$$y, s \geq 0. \tag{7}$$

Finally, Mitchel et al. [24] studied the convex hull of the feasible set of CCOP for the special case considered by Nguyen et al. [26] (i.e. with constraints (5), (6), and (7)) with $A = 0$ and $C = I$. Unlike the other aforementioned studies, [24] gives computational results on the use of their inequalities to solve instances of their special case of CCOP.

The remainder of the paper is organized as follows. In Section 2 we review some of the results of [10] and we generalize its two main inequalities. In Section 3 we give separation heuristics for the two inequalities. In Section 4 we specify the platform used in the computational tests, the instances tested, and what was tested. In Sections 5 through 9 we give and analyze the results of our computational experiments. Finally, in Section 10 we give conclusions and directions for further research.

2 Polyhedral Results

We now review the concepts and results introduced in de Farias et al. [10] that will be most relevant here. We also generalize two families of facet-defining inequalities for P given in [10]. The generalized inequalities are the ones used as cuts in our B&C approach to CCOP (besides the underlying optimization solver's built-in MIP cuts).

We start with the most basic facts about P.

Proposition 1 *P is full-dimensional.* □

Proposition 2 *The inequalities $x_{ij} \geq 0$ are facet-defining for P $\forall j \in N_i, i \in M$.* □

Proposition 3 *Inequality (4) is facet-defining for P iff $\sum_{i \in M - i'} a_{i1} + a_{i'n_{i'}} \geq b$ $\forall i' \in M$.* □

Proposition 4 *For $i \in M$,*

$$\sum_{j \in N_i} x_{ij} \leq 1 \tag{8}$$

is valid for P. It defines a facet of P iff $a_{in_i} < b$. □

We now extend the concepts of *cover* and *cover inequality*, originally introduced in the context of 0-1 programming by Balas [1], Hammer et al. [15], and Wolsey [31], to the context of complementarity optimization. The two SOS1 cuts we use in our B&C approach to CCOP are *lifted cover inequalities*.

Definition 1 Let $C = \{i_1 j_1, \ldots, i_k j_k\} \subseteq I$, with $i_1, \ldots, i_k$ all distinct. The set C is called a cover if $\sum_{ij \in C} a_{ij} > b$. Given a cover C, the inequality

$$\sum_{ij \in C} a_{ij} x_{ij} \leq b \tag{9}$$

is called a cover inequality. □

Clearly, (9) is valid not only for P but also for the LPR feasible set of CCOP. Therefore, (9) cannot be used as a cutting plane. To obtain a cut, (9) needs to be *lifted*. The details of *lifting* in the context of complementarity-constrained optimization, which we illustrate below, are given in [10].

Example 1 Let $m = 5$, $n_1 = n_2 = n_3 = n_5 = 2$, $n_4 = 3$, and (4) be given by

$$\begin{aligned}(6x_{11} + x_{12}) + (2x_{21} + x_{22}) + (4x_{31} + 3x_{32}) \\ + (8x_{41} + 6x_{42} + x_{43}) + (9x_{51} + 4x_{52}) \leq 13.\end{aligned} \tag{10}$$

Then, $C = \{11, 21, 32, 42\}$ is a cover and

$$6x_{11} + 2x_{21} + 3x_{32} + 6x_{42} \leq 13 \tag{11}$$

is a cover inequality. Inequality (11) and constraints (2) imply that

$$(6x_{11} + 2x_{12}) + 2x_{21} + 3x_{32} + 6x_{42} \leq 13 \tag{12}$$

is valid for P. Note that (12) is not valid for the feasible set of the LPR of CCOP. Inequality (12) cuts off, for example, $x_{11} = \frac{1}{5}$, $x_{12} = \frac{4}{5}$, $x_{21} = x_{32} = x_{42} = 1$, and $x_{ij} = 0$ otherwise, which is a vertex of LPR $\cap \left\{x \in \Re^d : x \text{ satisfies } (8)\ \forall i \in M\right\}$. Here, (11) was *lifted with respect to* x_{12}. □

By lifting (9), de Farias et al. [10] obtained two families of facet-defining inequalities, which we now give.

Theorem 1 *Let C be a cover, and suppose that $j = 1\ \forall ij \in C$. Additionally, assume that $\exists i' \in M_C$ and $j' \in N_{i'}$ such that*

$$\sum_{i \in M_C - i'} a_{i1} + a_{i'j'} < b. \tag{13}$$

Then,

$$\sum_{i \in M_C} \sum_{j=1}^{n_i} max \left\{ a_{ij}, b - \sum_{k \in M_C - i} a_{k1} \right\} x_{ij} \leq b \tag{14}$$

is valid and facet-defining for P. □

Example 1 (Continued) The set $C = \{41, 51\}$ is a cover that satisfies (13) for $i' = 4$, $j' = 3$ (or $i' = 5$, $j' = 2$). Thus,

$$(8x_{41} + 6x_{42} + 4x_{43}) + (9x_{51} + 5x_{52}) \leq 13 \tag{15}$$

is valid and facet-defining for P.

Inequality (15) cuts off, for example, $x_{41} = 1$, $x_{51} = \frac{1}{5}$, $x_{52} = \frac{4}{5}$, and $x_{ij} = 0$ otherwise, which is a vertex of LPR $\cap \left\{ x \in \Re^d : x \text{ satisfies } (8)\ \forall i \in M \right\}$. □

We now give the second family of facet-defining inequalities derived in [10].

Theorem 2 *Let C be a cover and $i'j' \in C$ with $j' < n_{i'}$. Suppose that $j = n_i$ $\forall ij \in C - i'j'$. Finally, suppose that $\sum_{i \in M_C} a_{in_i} < b$. Then,*

$$\sum_{j \in N_{i'}} max \left\{ a_{i'j}, b - \sum_{k \in M_C - i'} a_{kn_k} \right\} x_{i'j} +$$

$$\sum_{i \in M_C - i'} \sum_{j=1}^{n_i} a_{in_i} max \left\{ 1, \frac{a_{ij}}{b - \sum_{k \in M_C - i} a_{kn_k}} \right\} x_{ij} \leq b \tag{16}$$

is valid and facet-defining for P. □

Example 1 (Continued) The set $C = \{22, 32, 43, 51\}$ is a cover that satisfies the assumptions of Theorem 2 with $i'j' = 51$. Thus,

$$(x_{21} + x_{22}) + (3x_{31} + 3x_{32}) + \left(\frac{8}{5} x_{41} + \frac{6}{5} x_{42} + x_{43} \right) + (9x_{51} + 8x_{52}) \leq 13 \tag{17}$$

is valid and facet-defining for P. Inequality (17) cuts off, for example, $x_{22} = x_{32} = x_{43} = 1$, $x_{51} = \frac{4}{5}$, $x_{52} = \frac{1}{5}$, and $x_{ij} = 0$ otherwise, which is a vertex of LPR $\cap$ $\left\{ x \in \Re^d : x \text{ satisfies } (8)\ \forall i \in M \right\}$. □

Inequalities (14) and (16) are valid and facet-defining under fairly general conditions, and indeed, in our preliminary tests, their use within B&C to solve difficult instances of CCOP was encouraging. It appeared to us, however, that the assumptions that $j = 1$ $\forall ij \in C$ in Theorem 1 and $j = n_i$ $\forall ij \in C - i'j'$ in Theorem 2 were, at times, restrictive, limiting too much the number of separated inequalities. We then generalized Theorems 1 and 2 to drop the assumptions. The resulting inequalities, which we give next, turned out to be considerably more efficient than (14) and (16).

They are the inequalities we used in the computational tests that we report in the paper.

The validity of the inequalities is a consequence of Theorems 1 and 2 and Propositions 5 and 6.

Proposition 5 *Let $T \subset I$. Suppose that*

$$\sum_{ij \in T} \alpha_{ij} x_{ij} \leq \gamma \tag{18}$$

is valid for $\bar{P} = P \cap \left\{x \in \Re^d : x_{ij} = 0 \; \forall ij \in I - T\right\}$. Then, (18) is valid for P.

Proof Let $x^* \in S$, and $\bar{x} \in \Re^d$ be given by

$$\bar{x}_{ij} = \begin{cases} x^*_{ij}, \text{ if } ij \in T \\ 0, \quad \text{otherwise.} \end{cases}$$

Clearly,

$$\sum_{ij \in T} \alpha_{ij} x^*_{ij} = \sum_{ij \in T} \alpha_{ij} \bar{x}_{ij}. \tag{19}$$

Because of Assumption 3, $\bar{x} \in \bar{P}$, and so it satisfies (18). It then follows from (19) that x^* satisfies (18). Therefore, (18) is valid for P. □

Proposition 6 *Let $\bar{M} \subseteq M$, and $\forall i \in \bar{M}$ $r_i, t_i \in N_i$ with $r_i < t_i$. Suppose that*

$$\sum_{i \in \bar{M}} \sum_{j=r_i}^{t_i} \alpha_{ij} x_{ij} \leq b \tag{20}$$

is valid for P. Let $R = \left\{i \in \bar{M} : \alpha_{ir_i} = a_{ir_i}\right\}$ and $T = \left\{i \in \bar{M} : \alpha_{it_i} = a_{it_i}\right\}$. Then,

$$\sum_{i \in R} \sum_{j=1}^{r_i - 1} a_{ir_i} x_{ij} + \sum_{i \in \bar{M}} \sum_{j=r_i}^{t_i} \alpha_{ij} x_{ij} + \sum_{i \in T} \sum_{j=t_i+1}^{n_i} a_{ij} x_{ij} \leq b \tag{21}$$

is valid for P.

Proof Let $x^* \in S$, and $\bar{x} \in \Re^d$ be given by

$$\bar{x}_{ij} = \begin{cases} x^*_{ij}, & \text{if } i \in \bar{M}, j \in \{r_i, \ldots, t_i\}, \text{ and } x^*_{ij} > 0 \\ x^*_{il}, & \text{if } i \in R, j = r_i, \text{ and } x^*_{il} > 0 \text{ for some } l \in \{1, \ldots, r_i - 1\} \\ \frac{a_{ij}}{a_{it_i}} x^*_{il}, & \text{if } i \in T, j = t_i, \text{ and } x^*_{il} > 0 \text{ for some } l \in \{t_i + 1, \ldots, n_i\} \\ 0, & \text{otherwise.} \end{cases}$$

Clearly,

$$\sum_{i\in R}\sum_{j=1}^{r_i-1} a_{ir_i}x^*_{ij} + \sum_{i\in \bar{M}}\sum_{j=r_i}^{t_i} \alpha_{ij}x^*_{ij} + \sum_{i\in T}\sum_{j=t_i+1}^{n_i} a_{ij}x^*_{ij} = \sum_{i\in \bar{M}}\sum_{j=r_i}^{t_i} \alpha_{ij}\bar{x}_{ij}. \quad (22)$$

It is also easy to see that $\bar{x} \in S$. So, $\bar{x}$ satisfies (20). It then follows from (22) that x^* satisfies (21). Therefore, (21) is valid for P. □

We now give the first family of inequalities we used in our computational tests.

Theorem 3 *Let C be a cover. Denote the elements of C as ir_i ($\forall i \in M_C$). Assume that $\sum_{i\in M_C - i'} a_{ir_i} + a_{i'j'} < b$ for some $i' \in M_C$ and $j' \in \{j \in N_{i'} : r_{i'} < j' \leq n_{i'}\}$. Then,*

$$\sum_{i\in M_C}\sum_{j=1}^{r_i-1} a_{ir_i}x_{ij} + \sum_{i\in M_C}\sum_{j=r_i}^{n_i} \max\left\{a_{ij}, b - \sum_{k\in M_C - i} a_{kr_k}\right\} x_{ij} \leq b \quad (23)$$

is valid for P. Inequality (23) is facet-defining when $r_i = 1\ \forall i \in M_C$.

Proof By applying Theorem 1 we see that

$$\sum_{i\in M_C}\sum_{j=r_i}^{n_i} \max\left\{a_{ij}, b - \sum_{k\in M_C - i} a_{kr_k}\right\} x_{ij} \leq b \quad (24)$$

is valid for $P \cap \left\{x \in \Re^d : x_{ij} = 0 \text{ when } i \in M_C \text{ and } j < r_i\right\}$. By applying Proposition 5, it follows that (24) is valid for P. Finally, we apply Proposition 6 with $\bar{M} = M_C$ and $t_i = n_i\ \forall i \in M_C$ to obtain (23) (note that $R = M_C$ and that T is irrelevant, since $t_i = n_i\ \forall i \in M_C$, and consequently the third term of (21) is not present). That (23) is facet-defining when $r_i = 1\ \forall i \in M_C$ follows directly from Theorem 1. □

Finally, we give the second family of inequalities we used in our computational tests. We omit the proof of Theorem 4, since its idea is similar to that in the the proof of Theorem 3.

Theorem 4 *Let C be a cover and $i'j' \in C$ with $j' < n_{i'}$. Denote the elements of C other than $i'j'$ as it_i ($\forall i \in M_C - i'$). Suppose that $\sum_{i\in M_C - i'} a_{it_i} + a_{i'n_{i'}} < b$. Then,*

$$\sum_{j\in N_{i'}} \max\left\{a_{i'j}, b - \sum_{k\in M_C - i'} a_{kt_k}\right\} x_{i'j} +$$

$$\sum_{i\in M_C - i'}\left(\sum_{j=1}^{t_i} a_{it_i} \max\left\{1, \frac{a_{ij}}{b - \sum_{k\in M_C - \{i,i'\}} a_{kt_k} - a_{i'n_{i'}}}\right\} x_{ij} + \sum_{j=t_i+1}^{n_i} a_{ij}x_{ij}\right) \leq b \quad (25)$$

is valid for P. Inequality (25) is facet-defining when $t_i = n_i\ \forall i \in M_C - i'$. □

3 Separation

We do not know of polynomial-time separation algorithms for inequalities (23) and (25). As a matter of fact, because 0-1 cover separation is NP-Complete (see [25]) and (23) and (25) are lifted cover inequalities, we conjecture that their separation is NP-Complete. We then used simple separation heuristics for them, which we now describe.

Let x^* be the optimal solution to the LPR of the current branch-and-bound (B&B) node. Suppose that x^* does not satisfy (2) for some SOS1. For each knapsack constraint within $Ax \leq \beta$ (i.e. each row of A), we attempt to find an inequality (23) and an inequality (25). To simplify the presentation, assume that the entries of the row of A being considered satisfy Assumption 5. We also omit the index of the row.

Let $M_1 = \{i \in M : a_{ij}x_{ij} > 0 \text{ for some } j \in N_i\}$ and $M_2 = \{i \in M : x_{ij} > 0 \text{ for some } j \in N_i\}$. Suppose that $M_1 \neq \emptyset$ and $M_2 \neq \emptyset$. For each $i \in M_1$ let

$$r_i = \min\{j \in N_i : a_{ij}x_{ij} > 0\},$$

and for each $i \in M_2$ let

$$t_i = \max\{j \in N_i : x_{ij} > 0\}.$$

To separate (23), we form the set $C = \{ir_i : i \in M_1\}$. If $\sum_{ij \in C} a_{ij} > b$, C is a cover. We then search, in increasing order of $i \in M_C$ and $j \in \{r_i, \ldots, n_i\}$, a pair $i'j'$ satisfying $\sum_{i \in M_C - i'} a_{ir_i} + a_{i'j'} < b$. If none is found, we quit. Otherwise, for the first pair found we form an inequality (23). If x^* violates the resulting inequality (23) by at least 0.01 (the difference between the left-hand-side and the right-hand-side of (23)), we add the inequality to the cut pool. Otherwise, we quit. We separate at most 1,000 inequalities (23). The minimum violation and maximum number of cuts were determined after preliminary tests, where these values gave the best computational results.

To separate (25), we first test whether $\sum_{i \in M_2} a_{it_i} > b$. If no, we quit. If yes, let $s \in M$ be such that

$$\sum_{\substack{i \in M_2 \\ i < s}} a_{it_i} < b \text{ and } \sum_{\substack{i \in M_2 \\ i \leq s}} a_{it_i} \geq b.$$

We search, in increasing order of $i \in \{k \in M_2 : k < s\}$ and $j \in \{1, \ldots, t_i\}$, a pair $i'j'$ satisfying

$$\sum_{\substack{i \in M_2 - i' \\ i < s}} a_{it_i} + a_{i'j'} > b.$$

If none is found, we quit. Otherwise, for the first pair found we form the set $C = \{it_i : i \in M_2 - i', i < s\} \cup i'j'$, which is a cover. We then apply Theorem 4 to derive an inequality (25). As before, and for the same reason, if x^* violates the resulting inequality (23) by at least 0.01, we add the inequality to the cut pool. Otherwise, we quit. We separate at most 1,000 inequalities (25).

4 Platform, Problems and Instances, and Tests Conducted

We performed our computational tests in the Texas Tech High Performance Computing Center's Intel Xeon E5450 3.0GHz CPU with 16GB RAM nodes (two CPUs on a single board for each node) [16]. We used the callable libraries of GUROBI 5.0.1, which we ran on a single thread. The reason for single thread is that results on multiple threads are not reproducible, as the number of enumeration nodes may vary in this case. We allowed a maximum computational time of 3,600 seconds of CPU for each instance.

4.1 Problems and Instances

We tested three types of CCOP, complementarity-constrained: LP, binary IP (or BIP), and IP with general integer variables (or GIP). We generated 1,800 LP, 1,800 BIP, and $5\times$1,800 = 9,000 GIP (total of 12,600) instances of CCOP (the 5 different groups of GIP instances are explained below). In all LP instances $x_{ij} \in [0,1]\ \forall j \in N_i, i \in M$; and for a given GIP instance, $x_{ij} \in \{0,\ldots,u\}\ \forall j \in N_i, i \in M$, where u is a positive integer. In other words, the upper bound to all integer variables of a GIP is the same. When the upper bound to the variables of a GIP instance is u, we refer to it as a u-IP. Finally, for a given instance, $n_i = n\ \forall i \in M$, where n is a positive integer. In other words, the number of elements of every SOS1 of an instance is the same. We denote $N = \{1,\ldots,n\}$.

We denote as ℓ the number of rows of A. For the instances generated, $\ell \in \{120, 160, 200\}$, $m \in \{60, 80, 100, 120\}$, $n \in \{5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$, and, for the GIP instances, $u \in \{2, 10, 100, 1{,}000, 10{,}000\}$ (the 5 different values of u give the 5 different GIP instance groups; we generated 1,800 u-IP instances for each u). In our initial computational tests we also considered $n = 2$. But these instances proved too easy (almost all solved at the root node with or without SOS1 cuts), and so we discarded them. We call the SOS1s with $n = 5$ "small" and the ones with $n \geq 10$ "large".

In our initial computational tests, the instances with A matrix density 5% or above proved too difficult, almost all of them unsolved after 3,600 seconds of CPU time, with or without SOS1 cuts. For each row of A, let Δ be the expected percentage of SOS1s with a positive entry value of A in at least one element, and, for these SOS1s, let δ be the expected percentage of elements with positive entry values of A (so the density of A is $\Delta \times \delta$). We took $\Delta = 3\%$ and $\delta \in \{50\%, 75\%, 100\%\}$. In our initial computational tests we also considered δ = 25%. But these instances proved too easy (almost all solved at the root node with or without SOS1 cuts), and so we discarded them. For each problem type, i.e. continuous, BIP, and each u-IP, we generated 5 instances for each quadruple (δ, ℓ, m, n).

The instances were generated as follows. Let $v_{ij} \sim \text{Unif}\{1,\ldots,4\}$, $i \in M$, $j \in N$. The objective function coefficients were generated as:

$$c_{ij} = \begin{cases} v_{in}, & \text{if } j = n \\ c_{i(j+1)} + v_{ij}, & \text{otherwise.} \end{cases}$$

We now describe how the entries of each row of A were generated. Let $D_i \sim \text{Unif}[0,1]$, $i \in M$, and $d_{ij} \sim \text{Unif}[0,1]$, $i \in M$, $j \in N$. Let $w_{in} \sim \text{Unif}\{1,\ldots,3\}$, $i \in M$, and $w_{ij} \sim \text{Unif}\{1,\ldots,5\}$, $i \in M$, $j \in N-n$. Finally, let

$$z_{ij} = \begin{cases} w_{ij}, & \text{if } j = n \\ w_{ij} + \left\lceil \dfrac{c_{ij} a_{i(j+1)}}{c_{i(j+1)}} \right\rceil, & \text{otherwise.} \end{cases}$$

Then,

$$a_{ij} = \begin{cases} 0, & \text{if } D_i \geq \Delta \text{ or } d_{ij} \geq \delta \\ z_{ij}, & \text{otherwise,} \end{cases}$$

and

$$b = \max\left\{1 + \max\{a_{ij} : i \in M, j \in N\}, \left\lfloor 1.3 \cdot \sum_{i \in M} a_{in} \right\rfloor\right\}.$$

4.2 Tests Conducted

We evaluated GUROBI's performance, i.e. required computational time and number of enumeration nodes, when solving CCOP to proven optimality. We compared the performance that results when:

- SOS1 cuts are used or not
- the "usual" MIP model, LOG model, or SOS1 approach is used to enforce (2).

The results on formulation comparison will be the topic of a follow-up paper. Because the "usual" MIP model was the most efficient in the vast majority of cases, it is the formulation we report our results on.

In [11] we observed that, many times, the best results were obtained with the following cutting plane strategy, which we called B&B + SOS2 cuts: add the SOS2 cuts to GUROBI with all its MIP facilities off (i.e. with MIP default cutting planes, preprocessing, and primal heuristics off). Here we make a similar assessment, i.e. we do not just compare GUROBI in default setting (which we call Default) against Default + SOS1 cuts; we also compare Default against B&B + SOS1 cuts.

For the remainder of the paper, a

- *strategy* will refer to one of the following cutting plane strategies: GUROBI with all its MIP facilities off (which we call B&B), and so no cuts; B&B + SOS1 cuts, and so SOS1 cuts only; Default, and so GUROBI MIP cuts only; and Default + SOS1 cuts, and so GUROBI MIP cuts and SOS1 cuts
- *strategy is more efficient than another* for a CCOP instance when the computational time required to solve the instance is smaller for that strategy
- CCOP instance is *solved* when it is solved to proven optimality.

5 Computational Results

We now give and analyze our computational results. We break them by problem type, i.e. continuous, BIP, and u-IP CCOP. As we will see, within a given problem type, the instances with large SOS1s ($n \geq 10$) are typically harder than the ones with small SOS1s ($n = 5$). Because of this, we report separately results on instances with small and with large SOS1s. We now give a number of results and make a few observations that hold for all problem types.

We tested adding (23) and (25) separately and simultaneously. Individually, (23) performed better than (25) in terms of computational time in most cases. However, adding them simultaneously was consistently better than individually. So we only report results with them added simultaneously.

When separating inequalities implied by knapsack constraints, such as 0-1 cover, one typically chooses knapsacks that are satisfied at equality by the LPR optimal solution, see for example Gu et al. [14]. Surprisingly, for our SOS1 cuts, separating inequalities implied by all knapsacks, satisfied at equality by the LPR optimal solution or not, turned out to be the best option. So, for the results reported, the cuts were obtained this way. As a matter of fact, of the SOS1 inequalities added, about 40% were implied by knapsacks that were not active in the LPR optimal solution.

It is widely agreed that the most important measure of efficiency of cutting planes is reduction of computational time, followed by reduction of number of enumeration nodes. However, it is also interesting to observe how much the integrality gap (the difference between the LPR and the CCOP optimal values divided by the LPR optimal value) is reduced at the root node, i.e. before branching effects take place. For most of our instances, however, the integrality gaps were extremely small, of the order of 0.1%. This is not uncommon for problems with SOS1 constraints, see for example [8,9,12,21]. For this reason, we do not report the optimal values or the integrality gaps of the instances.

Although we discuss other important results, such as number of enumeration nodes and cutting planes separated, to avoid an excessive number of tables, we only present tables that give averages of computational times and number of instances solved. We also present graphs on computational times.

6 CCOP with Continuous Variables

6.1 Number of Instances Solved

In Table 1 we give the number of instances solved for different strategies and different problem sizes. The table is color-coded to reflect the number of instances solved: the entry is darker if the number of solved instances is smaller. The columns list the instances by strategy, and for each strategy by n. The rows list the instances by δ, for each δ by ℓ, and for each pair $\delta.\ell$ by m.

B&B was able to solve only 235 instances, or 13% of the total number of instances. B&B + SOS1 cuts solved 607, or 34%. This was slightly better than Default, which solved 601, or 33%. The best strategy in terms of number of instances solved

was Default + SOS1 cuts, which completed enumeration for 694, or 39% of the instances. Such small percentages show the difficulty of solving CCOP exactly.

As shown in Table 1, CCOP becomes harder as δ increases. For B&B, 189, or 32% of the instances with δ = 50% were solved; 30, or 5% of the instances with δ = 75% were solved; 16, or 3% of the instances with δ = 100% were solved. The reduction in number of instances solved with increasing δ is bigger for large than for small SOS1s. For $n = 5$, 50% of the instances with δ = 50%, 37% of the instances with δ = 75%, and 27% of the instances with δ = 100% were solved; on the other hand, for $n \geq 10$, 29% of the instances with δ = 50%, 1.4% of the instances with δ = 75%, and 0% of the instances with δ = 100% were solved.

For B&B + SOS1 cuts, 419, or 60% of the instances with δ = 50% were solved; 136, or 23% of the instances with δ = 75% were solved; 52, or 9% of the instances with δ = 100% were solved. In particular, for $n = 5$, 93% of the instances with δ = 50%, 70% of the instances with δ = 75%, and 58% of the instances with δ = 100% were solved; for $n \geq 10$, 67% of the instances with δ = 50%, 17% of the instances with δ = 75%, and 3% of the instances with δ = 100% were solved.

For Default, 471, or 79% of the instances with δ = 50% were solved; 104, or 17% of the instances with δ = 75% were solved; 26, or 4% of the instances with δ = 100% were solved. In particular, for $n = 5$, 90% of the instances with δ = 50%, 63% of the instances with δ = 75%, and 37% of the instances with δ = 100% were solved; for $n \geq 10$, 77% of the instances with δ = 50%, 12% of the instances with δ = 75%, and 0.7% of the instances with δ = 100% were solved.

Finally, for Default + SOS1 cuts, 487, or 81% of the instances with δ = 50% were solved; 152, or 25% of the instances with δ = 75% were solved; 55, or 9% of the instances with δ = 100% were solved. In particular, for $n = 5$, 93% of the instances with δ = 50%, 70% of the instances with δ = 75%, and 58% of the instances with δ = 100% were solved; for $n \geq 10$, 72% of the instances with δ = 50%, 20% of the instances with δ = 75%, and 4% of the instances with δ = 100% were solved.

Table 1 also shows that the number of instances solved depends on n (last row of the table). There is a sharp decrease when the size of the SOS1 changes from small ($n = 5$) to large ($n \geq 10$). Specifically, the percentage of instances solved decreases from 38% to a maximum of 14% for B&B; 74% to a maximum of 42% for B&B + SOS1 cuts; 63% to a maximum of 37% for Default; and 74% to a maximum of 45% for Default + SOS1 cuts. This is followed by a slow, not monotone, decrease with increasing n.

6.2 Reduction in Computational Time

We now report and analyze the reduction in computational time due to the use of SOS1 cuts. Specifically, we compare B&B + SOS1 cuts and Default + SOS1 cuts against Default. The percentage time reduction of B&B + SOS1 cuts for an instance is

$$\%\text{Red. B\&B + SOS1 cuts} = \frac{\text{comp. time Default} - \text{comp. time B\&B + SOS1 cuts}}{\text{comp. time Default}},$$

and similarly for the percentage time reduction of Default + SOS1 cuts.

Since the goal of this section is to analyze the reduction in computational time, here, and in all forthcoming sections where we discuss computational time, we only consider instances that were solved by at least one of Default, B&B + SOS1 cuts, or Default + SOS1 cuts. The total number of instances considered here is then 703, of which: 491 with $\delta = 50\%$, 155 with $\delta = 75\%$, and 57 with $\delta = 100\%$. When an instance is not solved by a strategy, we take the computational time of the strategy for that instance to be 3,600 seconds. This means that, in some cases, what we report as reduction is actually a threshold for the reduction (e.g. if the instance is not solved by Default, but it is solved by Default + SOS1 cuts in 1,800 seconds, we give as reduction 50%, which is less than the actual reduction).

Figures 1 and 2 give an overview of computational time reduction. In both figures the individual instances are represented as dots. The horizontal component of a dot is the computational time of Default. In Figure 1 (resp. Figure 2) the vertical component is %Red. B&B + SOS1 cuts (resp. %Red. Default + SOS1 cuts). For instances with a reduction smaller than -100% we give as vertical component -100%, in order to have all instances plotted. Note that it is possible, and not unlikely, that more than one instance occupies the same dot. The percentages next to a table in Figure 1 (resp. 2) are the fraction of instances for which %Red. B&B + SOS1 cuts (resp. %Red. Default + SOS1 cuts) is positive (upper percentage) and negative (lower percentage). The percentage of ties is 100% minus their sum (these are the dots plotted on the horizontal axis).

Figures 1 and 2 show that, for CCOP with continuous variables, either strategy that uses SOS1 cuts is more efficient than Default for the vast majority of instances. For $n = 5$, the fraction of instances for which B&B + SOS1 cuts is more successful than Default is the same as that for Default + SOS1 cuts (although the fraction of instances for which it is less successful is slightly bigger). For $n \geq 10$, however, Default + SOS1 cuts clearly gives a better outcome.

Figure 1 (resp. 2) also indicates that for a significant fraction of instances with positive %Red. B&B + SOS1 cuts (resp. %Red. Default + SOS1 cuts), the reduction is of 50% or more. This is indeed the case. For $n = 5$, both %Red. B&B + SOS1 cuts and %Red. Default + SOS1 cuts are at least 50% in 81% of the cases where they are positive. For $n \geq 10$, %Red. B&B + SOS1 cuts is at least 50% in 84% of the cases, and %Red. Default + SOS1 cuts in 76% of the cases.

It is widely believed that the result of adding cutting planes in a B&C system to solve instances that would otherwise be solved within a few seconds is increase in computational time. However, recent technologies have brought about the need to reduce further the computational time of such instances. This is true, for example, in the case of search engines, see [6]. Figures 1 and 2 indicate that, for CCOP with continuous variables, SOS1 cuts, in many cases, reduce the computational time of these instances. We now give results specifically for instances that Default was able to solve within 10 seconds of CPU time.

For $n = 5$, Default solved 32 instances in 10 seconds or less. B&B + SOS1 cuts was more efficient than Default for 19 of them, was less efficient for 1, and tied for the other 12. On average, B&B + SOS1 cuts reduced Default's computational time by 45%; and the reduction was of 50% or more for 50% of the instances. Default + SOS1

cuts was more efficient than Default for 20 instances, was less efficient for 2, and tied for the other 10. On average, Default + SOS1 cuts reduced Default's computational time by 51%; and the reduction was of 50% or more for 51% of the instances.

For $n \geq 10$, Default solved 88 instances in 10 seconds or less. B&B + SOS1 cuts was more efficient than Default for 51 of them, was less efficient for 8, and tied for the other 29. On average, B&B + SOS1 cuts reduced Default's computational time by 44%; and the reduction was of 50% or more for 49% of the instances. Default + SOS1 cuts was more efficient than Default for 53 instances, was less efficient for 3, and tied for the other 22. On average, Default + SOS1 cuts reduced Default's computational time by 56%; and the reduction was of 50% or more for 57% of the instances. So contrary to current wisdom, SOS1 cuts have the potential to considerably reduce the computational time of instances that state-of-the-art B&C software solves within a few seconds of CPU time.

In Tables 2 and 3 we give the average reductions in time for the instances with small and large SOS1s, respectively. In both tables we list the instances in the same order as the rows of Table 1, i.e. by δ, for each δ by ℓ, and for each pair $\delta.\ell$ by m. So, the tables should be read from top to bottom, and within a row, first the left, then the right entry. The column "Time Def." gives the computational time averages for Default. The columns "%Red. BB+C" and "%Red. D+C" give the percentage reduction of the computational time averages for B&B + SOS1 cuts and Default+ SOS1 cuts, respectively, over the average computational time of Default.

For the instances with small SOS1s, all average reductions for both B&B + SOS1 cuts and Default + SOS1 cuts are positive, except the Default + SOS1 cuts reduction for $\delta.\ell.m$ = 50.160.100, which is 0. With B&B + SOS1 cuts, the average reduction was at least 50% in 56% of the entries of Table 2; while with Default + SOS1 cuts, the average reduction was at least 50% in 69% of the entries. In a face-off, however, B&B + SOS1 cuts emerges slightly better than Default + SOS1 cuts: They tie 4 times, B&B + SOS1 cuts performs better 16 times, and Default + SOS1 cuts wins 12 times. The overall average reduction for B&B + SOS1 cuts was 57%, and the overall average reduction for Default + SOS1 cuts was 54%.

For the instances with large SOS1s, the average reductions for B&B + SOS1 cuts are positive in 70% of the entries of Table 3; while for Default + SOS1 cuts, they are positive in 83% of the entries (note that the number of entries in Table 3 is smaller than in Table 2: 23, as opposed to 32). With B&B + SOS1 cuts, the average reduction was at least 50% in 35% of the entries of Table 3; while with Default + SOS1 cuts, the average reduction was at least 50% in 57% of the entries. In a face-off, Default + SOS1 cuts is the clear winner: Default + SOS1 cuts performs better 20 times, B&B + SOS1 cuts performs better 2 times, and they tie once. The overall average reduction for B&B + SOS1 cuts was -3%, and the overall average reduction for Default + SOS1 cuts was 38%.

6.3 Number of Cutting Planes and Reduction in Number of Enumeration Nodes

As in Section 6.2, here, and in all forthcoming sections where we discuss number of cutting planes and number of enumeration nodes, we only consider instances that

were solved by at least one of Default, B&B + SOS1 cuts, or Default + SOS1 cuts. First, we give the number of cutting planes separated and used. Then, we give the average reduction in number of enumeration nodes.

For both Default and Default + SOS1 cuts, the GUROBI MIP cuts used were Flow cover, MIR, and Gomory. For Default, the average number of Flow cover cuts used was 118 for δ = 50%, 190 for δ = 75%, and 107 for δ = 100%; the average number of MIR cuts used was 208 for δ = 50%, 404 for δ = 75%, and 189 for δ = 100%; the average number of Gomory cuts used was 39 for δ = 50%, 34 for δ = 75%, and 47 for δ = 100%. For Default + SOS1 cuts, the average number of Flow cover cuts used was 103 for δ = 50%, 163 for δ = 75%, and 55 for δ = 100%; the average number of MIR cuts used was 97 for δ = 50%, 137 for δ = 75%, and 63 for δ = 100%; the average number of Gomory cuts used was 26 for δ = 50%, 19 for δ = 75%, and 21 for δ = 100%.

GUROBI filters user cuts, and typically the number of cuts used is considerably smaller than the number of cuts separated. In average, for B&B + SOS1 cuts, 1,678 SOS1 cuts were separated and 446 were used for δ = 50%; 1,980 were separated and 534 were used for δ = 75%; and 1,999 were separated and 282 used for δ = 100%. For Default + SOS1 cuts, in average 1,543 SOS1 cuts were separated and 288 were used for δ = 50%; 1,979 were separated and 377 were used for δ = 75%; and 1,998 were separated and 205 used for δ = 100%.

We now give the average reduction in number of enumeration nodes due to the use of SOS1 cuts. As with computational time, we compare B&B + SOS1 cuts and Default + SOS1 cuts against Default. The percentage enumeration node reduction is calculated similarly to the percentage computational time reduction. For $n = 5$, Default's average number of nodes was 230,157. The average number of enumeration node reductions for B&B + SOS1 and Default + SOS1 were similar, 74% and 68%, respectively. This is in accordance with the similar performance of B&B + SOS1 cuts and Default + SOS1 cuts in terms of computational time. For $n \geq 10$, Default's average number of nodes was 96,371. The average number of enumeration node reduction for B&B + SOS1 was 32%, while for Default + SOS1 cuts was 70%. This is in accordance with the superior performance, in terms of computational time, of Default + SOS1 cuts relative to B&B + SOS1 cuts.

We conclude that the strategies using SOS1 cuts were considerably more efficient than Default (which was considerably more efficient than B&B). Plenty of times B&B + SOS1 cuts was more efficient than Default + SOS1 cuts. But overall, Default + SOS1 cuts was the best strategy for CCOP with continuous variables, both in terms of number of instances solved as well as computational time.

7 CCOP with Binary and 2-IP Variables

Starting now, and for the remainder of the paper, we present and analyze our results on CCOP with integer variables (both binary and general integer). The presentation is organized as: binary and 2-IP, 10-IP and 100-IP, and 1,000-IP and 10,000-IP. The integer variables in a pair are not just the closest by range, but by efficiency of the SOS1 cuts as well. As a matter of fact, the results for 10-IP and 100-IP and for

1,000-IP and 10,000-IP are combined. As we will see, as compared with Default, the strategies using SOS1 cuts are terribly inefficient for BIP, partially efficient for 2-IP (not as efficient as for continuous CCOP), efficient for (10, 100)-IP (closer to continuous CCOP), and absolutely efficient for (1,000, 10,000)-IP (more efficient than for continuous CCOP).

Remember that SOS1 cuts were developed for continuous CCOP. So, the negative results for BIP are not surprising, especially after taking into account the mature state-of-the-art of 0-1 cover and GUB cover implementation. The positive results for GIP, though, came to us as a surprise, especially the ones for 1,000-IP and 10,000-IP. The reason is that cuts developed for LP with combinatorial constraints (such as SOS1 cuts) are seldom efficient for IP, even when they are efficient for LP.

7.1 Binary CCOP

Unlike in the previous and forthcoming sections, here, Default is considerably more efficient than the strategies using SOS1 cuts. First, we discuss number of instances solved.

7.1.1 Number of Instances Solved

Table 4 shows that B&B is once more the strategy that solved the least number of instances, 437, or 24% of the total, followed by B&B + SOS1 cuts, which solved 645, or 36%. Default and Default + SOS1 cuts solved the greatest number of instances: 1,152, or 64% for the first, and 1,147, or 64% for the second. The fact that the number of instances solved by Default and Default + SOS1 cuts were so close, and so much higher than the number of instances solved by B&B + SOS1 cuts is the first sign of the importance of the GUROBI MIP cuts and the minor role played by the SOS1 cuts in this case.

As before, CCOP becomes harder as δ increases. For B&B, 332, or 55% of the instances with δ = 50% were solved; 83, or 14% of the instances with δ = 75% were solved; and 22, or 4% of the instances with δ = 100% were solved. The reduction in number of instances solved with increasing δ is again bigger for those with large SOS1s than for instances with small SOS1s. For $n = 5$, 67% of the instances with δ = 50%, 52% of the instances with δ = 75%, and 35% of the instances with δ = 100% were solved; on the other hand, for $n \geq 10$, 54% of the instances with δ = 50%, 10% of the instances with δ = 75%, and 0.2% of the instances with δ = 100% were solved.

For B&B + SOS1 cuts, 458, or 76% of the instances with δ = 50% were solved; 161, or 27% of the instances with δ = 75% were solved; and 26, or 4% of the instances with δ = 100% were solved. In particular, for $n = 5$, 80% of the instances with δ = 50%, 62% of the instances with δ = 75%, and 38% of the instances with δ = 100% were solved; for $n \geq 10$, 76% of the instances with δ = 50%, 23% of the instances with δ = 75%, and 0.6% of the instances with δ = 100% were solved.

For Default, 600, or 100% of the instances with δ = 50% were solved; 421, or 70% of the instances with δ = 75% were solved; and 131, or 22% of the instances with δ = 100% were solved. In particular, for $n = 5$, 100% of the instances with δ =

50%, 100% of the instances with δ = 75%, and 92% of the instances with δ = 100% were solved; for $n \geq 10$, 100% of the instances with δ = 50%, 67% of the instances with δ = 75%, and 14% of the instances with δ = 100% were solved.

Finally, for Default + SOS1 cuts, 600, or 100% of the instances with δ = 50% were solved; 416, or 69% of the instances with δ = 75% were solved; and 131, or 22% of the instances with δ = 100% were solved. In particular, for $n = 5$, 100% of the instances with δ = 50%, 100% of the instances with δ = 75%, and 90% of the instances with δ = 100% were solved; for $n \geq 10$, 100% of the instances with δ = 50%, 66% of the instances with δ = 75%, and 14% of the instances with δ = 100% were solved.

As before again, the number of instances with small SOS1s solved was higher than the number of instances with large SOS1s. The percentage of instances solved decreases from 51% to a maximum of 24% for B&B; 60% to a maximum of 37% for B&B + SOS1 cuts; 97% to a maximum of 74% for Default; and 97% to a maximum of 74% for Default + SOS1 cuts. Next, we discuss computational time, number of enumeration nodes, and number of cutting planes.

7.1.2 Computational Time, Number of Enumeration Nodes, and Number of Cutting Planes

Here, the total number of instances considered is 1,159, of which: 600 with $\delta = 50\%$, 424 with $\delta = 75\%$, and 135 with $\delta = 100\%$. For the instances solved by at least one of B&B or B&B + SOS1 cuts (645 total), B&B + SOS1 cuts was more efficient: the overall computational time reduction of B&B + SOS1 cuts over B&B for them was of 25%. However, as clearly shown in Figure 1, Default was considerably more efficient than B&B + SOS1 cuts. Overall, the computational time reduction of B&B + SOS1 cuts over Default was of -1,023%. As for number of enumeration nodes, on average B&B enumerated 7,472,191 nodes. B&B + SOS1 cuts reduced over the average number of nodes of B&B by 92%. On the other hand, the reduction over the average number of enumeration nodes for Default was of -1,098%.

Default + SOS1 cuts was considerably more efficient than B&B: the overall computational time reduction over B&B, for the instances solved by at least one of them (1,147 total), was of 94%. However, as shown in Figure 2, again, Default was more efficient than Default + SOS1 cuts. Overall, the computational time reduction of Default + SOS1 cuts over Default was of -20%. As for number of enumeration nodes, Default + SOS1 cuts reduced over the average number of nodes of B&B by 99%. On the other hand, the reduction over the number of enumeration nodes for Default was of -17%.

For 154 instances, Default + SOS1 cuts turned out to be more efficient than Default. That, of course, does not make Default + SOS1 cuts competitive with Default. What might be interesting though is that 9 of them were not solved by Default but were solved by Default + SOS1 cuts. So there may be situations when the use of SOS1 cuts for solving BIP CCOP is advantageous.

Clearly, the reason Default for BIP is so much more efficient than the strategies using SOS1 cuts is in the MIP cuts used here. For both Default and Default + SOS1 cuts, the MIP cuts used were: clique, cover, flow cover, MIR, Gomory, GUB cover,

0-$\frac{1}{2}$, and Mod k. We conjecture that such cuts as cover, GUB cover, and clique dominated the SOS1 cuts. As we show below, these cuts (plus MIR) were the ones used the most. We note that for BIP there are many more different types of MIP cuts than for continuous CCOP.

For Default, the average number of clique cuts used was 30 for δ = 50%, 86 for δ = 75%, and 42 for δ = 100%; the average number of cover cuts used was 55 for δ = 50%, 111 for δ = 75%, and 45 for δ = 100%; the average number of flow cover cuts used was 0 for δ = 50%, 1 for δ = 75%, and 1 for δ = 100%; the average number of MIR cuts used was 24 for δ = 50%, 98 for δ = 75%, and 126 for δ = 100%; the average number of Gomory cuts used was 3 for δ = 50%, 4 for δ = 75%, and 21 for δ = 100%; the average number of GUB cuts used was 15 for δ = 50%, 74 for δ = 75%, and 87 for δ = 100%; the average number of 0-$\frac{1}{2}$ cuts used was 2 for δ = 50%, 13 for δ = 75%, and 20 for δ = 100%; and the average number of Mod k cuts used was 0 for δ = 50%, 0 for δ = 75%, and 2 for δ = 100%.

For Default + SOS1 cuts, the average number of MIP cuts used was remarkably similar to the number of cuts for Default. The average number of clique cuts used was 30 for δ = 50%, 89 for δ = 75%, and 42 for δ = 100%; the average number of cover cuts used was 49 for δ = 50%, 97 for δ = 75%, and 33 for δ = 100%; the average number of flow cover cuts used was 0 for δ = 50%, 1 for δ = 75%, and 1 for δ = 100%; the average number of MIR cuts used was 24 for δ = 50%, 96 for δ = 75%, and 123 for δ = 100%; the average number of Gomory cuts used was 4 for δ = 50%, 3 for δ = 75%, and 18 for δ = 100%; the average number of GUB cuts used was 16 for δ = 50%, 75 for δ = 75%, and 85 for δ = 100%; the average number of 0-$\frac{1}{2}$ cuts used was 2 for δ = 50%, 11 for δ = 75%, and 17 for δ = 100%; and the average number of Mod k cuts used was 0 for δ = 50%, 0 for δ = 75%, and 2 for δ = 100%.

As for SOS1 cuts, on average, for B&B + SOS1 cuts, 1,589 cuts were separated and 561 were used for δ = 50%; 1,995 were separated and 820 were used for δ = 75%; and 2,000 were separated and 454 used for δ = 100%. For Default + SOS1 cuts, 218 cuts were separated and 21 were used for δ = 50%; 1,444 were separated and 54 were used for δ = 75%; and 991 were separated and 40 used for δ = 100%.

7.2 CCOP with 2-IP variables

7.2.1 Number of Instances Solved

As in the previous cases, from Table 5, B&B was the strategy that solved the least number of instances, 186 or 10% of the total, of which: 180, or 30% with δ = 50%; 6, or 1% with δ = 75%; and 0, or 0% with δ = 100%. In particular, for $n = 5$, 37% of the instances with δ = 50%, 10% of the instances with δ = 75%, and 0% of the instances with δ = 100% were solved; for $n \geq 10$, 29% of the instances with δ = 50%, 0% of the instances with δ = 75%, and 0% of the instances with δ = 100% were solved.

For B&B + SOS1 cuts, 414 or 23% of the total number of instances was solved, of which: 367, or 61% of the instances with δ = 50%; 45, or 8% of the instances with δ = 75%; and 2, or 0.3% of the instances with δ = 100%. In particular, for $n = 5$, 50% of the instances with δ = 50%, 12% of the instances with δ = 75%, and 3% of

the instances with δ = 100% were solved; for $n \geq 10$, 62% of the instances with δ = 50%, 7% of the instances with δ = 75%, and 0% of the instances with δ = 100% were solved.

For Default, 616 or 34% of the total number of instances was solved, of which: 509, or 85% of the instances with δ = 50%; 90, or 15% of the instances with δ = 75%; and 17, or 3% of the instances with δ = 100%. In particular, for $n = 5$, 83% of the instances with δ = 50%, 42% of the instances with δ = 75%, and 27% of the instances with δ = 100% were solved; for $n \geq 10$, 85% of the instances with δ = 50%, 12% of the instances with δ = 75%, and 0.2% of the instances with δ = 100% were solved.

Finally, Default + SOS1 cuts was the strategy that solved the greatest number of instances, 624 or 35% of the total number of instances was solved, of which: 516, or 86% of the instances with δ = 50%; 91, or 17% of the instances with δ = 75%; and 17, or 3% of the instances with δ = 100%. In particular, for $n = 5$, 83% of the instances with δ = 50%, 40% of the instances with δ = 75%, and 27% of the instances with δ = 100% were solved; for $n \geq 10$, 86% of the instances with δ = 50%, 12% of the instances with δ = 75%, and 0.2% of the instances with δ = 100% were solved.

Regarding the dive in number of instances solved from small to large SOS1s, the percentage of instances solved decreased from 16% to a maximum of 11% for B&B; 51% to a maximum of 34% for Default; and 50% to a maximum of 34% for Default + SOS1 cuts. For B&B with SOS1 cuts, the percentage remained flat, around 22%.

7.2.2 Computational Time, Number of Enumeration Nodes, and Number of Cutting Planes

Here, the total number of instances considered is 635, of which: 519 with $\delta = 50\%$, 98 with $\delta = 75\%$, and 18 with $\delta = 100\%$. From Figure 1, it appears that, just as with BIP, Default was more efficient than B&B + SOS1 cuts for CCOP with 2-IP. Indeed, B&B + SOS1 cuts reduced over the average computational time of Default by -210% for $n = 5$ and by -161% for $n \geq 10$.

On the other hand, Figure 2 shows that Default + SOS1 cuts was more efficient than Default for the majority of instances. The figure also indicates that for $n \geq 10$ a significant fraction of the positive reductions by Default + SOS1 cuts over Default was of 50% or more. That was indeed the case: Default + SOS1 cuts reduced over the computational time of Default by at least 50% in 63% of the instances for which %Red. Default + SOS1 cuts was positive.

Figure 2 indicates that, for $n \geq 10$, Default + SOS1 cuts was more efficient than Default for the instances solved by Default within 10 seconds a considerable number of times. Indeed, for $n \geq 10$, Default solved 130 instances in 10 seconds or less. For these instances, Default + SOS1 cuts was more efficient than Default for 69 of them, was less efficient for 12, and tied for the other 49. On average, for these instances, Default + SOS1 cuts reduced Default's computational time by 33%, and the reduction, for those with positive %Red. Default + SOS1, was of 50% or more for 34% of them.

In Tables 6 and 7 we give the average reductions in time for the instances with small and large SOS1s, respectively. For both tables, the reductions for B&B + SOS1 cuts were negative in almost all entries, the overall reduction being -210% for $n = 5$

and -161% for $n \geq 10$. On the other hand, the reductions for Default + SOS1 cuts were positive in almost all entries, the overall reduction being 14% for $n = 5$ and 0.2% for $n \geq 10$.

As for number of enumeration nodes, on average Default enumerated 60,301 nodes. B&B + SOS1 cuts reduced over the average number of nodes of Default by 25%; and Default + SOS1 cuts reduced over the average number of nodes of Default by 70%.

Regarding cutting planes, both Default and Default + SOS1 cuts used the same MIP cuts as BIP: clique, cover, flow cover, MIR, Gomory, GUB cover, 0-$\frac{1}{2}$, and Mod k; and, on the top of them, implied bound. Here, however, by far, the most used cut in either case was MIR. In particular, the number of cover and GUB cover cuts was quite small.

For Default, the average number of clique cuts used was 3 for δ = 50%, 0 for δ = 75%, and 0 for δ = 100%; the average number of cover cuts used was 6 for δ = 50%, 0 for δ = 75%, and 0 for δ = 100%; the average number of flow cover cuts used was 48 for δ = 50%, 48 for δ = 75%, and 32 for δ = 100%; the average number of MIR cuts used was 214 for δ = 50%, 907 for δ = 75%, and 403 for δ = 100%; the average number of Gomory cuts used was 38 for δ = 50%, 33 for δ = 75%, and 54 for δ = 100%; the average number of GUB cuts used was 1 for δ = 50%, 0 for δ = 75%, and 0 for δ = 100%; the average number of 0-$\frac{1}{2}$ cuts used was 46 for δ = 50%, 50 for δ = 75%, and 9 for δ = 100%; the average number of Mod k cuts used was 3 for δ = 50%, 2 for δ = 75%, and 4 for δ = 100%; and the average number of implied bound cuts used was 15 for δ = 50%, 1 for δ = 75%, and 2 for δ = 100%.

For Default + SOS1 cuts, the average number of clique cuts used was 2 for δ = 50%, 0 for δ = 75%, and 0 for δ = 100%; the average number of cover cuts used was 4 for δ = 50%, 0 for δ = 75%, and 0 for δ = 100%; the average number of flow cover cuts used was 47 for δ = 50%, 41 for δ = 75%, and 16 for δ = 100%; the average number of MIR cuts used was 186 for δ = 50%, 713 for δ = 75%, and 387 for δ = 100%; the average number of Gomory cuts used was 28 for δ = 50%, 21 for δ = 75%, and 38 for δ = 100%; the average number of 0-$\frac{1}{2}$ cuts used was 35 for δ = 50%, 32 for δ = 75%, and 5 for δ = 100%; the average number of Mod k cuts used was 3 for δ = 50%, 1 for δ = 75%, and 2 for δ = 100%; and the average number of implied bound cuts used was 3 for δ = 50%, 1 for δ = 75%, and 2 for δ = 100%. No GUB cuts were used here.

As for SOS1 cuts, on average, for B&B + SOS1 cuts, 1,672 cuts were separated and 555 were used for δ = 50%; 1,999 were separated and 638 were used for δ = 75%; and 2,000 were separated and 433 used for δ = 100%. For Default + SOS1 cuts, 1,407 cuts were separated and 166 were used for δ = 50%; 1,986 were separated and 128 were used for δ = 75%; and 2,000 were separated and 158 used for δ = 100%.

We conclude that, for CCOP with binary variables, Default was clearly the best strategy. B&B + SOS1 cuts and Default + SOS1 cuts were surely better than B&B. Default + SOS1 cuts was better than B&B + SOS1 cuts: it solved more instances and it was more efficient. In terms of number of instances solved, Default was only slightly better than Default + SOS1 cuts. However, as far as computational time goes, Default was notably more efficient than Default + SOS1 cuts. We believe that the

MIP cuts, especially cover and GUB cover played a major role here, as opposed to the insignificant role played by the SOS1 cuts.

For CCOP with 2-IP variables, Default + SOS1 cuts was the best strategy. It was slightly better than Default in terms of number of instances solved; in terms of computational time, it was more efficient than Default. Default, on the other hand, solved more instances than B&B + SOS1 cuts and was more efficient than it.

8 CCOP with 10-IP and 100-IP Variables

In order to contain the report, here and in Section 9, we combine GIPs with different integer variable ranges. In this section we combine the results of CCOP with 10-IP and 100-IP. By examining Figures 1 and 2, we see that the strategies using SOS1 cuts perform slightly better for 100-IP than for 10-IP. The difference, however, seems to be small enough that combining the two classes of instances does not appear to be unreasonable.

8.1 Number of Instances Solved

Table 8 gives the number of instances solved for each strategy (now the maximum value for an entry in the table is 10). B&B solved 416 or 12% of the total, of which: 372, or 31% with δ = 50%; 34, or 3% with δ = 75%; and 10, or 0.8% with δ = 100%. In particular, for $n = 5$, 44% of the instances with δ = 50%, 20% of the instances with δ = 75%, and 8% of the instances with δ = 100% were solved; for $n \geq 10$, 30% of the instances with δ = 50%, 1% of the instances with δ = 75%, and 0% of the instances with δ = 100% were solved.

For B&B + SOS1 cuts, 1,067 or 30% of the total number of instances were solved, of which: 828, or 69% of the instances with δ = 50%; 205, or 17% of the instances with δ = 75%; and 34, or 3% of the instances with δ = 100%. In particular, for $n = 5$, 73% of the instances with δ = 50%, 42% of the instances with δ = 75%, and 25% of the instances with δ = 100% were solved; for $n \geq 10$, 69% of the instances with δ = 50%, 14% of the instances with δ = 75%, and 0.3% of the instances with δ = 100% were solved.

For Default, 1,036 or 29% of the total number of instances were solved, of which: 880, or 73% of the instances with δ = 50%; 137, or 11% of the instances with δ = 75%; and 19, or 2% of the instances with δ = 100%. In particular, for $n = 5$, 77% of the instances with δ = 50%, 43% of the instances with δ = 75%, and 15% of the instances with δ = 100% were solved; for $n \geq 10$, 73% of the instances with δ = 50%, 8% of the instances with δ = 75%, and 0.1% of the instances with δ = 100% were solved.

Finally, for Default + SOS1 cuts, 1,115 or 31% of the total number of instances were solved, of which: 891, or 74% of the instances with δ = 50%; 192, or 16% of the instances with δ = 75%; and 32, or 3% of the instances with δ = 100%. In particular, for $n = 5$, 82% of the instances with δ = 50%, 46% of the instances with δ = 75%, and 24% of the instances with δ = 100% were solved; for $n \geq 10$, 73% of the instances

with δ = 50%, 13% of the instances with δ = 75%, and 0.3% of the instances with δ = 100% were solved.

As for the reduction in the percentage number of instances solved from small to large SOS1s, the percentage decreased from 24% to a maximum of 12% for B&B; 46% to a maximum of 30% for B&B + SOS1 cuts; 45% to a maximum of 29% for Default; and 51% to a maximum of 30% for Default + SOS1 cuts.

So, the strategy that solved the greatest number of instances was Default + SOS1 cuts, followed by B&B + SOS1 cuts, then by Default, and finally B&B. Here, as in the previous cases, CCOP becomes harder with increasing δ, and the decrease in the number of instances solved with δ is bigger for large than for small SOS1s. Also, the number of instances solved decreases when the SOS1s are large. Also, the number of instances solved decreased as we changed from small to large SOS1s.

8.2 Reduction in Computational Time

Here, the total number of instances considered is 1,152, of which: 903 with $\delta = 50\%$, 214 with $\delta = 75\%$, and 35 with $\delta = 100\%$. Figures 1 and 2 give the fraction of instances for which B&B + SOS1 cuts and Default + SOS1 cuts, respectively, were more or less efficient than Default, for 10-IP and 100-IP separately. Except for 10-IP and $n = 5$, the strategies using SOS1 cuts were more efficient than Default for considerably many more instances. Note that, the number of times Default + SOS1 cuts was more efficient than Default was greater than the number of times B&B + SOS1 cuts was more successful than Default.

For $n = 5$, B&B + SOS1 cuts reduced over the computational time of Default by at least 50% in 48% of the cases where %Red. B&B + SOS1 cuts was positive; Default + SOS1 cuts reduced over the computational time of Default by at least 50% in 57% of the cases where %Red. Default + SOS1 cuts was positive. For $n \geq 10$, B&B + SOS1 cuts reduced over the computational time of Default by at least 50% in 86% of the cases where %Red. B&B + SOS1 cuts was positive; and Default + SOS1 cuts in 83% of the cases where %Red. Default + SOS1 cuts was positive.

As in continuous CCOP, the use of SOS1 cuts here improved the computational time of the instances solved by Default within a few seconds of CPU time. For $n = 5$, Default solved 48 instances in 10 seconds or less. B&B + SOS1 cuts was more efficient than Default for 17 of them, was less efficient for 12, and tied for the other 19. On average, B&B + SOS1 cuts reduced Default's computational time by -78%; and the reduction was of 50% or more for 6% of the instances. Default + SOS1 cuts was more efficient than Default for 18 instances, was less efficient for 10, and tied for the other 20. On average, Default + SOS1 cuts reduced Default's computational time by 10%; and the reduction was of 50% or more for 23% of the instances.

For $n \geq 10$, Default solved 158 instances in 10 seconds or less. B&B + SOS1 cuts was more efficient than Default for 116 of them, was less efficient for 3, and tied for the other 39. On average, B&B + SOS1 cuts reduced Default's computational time by 56%; and the reduction was of 50% or more for 64% of the instances. Default + SOS1 cuts was more efficient than Default for 129 instances, was less efficient for 0, and tied for the other 29. On average, Default + SOS1 cuts reduced Default's

computational time by 51%; and the reduction was of 50% or more for 54% of the instances.

In Tables 9 and 10 we give the average reductions in time for the instances with small and large SOS1s, respectively. For the instances with small SOS1s, 50% of the average reductions with B&B + SOS1 cuts are positive; all average reductions with Default + SOS1 cuts are positive, except one. With B&B + SOS1 cuts, the average reduction was at least 50% in 23% of the entries of Table 9; while with Default + SOS1 cuts, the average reduction was at least 50% in 47% of the entries. In a face-off, they tie in one entry, B&B + SOS1 cuts performs better 3 times, and Default + SOS1 cuts wins 26 times. The overall average reduction for B&B + SOS1 cuts was 17%; and the overall average reduction for Default + SOS1 cuts was 50%.

For the instances with large SOS1s, the average reductions for B&B + SOS1 cuts are positive in 78% of the entries of Table 10; while for Default + SOS1 cuts, they are positive in 83% of the entries (note that the number of entries in Table 10 is smaller than in Table 9: 18, as opposed to 30). With B&B + SOS1 cuts, the average reduction was at least 50% in 67% of the entries of Table 10; with Default + SOS1 cuts, the average reduction was at least 50% in 44% of the entries. In a face-off, B&B + SOS1 cuts wins over Default + SOS1 cuts this time: B&B + SOS1 cuts performs better 11 times, while Default + SOS1 cuts performs better 7 times. The overall average reduction for B&B + SOS1 cuts was 25%; and the overall average reduction for Default + SOS1 cuts was 33%.

8.3 Number of Cutting Planes and Reduction in Number of Enumeration Nodes

For both Default and Default + SOS1 cuts, the GUROBI MIP cuts used were Flow cover, MIR, Gomory, 0-$\frac{1}{2}$, and Mod k. For Default, the average number of Flow cover cuts used was 95 for δ = 50%, 87 for δ = 75%, and 62 for δ = 100%; the average number of MIR cuts used was 452 for δ = 50%, 1,077 for δ = 75%, and 632 for δ = 100%; the average number of Gomory cuts used was 42 for δ = 50%, 25 for δ = 75%, and 51 for δ = 100%; the average number of 0-$\frac{1}{2}$ cuts used was 4 for δ = 50%, 2 for δ = 75%, and 1 for δ = 100%; and the average number of Mod k cuts used was 2 for δ = 50%, 1 for δ = 75%, and 1 for δ = 100%. For Default + SOS1 cuts, the average number of Flow cover cuts used was 82 for δ = 50%, 66 for δ = 75%, and 28 for δ = 100%; the average number of MIR cuts used was 222 for δ = 50%, 711 for δ = 75%, and 509 for δ = 100%; the average number of Gomory cuts used was 31 for δ = 50%, 17 for δ = 75%, and 39 for δ = 100%; the average number of 0-$\frac{1}{2}$ cuts used was 3 for δ = 50%, 2 for δ = 75%, and 1 for δ = 100%; and the average number of Mod k cuts used was 2 for δ = 50%, 1 for δ = 75%, and 1 for δ = 100%. Note that the set of cuts used here is similar to the set of cuts used for continuous CCOP. In particular, no cover cuts were used.

We now give the average number of SOS1 cuts separated and used. For B&B + SOS1 cuts, 1,482 SOS1 cuts were separated and 693 were used for δ = 50%; 1,969 were separated and 878 were used for δ = 75%; and 1,999 were separated and 438 used for δ = 100%. For Default + SOS1 cuts, 1,442 SOS1 cuts were separated and

353 were used for δ = 50%; 1,977 were separated and 316 were used for δ = 75%; and 2,000 were separated and 326 were used for δ = 100%.

We now give the average reduction in number of enumeration nodes due to the use of SOS1 cuts. For $n = 5$, Default's average number of nodes was 132,286. The average number of enumeration node reduction for B&B + SOS1 was 88%, and 85% for Default + SOS1. For $n \geq 10$, Default's average number of nodes was 50,821. The average reduction for B&B + SOS1 was 24%, and 74% for Default + SOS1 cuts

We conclude that the strategies using SOS1 cuts were more efficient than Default, which was more efficient than B&B. Overall, Default + SOS1 cuts was the best strategy for CCOP with 10-IP and 100-IP variables, both in terms of number of instances solved as well as computational time.

9 CCOP with 1,000-IP and 10,000-IP Variables

We now present the combined results of CCOP with 1,000-IP and 10,000-IP. These were the instances SOS1 cuts performed best.

9.1 Number of Instances Solved

Table 11 gives the number of instances solved for each strategy. As in Table 8, the maximum value for an entry in the table is 10. B&B solved 477 or 13% of the total, of which: 387, or 32% with δ = 50%; 60, or 5% with δ = 75%; and 30, or 3% with δ = 100%. In particular, for $n = 5$, 50% of the instances with δ = 50%, 36% of the instances with δ = 75%, and 25% of the instances with δ = 100% were solved; for $n \geq 10$, 30% of the instances with δ = 50%, 2% of the instances with δ = 75%, and 0% of the instances with δ = 100% were solved.

For B&B + SOS1 cuts, 1,241 or 34% of the total number of instances were solved, of which: 870, or 73% of the instances with δ = 50%; 287, or 24% of the instances with δ = 75%; and 84, or 7% of the instances with δ = 100%. In particular, for $n = 5$, 90% of the instances with δ = 50%, 66% of the instances with δ = 75%, and 52% of the instances with δ = 100% were solved; for $n \geq 10$, 71% of the instances with δ = 50%, 19% of the instances with δ = 75%, and 2% of the instances with δ = 100% were solved.

For Default, 941 or 26% of the total number of instances were solved, of which: 747, or 62% of the instances with δ = 50%; 150, or 13% of the instances with δ = 75%; and 44, or 4% of the instances with δ = 100%. In particular, for $n = 5$, 87% of the instances with δ = 50%, 58% of the instances with δ = 75%, and 32% of the instances with δ = 100% were solved; for $n \geq 10$, 60% of the instances with δ = 50%, 8% of the instances with δ = 75%, and 1% of the instances with δ = 100% were solved.

Finally, for Default + SOS1 cuts, 1,214 or 34% of the total number of instances were solved, of which: 870, or 73% of the instances with δ = 50%; 257, or 21% of the instances with δ = 75%; and 87, or 7% of the instances with δ = 100%. In particular, for $n = 5$, 90% of the instances with δ = 50%, 63% of the instances with δ = 75%, and

51% of the instances with δ = 100% were solved; for $n \geq 10$, 71% of the instances with δ = 50%, 17% of the instances with δ = 75%, and 2% of the instances with δ = 100% were solved.

As for the reduction in the percentage number of instances solved from small to large SOS1s, the percentage decreased from 24% to a maximum of 12% for B&B; 46% to a maximum of 30% for B&B + SOS1 cuts; 45% to a maximum of 29% for Default; and 51% to a maximum of 30% for Default + SOS1 cuts.

So, the strategy that solved the greatest number of instances was B&B + SOS1 cuts, followed by Default + SOS1 cuts, then by Default, and finally B&B. Note that B&B + SOS1 cuts increased the number of instances solved by Default by 32%. This is the first sign of the major role played by SOS1 cuts, as opposed to the minor role played by the MIP cuts. Here, as in the previous cases, CCOP becomes harder with increasing δ, and the decrease in the number of instances solved with δ is bigger for large than for small SOS1s. Also, the number of instances solved decreases when the SOS1s are large. Also, the number of instances solved decreased as we changed from small to large SOS1s.

9.2 Reduction in Computational Time

Here, the total number of instances considered is 1,257, of which: 880 with $\delta = 50\%$, 288 with $\delta = 75\%$, and 89 with $\delta = 100\%$. Figures 1 and 2 give the fraction of instances for which B&B + SOS1 cuts and Default + SOS1 cuts, respectively, were more or less efficient than Default, for 1,000-IP and 10,000-IP separately. The strategies using SOS1 cuts were more efficient than Default for considerably many more instances. Note that, on this respect, B&B + SOS1 cuts and Default + SOS1 cuts performed similarly.

For $n = 5$, B&B + SOS1 cuts reduced over the computational time of Default by at least 50% in 75% of the cases where %Red. B&B + SOS1 cuts was positive; Default + SOS1 cuts reduced over the computational time of Default by at least 50% in 76% of the cases where %Red. Default + SOS1 cuts was positive. For $n \geq 10$, B&B + SOS1 cuts reduced over the computational time of Default by at least 50% in 90% of the cases where %Red. B&B + SOS1 cuts was positive; and Default + SOS1 cuts in 90% of the cases where %Red. Default + SOS1 cuts was positive.

As in continuous, 10-IP, and 100-IP CCOP, the use of SOS1 cuts here improved the computational time of the instances solved by Default within a few seconds of CPU time. For $n = 5$, Default solved 58 instances in 10 seconds or less. B&B + SOS1 cuts was more efficient than Default for 36 of them, was less efficient for 3, and tied for the other 20. On average, B&B + SOS1 cuts reduced Default's computational time by 33%; and the reduction was of 50% or more for 29% of the instances. Default + SOS1 cuts was more efficient than Default for 40 instances, was less efficient for 3, and tied for the other 15. On average, Default + SOS1 cuts reduced Default's computational time by 40%; and the reduction was of 50% or more for 55% of the instances.

For $n \geq 10$, Default solved 149 instances in 10 seconds or less. B&B + SOS1 cuts was more efficient than Default for 113 of them, was less efficient for 1, and tied for

the other 35. On average, B&B + SOS1 cuts reduced Default's computational time by 65%; and the reduction was of 50% or more for 71% of the instances. Default + SOS1 cuts was more efficient than Default for 104 instances, was less efficient for 0, and tied for the other 45. On average, Default + SOS1 cuts reduced Default's computational time by 58%; and the reduction was of 50% or more for 61% of the instances.

In Tables 12 and 13 we give the average reductions in time for the instances with small and large SOS1s, respectively. For the instances with small SOS1s, all average reductions with B&B + SOS1 cuts are positive, except one; all average reductions with Default + SOS1 cuts are positive. With B&B + SOS1 cuts, the average reduction was at least 50% in 59% of the entries of Table 2; while with Default + SOS1 cuts, the average reduction was at least 50% in 72% of the entries. In a face-off, they tie in one entry, B&B + SOS1 cuts performs better 19 times, and Default + SOS1 cuts wins 12 times. The overall average reduction for B&B + SOS1 cuts was 71%, and the overall average reduction for Default + SOS1 cuts was 60%.

For the instances with large SOS1s, the average reductions for both B&B + SOS1 cuts and Default + SOS1 cuts are positive in all entries of Table 3; (note that the number of entries in Table 3 is smaller than in Table 2: 20, as opposed to 32). With B&B + SOS1 cuts, the average reduction was at least 50% in 85% of the entries of Table 3; while with Default + SOS1 cuts, the average reduction was at least 50% in 75% of the entries. In a face-off, Default + SOS1 cuts performs better 7 times, B&B + SOS1 cuts performs better 12 times, and they tie once. The overall average reduction for B&B + SOS1 cuts was 70%, and the overall average reduction for Default + SOS1 cuts was 63%.

9.3 Number of Cutting Planes and Reduction in Number of Enumeration Nodes

For both Default and Default + SOS1 cuts, the GUROBI MIP cuts used were Flow cover, MIR, and Gomory. For Default, the average number of Flow cover cuts used was 90 for δ = 50%, 71 for δ = 75%, and 22 for δ = 100%; the average number of MIR cuts used was 265 for δ = 50%, 770 for δ = 75%, and 414 for δ = 100%; the average number of Gomory cuts used was 32 for δ = 50%, 22 for δ = 75%, and 26 for δ = 100%. For Default + SOS1 cuts, the average number of Flow cover cuts used was 41 for δ = 50%, 26 for δ = 75%, and 8 for δ = 100%; the average number of MIR cuts used was 90 for δ = 50%, 367 for δ = 75%, and 236 for δ = 100%; the average number of Gomory cuts used was 22 for δ = 50%, 13 for δ = 75%, and 21 for δ = 100%. Note that the set of cuts used here is the same as the set of cuts used for continuous CCOP.

We now give the average number of SOS1 cuts separated and used. For B&B + SOS1 cuts, 1,392 SOS1 cuts were separated and 725 were used for δ = 50%; 1,929 were separated and 978 were used for δ = 75%; and 1,995 were separated and 686 used for δ = 100%. For Default + SOS1 cuts, 1,395 SOS1 cuts were separated and 494 were used for δ = 50%; 1,949 were separated and 507 were used for δ = 75%; and 2,000 were separated and 561 used for δ = 100%.

We now give the average reduction in number of enumeration nodes due to the use of SOS1 cuts. For $n = 5$, Default's average number of nodes was 202,318. The aver-

age number of enumeration nodes reductions for B&B + SOS1 and Default + SOS1 were similar, 87% and 84%, respectively. This is in accordance with the similar performance of B&B + SOS1 cuts and Default + SOS1 cuts in terms of computational time. For $n \geq 10$, Default's average number of nodes was 79,271. The average number of enumeration nodes reduction for B&B + SOS1 was 44%, while for Default + SOS1 cuts was 78%. This is in accordance with the superior performance, in terms of computational time, of Default + SOS1 cuts relative to B&B + SOS1 cuts.

We conclude that the strategies using SOS1 cuts were considerably more efficient than Default. Overall, B&B + SOS1 cuts was the best strategy for CCOP with 1,000-IP and 10,000-IP variables, both in terms of number of instances solved as well as being more efficient than Default + SOS1 cuts. Here, the SOS1 cuts improved considerably our ability to solve CCOP, while the outcome of using MIP cuts was negative.

10 Summary of Conclusions and Further Research

Regarding CCOP, it appears that increasing the cardinality of the SOS1s, and especially the average number of variables in a set with a nonzero coefficient in the constraint matrix, makes it harder. At first, this may seem obvious, since it ultimately means increasing the number of variables. However, the statement is not trivial, since at most one variable in each set is allowed to be nonzero.

Regarding cutting planes,

- it seems that adding (23) and (25) simultaneously is better than adding just one of them
- contrary to general belief, we found that separating our cover inequalities from all knapsacks was better than from just the ones active at the current LPR optimal solution
- SOS1 cuts proved critical in solving a wide variety of CCOPs:
  - For BIP CCOP, the important cutting planes were the MIP cuts. However, for the other CCOPs, SOS1 cuts reduced considerably the computational time required to solve them
  - for 1,000-IP CCOP and 10,000-IP CCOP the best strategy was to use SOS1 cuts only (i.e. to turn off MIP cuts). This is particularly surprising, as their structural variables are integer (as opposed to the binary variables added to continuous CCOP to enforce complemetarity)
- surprisingly, SOS1 cuts reduced considerably the computational time of instances solved by Default within a few seconds of CPU time
- for u-IP, SOS1 cuts seem to become more efficient as u increases.

We are now investigating the polyhedron that results when Assumption 3 is lifted. Several other research directions are possible from here. Three examples are:

- developing further the theory and computational methods for the "simple" cuts of Ibaraki [18] and Jeroslow [20]

- developing algorithms and computational facilities for lifting cover inequalities on-the-fly, in the spirit of Gu et al. [14], rather than using closed form inequalities as here
- strengthening our inequalities for GIP.

Acknowledgements This research was partially supported by the Office of Naval Research (ONR) through grants N000140910332 and N000141310041. ONR's support is gratefully acknowledged.

References

1. E. Balas, "Facets of the Knapsack Polytope," *Mathematical Programming* 8, 146-164 (1975).
2. E.M.L. Beale and J.A. Tomlin, "Special Facilities in a General Mathematical Programming System for Nonconvex Problems Using Ordered Sets of Variables," in J. Lawrence (Ed.), *Proceedings of the Fifth International Conference on Operations Research*, Tavistock Publications, 1970, pp. 447-454.
3. J. Bean, C. Noon, S. Ryan, and G. Salton, "Selecting Tenants in a Shopping Mall," *Interfaces* 18, 1–9 (1988).
4. R.W. Cottle, J.S. Pang, and R.E. Stone, *The Linear Complementarity Problem*, Academic Press, 1992.
5. H. Crowder, E.L. Johnson, and M. Padberg, "Solving Large-Scale Zero-One Linear Programming Problems," *Operations Research* 31, 803-834 (1983).
6. E. Danna, "Solving Optimization Problems at Google," Talk presented at the 2010 INFORMS National Meeting, Austin, TX, 2010.
7. G.B. Dantzig, "On the Significance of Solving Linear Programming Problems with some Integer Variables," *Econometrica* 28, 30-44 (1960).
8. I.R. de Farias JR., E.L. Johnson, and G.L. Nemhauser, "A Generalized Assignment Problem with Special Ordered Sets: A Polyhedral Approach," *Mathematical Programming* 89, 187–203 (2000).
9. I.R. de Farias, E.L. Johnson, and G.L. Nemhauser, "Branch-and-Cut for Combinatorial Optimization Problems without Auxiliary Binary Variables," *Knowledge Engineering Review* 16, 25-39 (2001).
10. I.R. de Farias JR., E.L. Johnson, and G.L. Nemhauser, "Facets of the Complementarity Knapsack Polytope," *Mathematics of Operations Research* 27, 210-226 (2002).
11. I.R. de Farias JR., E. Kozyreff, R. Gupta, and M. Zhao, "Branch-and-Cut for Separable Piecewise Linear Optimization and Intersection with Semi-Continuous Constraints," *Mathematical Programming C*, in press.
12. I.R. de Farias JR. and G.L. Nemhauser, "A Polyhedral Study of the Cardinality Constrained Knapsack Problem," *Mathematical Programming* 96, 439-467 (2003).
13. K. Dowsland, "Nurse Scheduling with Tabu Search and Strategic Oscillation," *European Journal of Operational Research* 106 393-407 (1998).
14. Z. Gu, G.L. Nemhauser, and M.W.P. Savelsbergh, "Cover Inequalities for 0-1 Linear Programs: Computation," *INFORMS Journal on Computing* 10, 427-437 (1998).
15. P.L. Hammer, E.L. Johnson, and U.N. Peled, "Facets of Regular 0-1 Polytopes," *Mathematical Programming* 8, 179-206. (1975).
16. http://www.hpcc.ttu.edu/index.php.
17. T. Ibaraki, "Complementary Convex Programming," *Journal of the Operations Research Society of Japan* 15, 138-160 (1972).
18. T. Ibaraki, "The use of Cuts in Complementarity Programming," *Operations Research* 21, 353-359 (1973).
19. T. Ibaraki, T. Hasegawa, J. Teranaka, and K. Iwasa, "The Multiple-Choice Knapsack Problem," *Journal of the Operations Research Society of Japan* 21, 59-93 (1978).
20. R.G. Jeroslow, "Cutting Planes for Complementarity Constraints," *SIAM Journal on Control and Optimization* 16, 56-62 (1978).
21. A.B. Keha, I.R. de Farias JR., and G.L. Nemhauser, "A Branch-and-Cut Algorithm without Binary Variables for Nonconvex Piecewise Linear Optimization," *Operations Research* 54, 847-858 (2006).
22. R.R. Meyer, "Integer and Mixed-Integer Programming Models: General Properties," *Journal of Optimization Theory and Applications* 16, 191-206 (1975).
23. R.R. Meyer, "Mixed-Integer Minimization Models for Piecewise-Linear Functions of a Single Variable," *Discrete Mathematics* 16, 163-171 (1976).

24. J.E. Mitchell, J.S. Pang, and B. Yu, "Obtaining Tighter Relaxations of Mathematical Programs with Complementarity Constraints," *Journal of Global Optimization*, in press.
25. G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization*, Wiley, 1988.
26. T.T. Nguyen, M. Tawarmalani, J.P.P. Richard, "Convexification Techniques for Linear Complementarity Constraints," in O. Günlük and G.J. Woeginger (Eds.), *Integer Programming and Combinatorial Optimization (IPCO)*, Lecture Notes in Computer Science, Vol. 6655, Springer, 20011, pp. 336-348.
27. H.D. Sherali and W.P. Adams, "A Hierarchy of Relaxations Between the Continuous and Convex Hull Representations for Zero-One Programming Problems," *SIAM Journal on Discrete Mathematics* 3, 411-430 (1990).
28. J.P. Vielma, S. Ahmed, and G.L. Nemhauser, "Mixed-Integer Models for Nonseparable Piecewise Linear Optimization: Unifying Framework and Extensions," *Operations Research* 58, 303-315 (2010).
29. J.P. Vielma and G.L. Nemhauser, "Modeling Disjunctive Constraints with a Logarithmic Number of Binary Variables and Constraints," *Mathematical Programming* 128, 49-72 (2011).
30. H.P. Williams, *Model Building in Mathematical Programming*, 4^{th} Edition, Wiley, 1999.
31. L.A. Wolsey, "Faces for a Linear Inequality in 0-1 Variables," *Mathematical Programming* 8, 165-178 (1975).
32. M. Zhao and I.R. de Farias JR., "The Piecewise Linear Optimization Polytope: New Inequalities and Intersection with Semi-Continuous Constraints," *Mathematical Programming A*, in press.

Fig. 1: % Time reduction: B&B + SOS1 cuts (top) vs. Default (bottom)

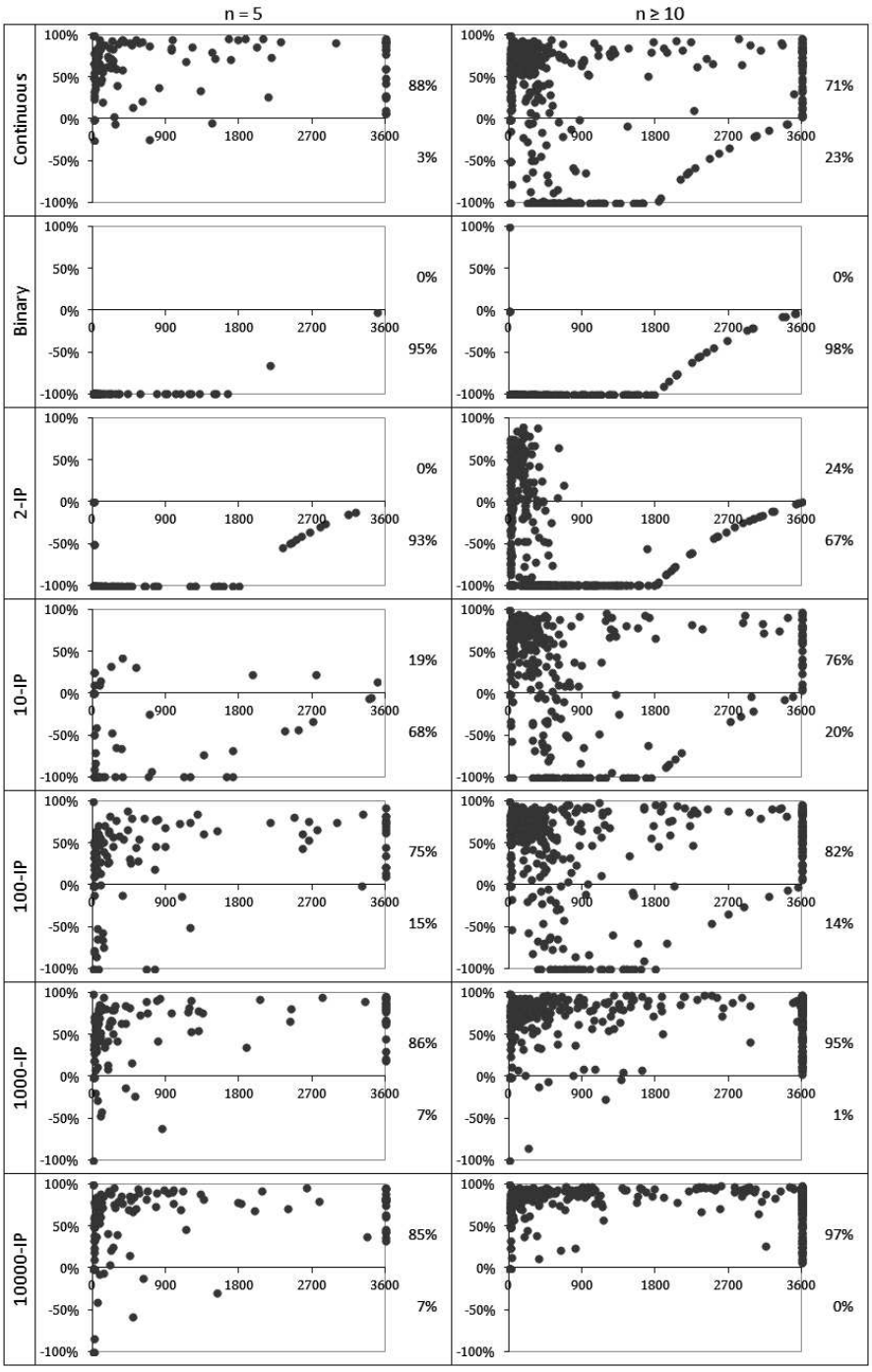

Fig. 2: % Time reduction: Default + SOS1 cuts (top) vs. Default (bottom)

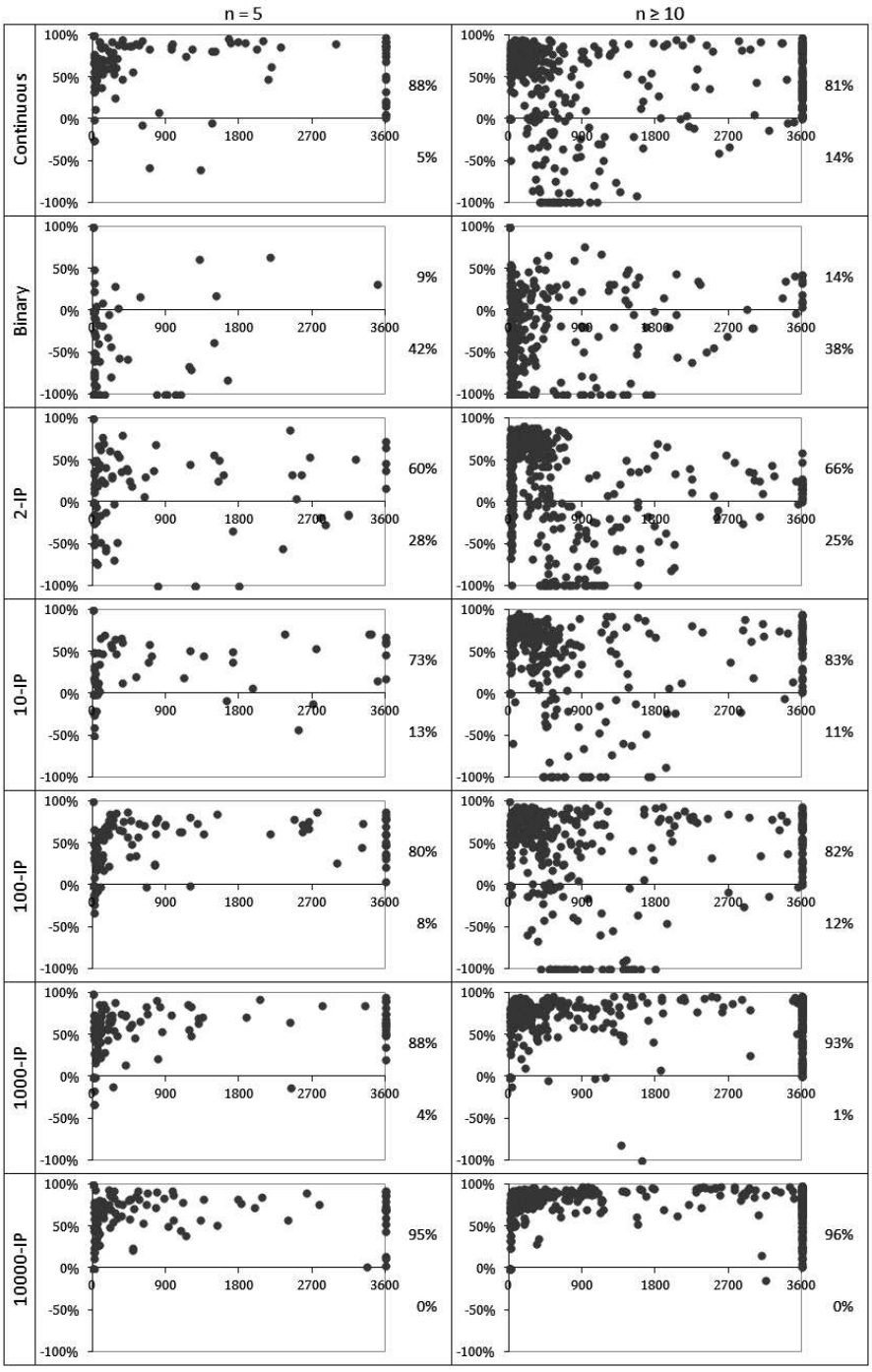

Table 1: Number of instances solved – Continuous variables

$n=$	5	10	15	20	25	30	35	40	45	50	5	10	15	20	25	30	35	40	45	50	5	10	15	20	25	30	35	40	45	50	5	10	15	20	25	30	35	40	45	50
$\delta.\ell.m$	B&B										B&B + SOS1 cuts										Default										Default + SOS1 cuts									
50.120.60	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.120.80	5	5	4	2	3	3	3	4	3	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.120.100	3	0	2	0	0	1	0	1	0	0	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.120.120	1	0	0	0	0	0	0	0	0	0	5	4	5	2	3	4	4	3	2	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.160.60	5	5	5	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.160.80	4	0	0	0	2	0	1	2	0	2	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.160.100	0	0	0	0	0	0	0	0	0	0	5	5	4	2	3	3	3	3	3	2	5	5	4	4	3	5	3	5	5	5	5	5	5	5	5	5	5	5	5	5
50.160.120	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	4	1	2	0	2	1	3	1	2	1	5	2	3	0	3	2	3	1	2	1
50.200.60	5	4	4	2	1	3	4	2	5	2	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.200.80	2	0	0	0	0	0	0	0	0	0	5	5	4	2	4	3	4	2	3	4	5	5	4	3	5	4	5	4	4	5	5	5	3	5	5	4	5	4	5	5
50.200.100	0	0	0	0	0	0	0	0	0	0	5	0	0	0	1	0	0	1	0	0	4	0	1	0	1	1	1	2	2	2	5	0	4	1	2	1	1	1	2	1
50.200.120	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
75.120.60	5	4	2	0	0	0	0	0	0	0	5	5	5	5	5	5	5	4	5	5	5	5	5	3	4	5	5	3	4	5	5	5	5	5	5	5	4	5	5	
75.120.80	5	2	0	0	0	0	0	0	0	0	5	5	4	3	1	2	1	0	0	0	5	5	2	2	0	2	0	0	0	0	5	5	5	4	1	3	2	2	2	1
75.120.100	0	0	0	0	0	0	0	0	0	0	5	1	1	0	0	0	0	0	0	0	5	1	0	0	0	0	0	0	0	0	5	2	1	0	0	0	0	0	0	0
75.120.120	1	0	0	0	0	0	0	0	0	0	5	1	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	5	2	0	0	0	0	0	0	0	0
75.160.60	5	0	0	0	0	0	0	0	0	0	5	5	5	4	2	1	1	0	2	2	5	5	2	2	1	0	0	0	0	0	5	5	5	4	2	2	3	0	3	3
75.160.80	1	0	0	0	0	0	0	0	0	0	5	2	1	0	0	0	0	0	0	0	5	2	0	0	0	0	0	0	0	0	5	2	1	0	0	0	0	0	0	0
75.160.100	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
75.160.120	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
75.200.60	4	0	0	0	0	0	0	0	0	0	5	3	2	1	0	0	0	0	0	0	5	1	2	0	0	0	0	0	0	0	5	3	2	0	0	0	1	0	0	0
75.200.80	1	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0
75.200.100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
75.200.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
100.120.60	5	0	0	0	0	0	0	0	0	0	5	5	4	1	0	0	0	1	1	0	5	2	1	0	0	0	0	1	0	0	5	4	4	2	0	1	0	1	1	0
100.120.80	2	0	0	0	0	0	0	0	0	0	5	2	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	5	2	0	0	0	0	0	0	0	0
100.120.100	1	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0
100.120.120	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
100.160.60	5	0	0	0	0	0	0	0	0	0	5	3	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	5	4	0	0	0	0	0	0	0	0
100.160.80	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0
100.160.100	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
100.160.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
100.200.60	3	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	5	0	1	0	0	0	0	0	0	0
100.200.80	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
100.200.100	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
100.200.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
% Solved	38	14	12	7	9	9	10	11	10	11	74	42	36	28	27	27	27	24	26	27	63	37	32	27	28	30	29	28	29	29	74	45	41	34	32	32	33	29	33	31

Table 2: Average solution time with Default and % Reduction with B&B + SOS1 cuts and Default + SOS1 cuts – Continuous variables and $n = 5$

$\delta.\ell.m$	Time Def.	% Red. BB+C	% Red. D+C	$\delta.\ell.m$	Time Def.	% Red. BB+C	% Red. D+C
50.120.60	1	50	50	50.120.80	1	40	60
50.120.100	38	65	71	50.120.120	99	55	68
50.160.60	1	43	57	50.160.80	22	46	63
50.160.100	256	19	0	50.160.120	1131	39	18
50.200.60	6	40	23	50.200.80	144	52	66
50.200.100	1,401	36	14	50.200.120	480	14	57
75.120.60	10	72	56	75.120.80	39	74	68
75.120.100	545	85	82	75.120.120	829	50	62
75.160.60	36	67	67	75.160.80	776	86	84
75.160.100	2,954	46	18	75.160.120	2897	15	21
75.200.60	788	77	70	75.200.80	2709	20	34
100.120.60	139	93	90	100.120.80	1342	93	93
100.120.100	2,511	67	76	100.120.120	3600	93	93
100.160.60	1,097	96	93	100.160.80	3600	65	73
100.160.100	2,032	17	10	100.200.60	1192	90	88
100.200.80	3,600	71	49	100.200.100	3600	11	2

Table 3: Average solution time with Default and % Reduction with B&B + SOS1 cuts and Default + SOS1 cuts – Continuous variables and $n \geq 10$

$\delta.\ell.m$	Time Def.	% Red. BB+C	% Red. D+C	$\delta.\ell.m$	Time Def.	% Red. BB+C	% Red. D+C
50.120.60	3	50	39	50.120.80	43	77	83
50.120.100	155	-13	58	50.120.120	450	-241	-6
50.160.60	27	79	83	50.160.80	191	22	67
50.160.100	1,004	-107	-2	50.160.120	1846	-95	-14
50.200.60	130	73	78	50.200.80	1084	-49	10
50.200.100	2,568	-34	3	50.200.120	2566	-40	-40
75.120.60	1,113	84	88	75.120.80	2371	31	51
75.120.100	2,692	9	59	75.120.120	3600	3	18
75.160.60	2,706	46	54	75.160.80	2073	74	69
75.200.60	2,715	52	49	100.120.60	2995	55	68
100.120.80	3,600	83	84	100.160.60	3600	35	40
100.200.60	3,600	0	2				

Table 4: Number of instances solved – Binary variables

$n=$	5	10	15	20	25	30	35	40	45	50	5	10	15	20	25	30	35	40	45	50	5	10	15	20	25	30	35	40	45	50	5	10	15	20	25	30	35	40	45	50
$\delta.\ell.m$	B&B										B&B + SOS1 cuts										Default										Default + SOS1 cuts									
50.120.60	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.120.80	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.120.100	5	3	4	4	4	5	3	5	5	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.120.120	3	0	1	0	1	1	3	2	1	2	5	4	5	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.160.60	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.160.80	5	4	5	5	5	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.160.100	2	0	1	2	0	0	0	0	1	1	4	4	5	3	3	4	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.160.120	0	0	0	0	0	0	0	0	0	0	2	0	1	0	2	0	3	1	1	2	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.200.60	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.200.80	5	2	1	1	3	1	0	2	3	3	5	5	4	5	4	4	5	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.200.100	0	0	0	0	0	0	0	0	0	0	2	0	1	0	1	1	1	1	2	1	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.200.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
75.120.60	5	5	5	4	4	4	5	3	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
75.120.80	5	3	0	0	0	0	0	0	0	0	5	4	4	4	1	5	3	3	2	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
75.120.100	4	0	0	0	0	0	0	0	0	0	5	1	0	0	0	0	0	0	0	0	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
75.120.120	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	5	5	5	5	4	5	5	2	5	5	5	5	5	4	4	5	4	3	5	5
75.160.60	5	4	2	1	1	0	1	0	0	0	5	5	5	4	3	3	5	3	3	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
75.160.80	5	0	0	0	0	0	0	0	0	0	5	2	0	0	0	0	0	0	0	0	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
75.160.100	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	5	4	3	1	1	2	3	0	2	3	5	4	3	1	1	3	3	1	2	3
75.160.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	3	2	0	0	0	0	0	0	0	5	3	2	0	0	0	0	0	0	0
75.200.60	5	1	0	0	0	0	0	0	0	0	5	4	4	1	0	0	1	0	0	0	5	5	5	5	4	5	5	5	5	5	5	5	5	5	4	5	5	5	5	5
75.200.80	2	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	5	5	5	2	1	3	2	2	3	3	5	5	5	1	0	3	2	1	2	2
75.200.100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	1	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0
75.200.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0
100.120.60	5	1	0	0	0	0	0	0	0	0	5	1	0	0	0	0	0	1	0	0	5	5	5	5	3	3	3	2	2	2	5	5	5	5	3	3	4	2	2	1
100.120.80	3	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	5	4	3	1	0	1	0	0	0	0	5	4	3	2	0	1	0	0	0	0
100.120.100	2	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	5	3	1	0	0	0	0	0	0	0	5	2	1	0	0	0	0	0	0	0
100.120.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	2	0	0	0	0	0	0	0	0	5	2	0	0	0	0	0	0	0	0
100.160.60	5	0	0	0	0	0	0	0	0	0	5	1	0	0	0	0	0	0	0	0	5	5	5	5	1	2	1	0	0	0	5	5	5	5	1	1	1	0	0	0
100.160.80	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	5	1	0	0	0	0	0	0	0	0	5	2	1	0	0	0	0	0	0	0
100.160.100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0
100.160.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0
100.200.60	5	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	5	5	4	1	0	0	0	0	0	0	5	5	4	1	0	0	0	0	0	0
100.200.80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	1	0	0	0	0	0	0	0	0	5	1	0	0	0	0	0	0	0	0
100.200.100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0
100.200.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
% Solved	51	24	22	21	21	19	21	21	22	22	60	37	36	31	30	32	34	32	32	34	97	74	68	61	55	59	58	53	57	57	97	74	69	61	54	59	58	54	56	56

Table 5: Number of instances solved – 2-IP variables

$n =$	5	10	15	20	25	30	35	40	45	50	5	10	15	20	25	30	35	40	45	50	5	10	15	20	25	30	35	40	45	50	5	10	15	20	25	30	35	40	45	50
$\delta.\ell.m$	B&B										B&B + SOS1 cuts										Default										Default + SOS1 cuts									
50.120.60	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.120.80	4	4	4	3	3	3	4	5	3	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.120.100	1	0	0	1	0	0	0	1	0	0	3	4	5	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.120.120	0	0	0	0	0	0	0	0	0	0	1	1	4	2	3	3	4	3	3	3	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.160.60	5	5	5	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.160.80	2	0	0	0	2	0	2	2	1	2	4	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.160.100	0	0	0	0	0	0	0	0	0	0	1	0	3	2	3	3	2	3	4	1	4	4	5	5	5	5	5	5	5	5	4	4	5	5	5	5	5	5	5	5
50.160.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	2	2	2	4	4	4	5	4	4	3	1	3	3	4	5	4	5	4	4
50.200.60	4	0	2	2	2	4	4	2	5	3	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
50.200.80	1	0	0	0	0	0	0	0	0	0	1	2	2	2	3	2	3	2	3	4	5	5	5	5	5	4	5	5	5	5	5	4	5	5	5	5	5	5	5	5
50.200.100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	2	1	2	2	4	5	3	4	2	0	2	1	2	2	4	5	5	5
50.200.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	1	0	0	0	0	1	1	2	0	0
75.120.60	3	0	0	0	0	0	0	0	0	0	4	4	5	2	3	4	5	4	4	5	5	4	5	3	5	5	5	4	4	5	5	5	5	4	5	5	5	4	4	5
75.120.80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	2	2	1	0	3	1	1	2	1	5	3	3	1	0	2	1	1	1	1
75.120.100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	1	0	3	0	0	0	0	0	0	0	0	0
75.120.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
75.160.60	2	0	0	0	0	0	0	0	0	0	2	0	0	2	0	0	0	0	0	0	5	1	2	2	0	0	1	1	1	1	5	1	2	2	1	0	1	1	1	1
75.160.80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
75.160.100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
75.160.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
75.200.60	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	4	0	1	1	0	0	0	0	0	0	3	0	1	1	0	0	0	0	0	0
75.200.80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
75.200.100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
75.200.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
100.120.60	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	3	1	0	0	0	0	0	0	0	0	4	1	0	0	0	0	0	0	0	0
100.120.80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0
100.120.100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
100.120.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
100.160.60	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0
100.160.80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
100.160.100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
100.160.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
100.200.60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0
100.200.80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
100.200.100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
100.200.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
% Solved	16	8	9	8	9	9	11	11	11	11	22	19	24	22	23	23	24	23	24	24	51	30	33	31	31	33	33	34	33	33	50	30	34	32	32	33	34	35	33	34

Table 6: Average solution time with Default and % Reduction with B&B + SOS1 cuts and Default + SOS1 cuts – 2-IP variables and $n = 5$

$\delta.\ell.m$	Time Def.	% Red. BB+C	% Red. D+C	$\delta.\ell.m$	Time Def.	% Red. BB+C	% Red. D+C
50.120.60	1	-175	25	50.120.80	3	-3976	12
50.120.100	38	-4138	58	50.120.120	68	-4216	40
50.160.60	1	-86	43	50.160.80	55	-2394	61
50.160.100	1,525	-100	33	50.160.120	578	-523	40
50.200.60	23	-2425	53	50.200.80	261	-1004	-86
50.200.100	1,929	-87	45	50.200.120	1715	-110	-34
75.120.60	158	-512	35	75.120.80	2082	-73	52
75.120.100	214	-1580	30	75.120.120	1747	-106	-5
75.160.60	614	-268	-19	75.160.80	2182	-65	26
75.160.100	2,853	-26	-26	75.200.60	1132	-149	-27
75.200.80	3,137	-15	-15	100.120.60	1096	-212	39
100.120.80	1,499	-140	40	100.120.100	292	-1133	-47
100.160.60	919	-240	13	100.200.60	1599	-125	-16

Table 7: Average solution time with Default and % Reduction with B&B + SOS1 cuts and Default + SOS1 cuts – 2-IP variables and $n \geq 10$

$\delta.\ell.m$	Time Def.	% Red. BB+C	% Red. D+C	$\delta.\ell.m$	Time Def.	% Red. BB+C	% Red. D+C
50.120.60	2	20	30	50.120.80	21	28	70
50.120.100	77	-394	63	50.120.120	259	-681	-18
50.160.60	16	38	62	50.160.80	123	-204	60
50.160.100	650	-286	14	50.160.120	1574	-129	-20
50.200.60	80	-63	60	50.200.80	653	-236	-29
50.200.100	1,665	-116	-16	50.200.120	3142	-15	2
75.120.60	642	-98	46	75.120.80	1913	-88	8
75.120.100	2,014	-79	-79	75.160.60	2071	-64	14
75.200.60	1,977	-82	0	100.120.60	566	-536	78

Table 8: Number of instances solved – 10-IP and 100-IP variables

$n=$	5	10	15	20	25	30	35	40	45	50	5	10	15	20	25	30	35	40	45	50	5	10	15	20	25	30	35	40	45	50	5	10	15	20	25	30	35	40	45	50
$\delta.\ell.m$	B&B										B&B + SOS1 cuts										Default										Default + SOS1 cuts									
50.120.60	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
50.120.80	10	9	8	4	6	6	8	8	8	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
50.120.100	4	0	2	0	0	1	0	2	0	0	9	9	10	10	10	10	10	10	10	10	9	9	10	10	10	10	10	10	10	10	10	9	10	10	10	10	10	10	10	10
50.120.120	1	0	0	0	0	0	0	0	0	0	6	5	10	5	7	9	9	6	9	9	8	7	10	9	7	10	10	10	10	10	10	7	10	9	8	10	10	10	10	10
50.160.60	10	10	10	8	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
50.160.80	6	0	0	0	4	0	2	4	2	4	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
50.160.100	0	0	0	0	0	0	0	0	0	0	7	5	8	5	6	7	7	9	8	4	7	5	8	5	6	10	7	10	10	9	8	5	8	7	6	10	10	10	10	8
50.160.120	0	0	0	0	0	0	0	0	0	0	3	0	0	0	1	0	2	0	0	0	4	0	0	0	2	1	6	2	6	2	5	0	0	0	1	1	6	2	3	3
50.200.60	9	7	7	4	2	6	8	3	10	6	10	10	10	10	10	10	10	10	10	10	10	10	10	9	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
50.200.80	3	0	0	0	0	0	0	0	0	0	7	10	6	4	9	6	9	5	8	10	9	6	4	5	8	8	10	5	8	10	9	9	6	5	9	8	9	6	9	10
50.200.100	0	0	0	0	0	0	0	0	0	0	4	0	0	0	1	0	0	2	0	1	4	0	0	0	2	2	0	2	4	4	5	0	0	0	2	1	0	2	2	2
50.200.120	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
75.120.60	8	5	4	0	0	0	0	0	0	0	10	9	10	8	10	10	10	7	10	10	10	9	9	4	7	9	8	3	7	8	10	10	10	8	10	10	10	7	9	10
75.120.80	5	1	0	0	0	0	0	0	0	0	7	6	6	4	2	5	3	1	2	1	8	4	3	2	0	0	1	0	0	0	8	7	5	3	1	3	2	1	1	2
75.120.100	0	0	0	0	0	0	0	0	0	0	6	1	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	7	1	0	0	0	0	0	0	0	0
75.120.120	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0
75.160.60	7	0	0	0	0	0	0	0	0	0	8	5	6	6	3	1	4	0	2	4	10	2	2	4	0	0	0	0	0	0	9	5	4	6	1	0	1	0	1	4
75.160.80	1	0	0	0	0	0	0	0	0	0	5	2	1	0	0	0	0	0	0	0	6	1	0	0	0	0	0	0	0	0	6	2	0	0	0	0	0	0	0	0
75.160.100	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
75.160.120	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
75.200.60	2	0	0	0	0	0	0	0	0	0	7	3	2	0	0	0	1	0	0	0	7	1	1	0	0	0	0	0	0	0	7	1	2	0	0	0	0	0	0	0
75.200.80	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
75.200.100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
75.200.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
100.120.60	4	0	0	0	0	0	0	0	0	0	6	4	0	0	0	0	0	0	0	0	6	1	0	0	0	0	0	0	0	0	6	3	0	0	0	0	0	0	0	0
100.120.80	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0
100.120.100	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0
100.120.120	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
100.160.60	3	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0
100.160.80	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
100.160.100	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
100.160.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
100.200.60	3	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0
100.200.80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
100.200.100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
100.200.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
% Solved	24	12	11	7	9	9	11	10	11	11	46	30	30	26	28	27	29	25	28	28	45	26	27	24	26	28	28	26	29	29	51	30	29	27	27	29	30	27	29	30

Table 9: Average solution time with Default and % Reduction with B&B + SOS1 cuts and Default + SOS1 cuts – 10-IP and 100-IP variables and $n = 5$

$\delta.\ell.m$	Time Def.	% Red. BB+C	% Red. D+C	$\delta.\ell.m$	Time Def.	% Red. BB+C	% Red. D+C
50.120.60	2	-42	33	50.120.80	4	-80	25
50.120.100	408	-45	22	50.120.120	1171	-27	58
50.160.60	3	6	12	50.160.80	92	-92	62
50.160.100	778	-19	60	50.160.120	1183	-84	22
50.200.60	23	-182	27	50.200.80	524	-104	31
50.200.100	2,283	16	45	50.200.120	1088	-12	65
75.120.60	104	-13	29	75.120.80	556	-73	39
75.120.100	1,974	43	70	75.120.120	1789	22	55
75.160.60	926	-18	-6	75.160.80	1203	22	41
75.160.100	2,467	82	79	75.160.120	3600	74	32
75.200.60	1,162	10	51	75.200.80	65	-1066	34
100.120.60	508	-3	67	100.120.80	2018	67	72
100.120.100	2,494	44	56	100.120.120	3600	53	53
100.160.60	2,560	66	75	100.160.80	3600	34	17
100.160.100	2,754	67	88	100.200.60	1785	69	63

Table 10: Average solution time with Default and % Reduction with B&B + SOS1 cuts and Default + SOS1 cuts – 10-IP and 100-IP variables and $n \geq 10$

$\delta.\ell.m$	Time Def.	% Red. BB+C	% Red. D+C	$\delta.\ell.m$	Time Def.	% Red. BB+C	% Red. D+C
50.120.60	4	65	59	50.120.80	48	80	81
50.120.100	175	34	59	50.120.120	577	-89	10
50.160.60	35	85	79	50.160.80	256	63	60
50.160.100	1,040	-40	-2	50.160.120	1555	-120	-66
50.200.60	195	80	74	50.200.80	1279	8	18
50.200.100	1,435	-110	-72	75.120.60	1738	82	74
75.120.80	2,964	52	44	75.120.100	3600	65	84
75.160.60	3,172	68	41	75.160.80	2810	68	25
75.200.60	3,219	65	33	100.120.60	2991	61	49

Table 11: Number of instances solved – 1000-IP and 10000-IP variables

$n=$	5	10	15	20	25	30	35	40	45	50	5	10	15	20	25	30	35	40	45	50	5	10	15	20	25	30	35	40	45	50	5	10	15	20	25	30	35	40	45	50
$\delta.\ell.m$	B&B										B&B + SOS1 cuts										Default										Default + SOS1 cuts									
50.120.60	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
50.120.80	10	10	8	5	6	6	7	8	8	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
50.120.100	6	0	4	0	0	2	0	2	0	0	10	10	10	10	10	10	10	10	10	10	10	10	10	9	9	10	9	9	10	9	10	10	10	10	10	10	10	10	10	10
50.120.120	2	0	0	0	0	0	0	0	0	0	10	10	10	4	8	9	10	6	10	10	10	8	10	4	6	7	8	5	6	5	10	10	10	8	7	10	10	7	9	10
50.160.60	10	10	10	8	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
50.160.80	8	0	0	0	4	0	2	4	2	5	10	10	10	10	10	10	10	10	10	10	10	10	10	9	9	10	10	9	8	10	10	10	10	10	10	10	10	10	10	10
50.160.100	0	0	0	0	0	0	0	0	0	0	10	10	8	6	6	8	6	10	8	4	10	6	8	3	1	2	1	2	4	2	10	7	8	5	6	8	6	10	8	5
50.160.120	0	0	0	0	0	0	0	0	0	0	8	0	0	0	1	0	3	0	0	0	6	0	0	0	0	0	0	0	0	0	8	0	0	0	2	0	2	0	0	0
50.200.60	10	8	8	4	2	6	8	4	10	6	10	10	10	10	10	10	10	10	10	10	10	10	10	9	10	10	10	7	10	10	10	10	10	10	10	10	10	10	10	10
50.200.80	4	0	0	0	0	0	0	0	0	0	10	10	8	4	9	6	10	5	7	10	10	6	4	3	2	1	2	4	0	5	10	10	7	4	8	6	10	6	8	10
50.200.100	0	0	0	0	0	0	0	0	0	0	8	0	0	0	2	0	0	2	1	1	6	0	0	0	1	0	0	1	0	0	8	0	0	0	2	0	0	2	0	1
50.200.120	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
75.120.60	10	8	4	0	0	0	0	0	0	0	10	10	10	10	10	10	10	9	10	10	10	10	10	6	2	6	3	2	5	3	10	10	10	10	10	10	10	8	10	10
75.120.80	10	4	0	0	0	0	0	0	0	0	10	10	9	8	2	7	4	3	2	2	10	9	4	0	0	0	0	0	0	0	10	10	6	6	2	4	4	0	3	2
75.120.100	0	0	0	0	0	0	0	0	0	0	10	2	1	0	0	0	0	0	0	0	10	1	0	0	0	0	0	0	0	0	10	2	0	0	0	0	0	0	0	0
75.120.120	1	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0
75.160.60	10	1	0	0	0	0	0	0	0	0	10	10	10	8	4	2	7	1	4	5	10	7	4	3	0	0	0	0	0	0	10	10	7	7	4	0	3	0	3	5
75.160.80	2	0	0	0	0	0	0	0	0	0	10	4	2	0	0	0	0	0	0	0	9	2	0	0	0	0	0	0	0	0	10	4	2	0	0	0	0	0	0	0
75.160.100	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
75.160.120	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
75.200.60	8	0	0	0	0	0	0	0	0	0	10	6	4	0	0	0	2	0	0	0	8	2	2	0	0	0	0	0	0	0	10	5	4	0	0	0	0	0	0	0
75.200.80	2	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
75.200.100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
75.200.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
100.120.60	9	0	0	0	0	0	0	0	0	0	10	8	6	0	0	0	0	2	1	0	10	4	0	0	0	0	0	2	0	0	10	8	5	1	0	0	0	2	2	0
100.120.80	4	0	0	0	0	0	0	0	0	0	10	3	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	10	4	0	0	0	0	0	0	0	0
100.120.100	1	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0
100.120.120	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0
100.160.60	10	0	0	0	0	0	0	0	0	0	10	2	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	10	2	0	0	0	0	0	0	0	1
100.160.80	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	1	0	0	0
100.160.100	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
100.160.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
100.200.60	6	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0
100.200.80	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
100.200.100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
100.200.120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
% Solved	37	14	12	8	9	9	10	11	11	11	69	40	36	28	28	28	31	27	29	28	59	32	28	21	19	21	20	20	20	21	68	39	33	28	28	27	29	26	29	29

Table 12: Average solution time with Default and % Reduction with B&B + SOS1 cuts and Default + SOS1 cuts – 1000-IP and 10000-IP variables and $n = 5$

$\delta.\ell.m$	Time Def.	% Red. BB+C	% Red. D+C	$\delta.\ell.m$	Time Def.	% Red. BB+C	% Red. D+C
50.120.60	1	33	56	50.120.100	53	68	68
50.120.80	4	41	74	50.120.120	104	30	62
50.120.120	104	30	62	50.160.60	3	27	50
50.160.80	37	49	66	50.160.100	450	16	62
50.160.120	1,146	20	34	50.200.60	10	39	31
50.200.80	217	53	66	50.200.100	2320	63	58
50.200.120	554	-17	51	75.120.60	14	66	51
75.120.80	62	76	64	75.120.100	742	82	68
75.120.120	1,201	48	69	75.160.60	37	35	40
75.160.80	1,221	84	77	75.160.100	2188	54	36
75.160.120	3,600	87	72	75.200.60	1016	55	47
75.200.80	1,820	40	0	100.120.60	137	84	75
100.120.80	1,808	90	78	100.120.100	2645	72	71
100.120.120	3,600	90	86	100.160.60	1751	95	85
100.160.80	3,600	75	30	100.160.100	818	85	86
100.200.60	1,434	79	59	100.200.80	3600	82	35

Table 13: Average solution time with Default and % Reduction with B&B + SOS1 cuts and Default + SOS1 cuts – 1000-IP and 10000-IP variables and $n \geq 10$

$\delta.\ell.m$	Time Def.	% Red. BB+C	% Red. D+C	$\delta.\ell.m$	Time Def.	% Red. BB+C	% Red. D+C
50.120.60	4	70	62	50.120.100	511	85	87
50.120.80	55	86	86	50.120.120	1681	55	62
50.120.120	1,681	55	62	50.160.60	40	89	87
50.160.80	642	89	88	50.160.100	2626	69	63
50.160.120	3,600	29	32	50.200.60	466	93	92
50.200.80	2,731	72	65	50.200.100	2971	34	38
75.120.60	2,181	92	86	75.120.80	2867	62	45
75.120.100	3,261	56	55	75.160.60	3038	68	38
75.160.80	2,760	78	60	75.200.60	3313	74	47
100.120.60	3,073	54	65	100.160.60	3600	15	50