Modified Orbital Branching with Applications to Orbitopes and to Unit Commitment

James Ostrowski* Miguel F. Anjos† Anthony Vannelli‡

October 27, 2012

Abstract

The past decade has seen advances in general methods for symmetry breaking in mixed-integer linear programming. These methods are advantageous for general problems with general symmetry groups. Some important classes of MILP problems, such as bin packing and graph coloring, contain highly structured symmetry groups. This observation has motivated the development of problem-specific techniques. In this paper we show how to strengthen orbital branching in order to exploit special structures in a problem's symmetry group. The proposed technique, to which we refer as modified orbital branching, is able to solve problems with structured symmetry groups more efficiently. One class of problems for which this technique is effective is when the solution variables can be expressed as orbitopes, as this technique extends the classes of orbitopes where symmetry can be efficiently removed. We use the unit commitment problem, an important problem in power systems, to demonstrate the strength of modified orbital branching.

1 Introduction

The presence of symmetry in mixed integer linear programming (MILP) has long caused computational difficulties. Jeroslow [8] introduced a class of simple integer programming problems that require an exponential number of subproblems when using pure branch and bound due to the problem's large symmetry group. The past decade has seen advances in general methods for symmetry breaking, most notably isomorphism pruning [13, 14] and orbital branching [17]. These methods are advantageous for general problems with general symmetry

*Department of Industrial and Information Engineering, University of Tennessee Knoxville, Knoxville, TN, United States. jostrows@utk.edu

†Canada Research Chair in Discrete Nonlinear Optimization in Engineering, GERAD & École Polytechnique de Montréal, Montréal, QC, Canada H3C 3A7. anjos@stanfordalumni.org

‡School of Engineering, University of Guelph, Guelph, ON, Canada N1G 2W1. vannelli@uoguelph.ca

groups. There are important classes of MILP problems, such as bin packing and graph coloring, that contain highly structured symmetry groups which can be exploited. This observation has motivated the development of problem-specific techniques. For example, orbitopal fixing [11, 10] is an efficient way to break symmetry in bin packing problems. Symmetry breaking constraints can be added to formulations of telecommunication problems, noise pollution problems, and others, see e.g. [19].

In some cases, a general symmetry breaking-method such as orbital branching can be modified to take advantage of special structure. For example, consider the Jeroslow problem

$$\begin{array}{ll} \min & x_{n+1} \\ \text{s.t.} & \sum_{i=1}^{n} 2x_i + x_{n+1} = 2\lfloor \frac{n}{2} \rfloor + 1 \\ & x_i \in \{0,1\}\ \forall i \in \{1,\ \ldots,\ n+1\}, \end{array} \tag{1}$$

for any positive integer n. This problem contains a large amount of symmetry. Using pure branch-and-bound, solving (1) will require an exponentially large branch-and-bound tree [8]. However, this symmetry is very well structured. Every permutation of two variables among $\{x_1, \ldots,\ x_n\}$, while leaving the remaining variables unchanged, is a symmetry. The only permutations that are not symmetries are those that move the variable x_{n+1}. Because of this special structure, the constraints $x_i \geq x_{i+1}$, for $i \in \{1, \ldots, n-1\}$, can be added to the problem to break the symmetry. With these constraints, a pure branch-and-bound approach can solve (1) using just one branch. Similarly, the problem can be reformulated by letting $y = \sum_{i=1}^{n} x_i$; the resulting formulation is easy to solve. General symmetry-breaking methods are less efficient. Both isomorphism pruning [13, 14] and orbital branching [17] need $\lfloor \frac{n}{2} \rfloor + 1$ branches to solve (1).

In this paper we show how to strengthen orbital branching in cases where the problem's symmetry group contains additional structure. The proposed technique, to which we refer as modified orbital branching, is able to solve problems with structured symmetry groups more efficiently. For example, modified orbital branching solves (1) in one branch. This improvement is especially useful for problems that can be expressed as orbitopes, e.g. like those studied in [11, 10]. Modified orbital branching strengthens that work by extending the classes of orbitopes for which symmetry can be efficiently removed.

This paper is structured as follows. Section 2 provides an overview of symmetry in integer programming, as well as a description of orbital branching. Section 3 shows how orbital branching can be strengthened when the problem's symmetry group has a special structure. Section 4 shows how modified orbital branching can be used to efficiently remove all isomorphic solutions from MILP problems that can be expressed as orbitopes. Section 5 uses the unit commitment (UC) problem, an important problem in power systems that can exhibit a special symmetric structure, to demonstrate the strength of modified orbital branching. Section 6 concludes the paper.

2 Symmetry and Orbital Branching

2.1 Background on Symmetry

The set S^n is the set of all permutations of $I^n = \{1, \ldots, n\}$. This set forms the *full symmetric group* of I^n. Any subgroup of the full symmetric group is a *permutation group*. For any permutation group Γ, the following hold:

- Γ contains the *identity* permutation e.
- If $\pi \in \Gamma$, then $\pi^{-1} \in \Gamma$, where π^{-1} is the *inverse* permutation.
- For π_1 and $\pi_2 \in \Gamma$, the *composition* $\pi_1 \circ \pi_2$ that maps x to $\pi_1(\pi_2(x))$ is in Γ.

The permutation $\pi \in \Gamma$ maps the point $z \in \mathbb{R}^n$ to $\pi(z)$ by permuting the indices. The permutation group Γ acts on a set of points $\mathcal{Z} \subset \mathbb{R}^n$ such that if $z \in \mathcal{Z}$ then $\pi(z) \in \mathcal{Z}$. Let $\mathcal{F}$ denote the feasible set of a given MILP. The *symmetry group* $\mathcal{G}$ of an MILP is the set of permutations of the variables that maps each feasible solution onto a feasible solution of the same value:

$$\mathcal{G} \stackrel{\text{def}}{=} \{\pi \in \Pi^n \mid \pi(x) \in \mathcal{F} \quad \forall x \in \mathcal{F} \text{ and } c^T x = c^T \pi(x)\}.$$

The *orbit* of z under the action of the group Γ is the set of all elements of $\mathcal{Z}$ to which z can be sent by permutations in Γ:

$$\text{orb}(z, \Gamma) \stackrel{\text{def}}{=} \{\pi(z) \mid \pi \in \Gamma\}.$$

Orbits can also be extended to variables. By definition, if e_j (the jth unit vector) is in $\text{orb}(\{e_k\}, \Gamma)$, then $e_k \in \text{orb}(\{e_j\}, \Gamma)$, i.e. the variable x_j and x_k share the same orbit. Therefore, the union of the orbits

$$\mathcal{O}(\Gamma) \stackrel{\text{def}}{=} \bigcup_{j=1}^{n} \text{orb}(\{e_j\}, \Gamma) \tag{2}$$

forms a partition of I^n.

To study how Γ acts on a subset of I^n, we *project* Γ. The projection of Γ on $J \subseteq I^n$, $\texttt{Proj}_J(\Gamma)$, is the set of permutations π_P such that

$$\pi_P \in \texttt{Proj}_J(\Gamma) \Leftrightarrow \exists \pi \in \Gamma \text{ s.t. } \pi(j) = \pi_P(j) \; \forall \; j \in J. \tag{3}$$

Note that it only makes sense to project the symmetry group Γ onto orbits or unions of orbits.

If Γ is the set of all permutations of the elements $\{i_1, \ldots, \; i_n\}$, then we say that Γ is equivalent to S^n, or $\Gamma \cong S^n$. If J represents an orbit of variables and $\texttt{Proj}_J(\Gamma)$ consists of all permutations of the variables, then $\texttt{Proj}_J(\Gamma) \cong S^{|J|}$.

Example 2.1 *Permutations are commonly written in* cycle notation*. The expression* $(a_1,\ a_2,\ \ldots a_k)$ *denotes a cycle which sends* a_i *to* a_{i+1} *for* $i = 1,\ \ldots,\ k-1$ *and sends* a_k *to* a_1*. Permutations can be written as a product of cycles. For example, the expression* $(a_1,\ a_2)(a_3)$ *denotes a permutation which sends* a_1 *to* a_2*,* a_2 *to* a_1*, and* a_3 *to itself. We omit 1-element cycles for clarity.*

Let $\Gamma \subset S^5$ *be the permutation group containing the following permutations:*

$$\begin{aligned}\pi_0 &= e\\ \pi_1 &= (1,2)\\ \pi_2 &= (3,4,5)\\ \pi_3 &= (1,2)(3,4,5)\end{aligned}$$

Note that all the permutations in Γ *can be generated using only* π_3 *(for example* $\pi_0 = \pi_3^6$*). The orbital partition of* Γ *is* $\{1,\ 2\}$ *and* $\{3,\ 4,\ 5\}$*. If we projected* Γ *onto the set* $\{1,\ 2\}$ *we would have the following permutations*

$$\begin{aligned}\pi_0' &= e\\ \pi_1' &= (1,2)\\ \pi_2' &= e\\ \pi_3' &= (1,2).\end{aligned}$$

This projection contains all permutations of the set $\{1,2\}$*, so* $\texttt{Proj}_{\{1,\ 2\}}(\Gamma) \cong S^2$*. Projecting onto the set* $\{3,\ 4,\ 5\}$ *gives the permutations*

$$\begin{aligned}\pi_0'' &= e\\ \pi_1'' &= e\\ \pi_2'' &= (3,4,5)\\ \pi_3'' &= (3,4,5).\end{aligned}$$

This projection does not contain all permutations of the set $\{3,\ 4,\ 5\}$*, so* $\texttt{Proj}_{\{3,\ 4,\ 5\}}(\Gamma) \ncong S^3$*.*

2.2 Orbital Branching

The methods discussed in this section apply to general MILP problems. A subproblem a in the branch-and-bound tree is defined by two sets: the set of variables fixed to zero, F_0^a, and the set of variables fixed to one, F_1^a. We let $\mathcal{F}^a$ denote the feasible set of a and $\mathcal{G}^a$ its symmetry group. As variables are fixed, $\mathcal{G}^a$ changes and needs to be recomputed. For example, suppose x_i and x_j share an orbit at the root node (there was a permutation in $\mathcal{G}$ that mapped i to j). If $x_i \in F_0^a$ and $x_j \in F_1^a$, then for any $x \in \mathcal{F}^a$ and symmetry π mapping i to j, $\pi(x)_j = 0$, meaning $\pi(x) \notin \mathcal{F}^a$, and thus π is not in $\mathcal{G}^a$.

Orbital branching works as follows. Let $\mathcal{O}^i = \{x_{i_1},\ x_{i_2}, \ldots,\ x_{i_n}\}$ be an orbit of $\mathcal{G}^a$. Rather than branching on the disjunction

$$x_{i_1} = 1 \ \vee \ x_{i_1} = 0, \tag{4}$$

one uses the branching disjunction

$$\sum_{j=1}^{i_n} x_{i_j} \geq 1 \ \vee \ \sum_{j=1}^{i_n} x_{i_j} = 0. \tag{5}$$

Note that the right-side disjunction in (5) fixes all the variables involved to zero. While branching decisions typically use disjunctions of type (4), the disjunction (5) is a valid branching decision. Symmetry is exploited by the following observation regarding the left disjunction in (5): since all the variables in $\mathcal{O}^i$ are symmetric, and at least one of them must be equal to one, any variable in $\mathcal{O}^i$ can be arbitrarily chosen to be one, leading to the symmetry strengthened disjunction

$$x_{i_1} \geq 1 \ \vee \ \sum_{j=1}^{i_n} x_{i_j} = 0. \tag{6}$$

It is easy to see that (6) is a valid disjunction by considering the following. Because the variables are symmetric, the subproblem generated by fixing x_{i_1} to one will be identical to the subproblem generated by fixing x_{i_2} to one. Therefore, if there is an optimal solution with x_{i_1} equal to one, then there will be an optimal solution with $x_{i_2} = 1$. If there is an optimal solution that is feasible at a, there will be an optimal solution feasible in one of the subproblems of a.

The strength in orbital branching is that a total of $|\mathcal{O}^i|+1$ many variables are fixed by the branching, not the two that traditional branching fixes. Because of this, the larger the orbit, the more likely the branching decision will be strong. Stronger branching decisions will improve the lower bound faster, leading to shorter solution times.

Example 2.2 *This example shows the effectiveness of orbital branching on the Jeroslow problem* (1)*. Figure 1 shows the branch-and-bound tree for the instance where* $n = 8$*. A pure branch-and-bound approach requires exponentially many (in* n*) branches, whereas orbital branching requires only a linear number of branches. All nodes except* K *are pruned by infeasibility. The LP solution at node* K *is integer (and optimal).*

3 Modified Orbital Branching

The branching tree in Example 2.2 is very lopsided. While the unbalanced nature for this example is extreme, it is likely that the branch-and-bound tree will be unbalanced for highly symmetric problems, as the right branch is typically much stronger (fixing $|O^i|$ variables to zero) than the left branch (fixing one variable to one). This may be problematic because the symmetry group of the left subproblem is typically much smaller than that of the current node. Thus subsequent branching disjunctions in the left subproblem are likely to be weaker as the smaller symmetry groups lead to smaller orbits. The symmetry group of

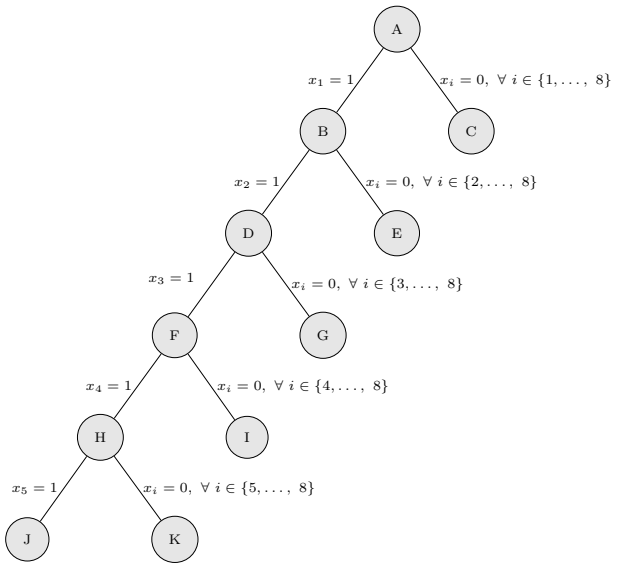

Figure 1: B&B tree for the Jeroslow problem with orbital branching

the right child does not change at all, leading to more powerful branches, but this only helps when the right node is not pruned.

In some situations, the branching decision can be modified to create a more balanced branch-and-bound tree. Suppose we branch on orbit $\mathcal{O}^i$ at subproblem a with symmetry group $\mathcal{G}^a$. If $\texttt{proj}_{\mathcal{O}^i}(\mathcal{G}^a)$ is equivalent to $S^{|\mathcal{O}^i|}$, then the branching disjunction

$$\sum_{j=1}^{i_n} x_{i_j} \geq b' \ \vee \ \sum_{j=1}^{i_n} x_{i_j} \leq b' - 1, \tag{7}$$

for some $b' \in \mathbb{Z}^+$, can be replaced by

$$x_{i_j} = 1 \ \forall\ j \in \{1, \ldots,\ b'\} \ \vee \ x_{i_j} = 0 \ \forall\ j \in \{b',\ b'+1,\ \ldots,\ |O^i|\}. \tag{8}$$

While different values of b' can be chosen for (7), the choice $b' = \lceil \sum_{i \in \mathcal{O}} x_i^* \rceil$ is natural.

Theorem 3.1 *Let $a = (F_0^a, F_1^a)$ be a node in the branch and bound tree with feasible region $\mathcal{F}^a$. Suppose orbit $\mathcal{O}^i$ with $\texttt{proj}_{\mathcal{O}^i}(\mathcal{G}^a) \cong S^{|\mathcal{O}^i|}$ was chosen for branching. Child l is formed by fixing $x_{i_j} = 1\ \forall\ j \in \{1, \ldots,\ b'\}$ and child r is formed by fixing $x_{i_j} = 0\ \forall\ j \in \{b',\ b'+1,\ \ldots,\ |O^i|\}$. For any optimal x^* in $\mathcal{F}^a$, there exists a $\pi \in \mathcal{G}^a$ with $\pi(x^*)$ contained in either $\mathcal{F}^l$ or $\mathcal{F}^r$.*

Proof: Assume that $\sum_{i \in \mathcal{O}^i} x^* \geq b'$. We construct $\pi \in \mathcal{G}^a$ such that $\pi(x^*) \in \mathcal{F}^l$.

Let $I_{\mathcal{O}^i} = \{i_j \in \mathcal{O}_i | x^*_{i_j} = 1\}$, $\mathcal{I} = \{j | j \notin S_{\mathcal{O}^i} \text{ and } 0 \leq j \leq b'\}$, and $\mathcal{J} = \{j | j \in S_{\mathcal{O}^i} \text{ and } b' < j\}$. By our assumption, we have $|\mathcal{I}| \leq |\mathcal{J}|$.

Let π initially be the identity permutation. For every $j \in \mathcal{I}$, choose a unique element $j' \in \mathcal{J}$. Because $\texttt{proj}_{\mathcal{O}^i}(\mathcal{G}^a) \cong S^{|\mathcal{O}^i|}$, there exists a $\pi_{j,j'}$ mapping j to j' (and vice versa) that leaves the remaining elements on $\mathcal{O}_i$ unchanged. Amend π such that $\pi \stackrel{\text{def}}{=} \pi \circ \pi_{j,j'}$.

By the definition of a group, the resulting π will be an element of $\mathcal{G}^a$ (as it is a composition of elements of $\mathcal{G}^a$). As such, $\pi(x^*)$ is in $\mathcal{F}^a$. Moreover, π maps variables that take the value of "1" in x^* to elements in $\mathcal{I}$ while leaving the variables that already take the value of "1" unchanged. As a result, $\pi(x^*)$ satisfies the constraint $\sum_{i \in O^i} x \geq b'$, and is in $\mathcal{F}^l$.

The case of $\pi(x^*)$ in $\mathcal{F}^r$ is proved similarly. □

Example 3.1 *Let us apply modified orbital branching to the Jeroslow problem (1). First, note that the orbit* $O^1 = \{1, \ldots, 8\}$ *is such that* $\texttt{proj}_{O^1}(\mathcal{G}) \cong S^8$*, as any permutation of the first 8 variables is a symmetry of the problem. While there are many basic optimal solutions to the LP relaxation, it is easy to verify that all optimal solutions to the LP at the root node have* $\sum_{i=1}^{8} x_i = 4.5$*, hence we use* $b' = 5$ *and branch on* $\mathcal{O}^1$*.*

The resulting branch-and-bound tree is shown in Figure 2. While node B is infeasible, the LP relaxation at C has $x_1 = x_2 = x_3 = x_4 = x_9 = 1$*, an integer (and optimal) solution. Note that for this particular problem, using modified orbital branching is equivalent to adding the symmetry breaking constraints* $x_1 \geq x_2 \geq \ldots \geq x_8$*.*

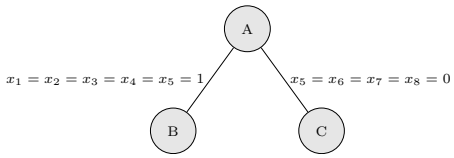

Figure 2: B&B tree for the Jeroslow problem with modified orbital branching

4 Orbital Branching and Orbitopes

Consider the set of all feasible $m \times n$ 0/1 matrices, where the symmetry acting on the variables consists of all permutations of the columns. A classic example of a problem with this structure is graph coloring. One type of symmetry found in graph coloring arises from permuting colors. If entry i, j equals 1 if and only if i is assigned color j then permuting colors corresponds to permuting two columns of the solution matrix. A common technique used to remove equivalent solutions to the graph coloring problem (as well as other problems) is to restrict the feasible region to matrices with lexicographically decreasing columns. This idea can be generalized to other problems with this structure.

The convex hull of all $m \times n$ 0/1 matrices with lexicographically decreasing columns is called a *full orbitope*. A *packing orbitope* is the convex hull of all matrices with lexicographically decreasing columns with *at most* one 1 per row,

and the *partitioning orbitope* is the set with *exactly* one 1 per row. Complete linear descriptions of packing and partitioning orbitopes are given in [11]. While these descriptions require exponentially many inequalities, a polynomial time algorithm can be used to test if a solution of the LP relaxations violates any of the constraints [10]. A complete description of full orbitopes is not known, though an extended formulation is given in [9].

In this section, we show how modified orbital branching can be used to restrict the branch-and-bound search to solutions in the full orbitope. Moreover, we show that modified orbital branching can be used to generate all elements of a full orbitope in polynomial time for a fixed m.

Let $\mathcal{P}(m,n) = \{x \in \mathbb{R}^{m \times n} | x_{i,j} \in \{0,\ 1\}\ \forall i \in \{1,\ \ldots, m\}$ and $j \in \{1,\ \ldots,\ n\}\}$. Let $\mathcal{P}_O(m,n) \subset \mathcal{P}(m,n)$ denote the set of matrices with lexicographically decreasing columns. Suppose the MILP has of the form

$$\min\{\sum_i \sum_j c_{ij} x_{ij} |\ \sum_i \sum_j a^k_{ij} x_{i,j} \geq b^k \quad x \in \mathcal{P}(m,n)\}, \tag{9}$$

and that the symmetry group contains all column permutations of the x variables. As a result of the symmetry, we can restrict our search to be over $\mathcal{P}_O(m,n)$.

While variable orbits can be computed extremely fast if the symmetry group is known, there are no known polynomial-time algorithms to compute the symmetry group of a subproblem in the branch-and-bound tree for a general integer programming problem. However, because of the special structure of orbitopes, computing column symmetries (and thus orbits) can be done in linear time with respect to the number of variables. Permutations that permute columns c_j and c'_j are in the symmetry group of the subproblem $a = (F^a_0, F^a_1)$ iff $x_{i,j} \in F^a_0 \leftrightarrow x_{i,j'} \in F^a_0$ and $x_{i,j} \in F^a_1 \leftrightarrow x_{i,j'} \in F^a_1$ for all i in $\{1,\ \ldots, m\}$.

To the best of our knowledge, the only method known to remove all isomorphic solutions from a full orbitope's symmetry group is to use isomorphism pruning. Unfortunately, however, isomorphism pruning requires a computationally expensive membership test for every subproblem in the branch-and-bound tree. No polynomial time algorithms are known for this test. Even though orbital branching removes a significant proportion of isomorphic solutions from the feasible region, it is still possible for some symmetry to remain unexploited. The extent to which symmetry remains depends on the branching decisions made. With an appropriate branching rule, however, it is possible to remove all symmetry. One such rule is the *minimum row-index branching rule*. For this branching rule, $x_{i,j}$ is a branching candidate iff variables $x_{i',j'}$ have already been fixed for all $i' < i$ and all $j' \in \{1, \ldots, n\}$. In other words, variables in row i are only eligible for branching if all variables in rows less than i have already been fixed. While this may seem highly restrictive, it can be exploited in order to provide a lot allow for greater flexibility.

Suppose our goal is to generate all solutions to (9), i.e., nodes are only pruned by feasibility. Let $\mathcal{C}$ be the set of solutions found (corresponding to leaf nodes of depth nm in the full branch-and-bound tree) when using modified

orbital branching with a minimum row-index branching rule. Theorem 4.1 shows that the minimum row-index branching rule combined with modified orbital branching can be used to enumerate all feasible solutions to $\mathcal{F} \cap \mathcal{P}_O(m,n)$.

Theorem 4.1 $\mathcal{C} = \mathcal{F} \cap \mathcal{P}_O(m,n)$

Proof: The set $\mathcal{F} \cap \mathcal{P}_O(m,n)$ contains no isomorphic solutions. By Theorem 3.1 we know that $\mathcal{C}$ contains at least one representative from each set of isomorphic solutions, so $|\mathcal{C}| \geq |\mathcal{F} \cap \mathcal{P}_O(m,n)|$. We need only to show that $\mathcal{C} \subseteq \mathcal{F} \cap \mathcal{P}_O(m,n)$ in order to prove the above theorem.

Suppose $\mathcal{C} \not\subseteq \mathcal{F} \cap \mathcal{P}_O(m,n)$. Let x be such that x is in $\mathcal{C}$ but not in $\mathcal{F} \cap \mathcal{P}_O(m,n)$. As x is not in $\mathcal{P}_O(m,n)$, it must contain two adjacent columns, j and $j+1$, with column $j+1$ lexicographically larger than column j. Let i be the first row where columns j and $j+1$ differ. We have $x_{i,j} = 0$ and $x_{i,j+1} = 1$.

By definition of $\mathcal{C}$, there is a node of depth mn in the branch-and-bound tree representing solution x. Let subproblem a be the earliest ancestor of this subproblem where either $x_{i,j}$ or $x_{i,j+1}$ was fixed by branching. At node a, the fixed variables in column j are identical to those in column $j+1$. If $x_{i,j}$ was fixed to zero by branching, then $x_{i,j+1}$ must be in the orbit branched upon. In that case, the branching disjunction that fixed $x_{i,j}$ to zero would have also fixed $x_{i,j+1}$ to zero (since $j < j+1$). Similarly, if $x_{i,j+1}$ was fixed to one by branching, then $x_{i,j}$ must have been fixed to one. This branching disjunction would have removed x from both children's feasible region, so no such x can exist. □

Theorem 4.1 can be extended to MILPs where only a subset of the variables can be expressed as 0/1 matrices.

4.1 The Complexity of Enumerating Elements of an Orbitope

In this section we show that for fixed m, there are polynomially many solutions in $\mathcal{P}_O(m,n)$. As a consequence of this, modified orbital braching can enumerate all feasible solutions in polynomial time (for a fixed m).

Theorem 4.2 *For fixed m, the number of solutions in $\mathcal{P}_O(m,n)$ grows polynomially in n.*

Proof: We use a combinatorial argument. Let $\mathcal{B}$ be the (numbered) collection of 0/1 m-vectors. We represent each feasible solution in $\mathcal{P}_O(m,n)$ as a collection of n of these m-vectors (with duplication). Note that $|\mathcal{B}| = 2^m$. Using a "stars-and-bars" argument, we know that there are $\binom{n+|\mathcal{B}|}{|\mathcal{B}|}$ ways to choose n elements from a set of $\mathcal{B}$ elements with replacements. Since $\binom{n+|\mathcal{B}|}{|\mathcal{B}|} = O(n^{|\mathcal{B}|})$ for large n, the result follows. □

Next, we need to show that the branch-and-bound tree grows polynomially as n increases. This is easy to see by observing that at any subproblem in the branch-and-bound tree, a feasible solution to $\mathcal{P}_O(m,n)$ can be found by fixing

all free variables to zero. As a result, there can be no more than $\binom{n+|\mathcal{B}|}{|\mathcal{B}|}$ many nodes at any depth in the tree. The depth of the branch-and-bound tree is bounded above by $mn+1$, so there can be at most $mn\binom{n+|\mathcal{B}|}{|\mathcal{B}|}$ many nodes. At each subproblem, only the orbits and the minimum row-index need to be computed. These can also be done in polynomial time.

4.2 Strength of LP Relaxation

Theorem 4.1 shows that modified orbital branching *eventually* removes isomorphic solutions from the feasible solutions. One concern is that the LP relaxations might be weaker than if the isomorphic solutions were removed at the root node and weaker relaxations might lead to larger branch-and-bound trees. Fortunately, though, this is not the case for full orbitopes. Friedman [6] shows that all isomorphic solutions can be removed from the LP relaxation's feasible region by adding the constraints

$$\sum_{k=1}^{m} 2^{m+1-k} x_{k,j} \geq \sum_{k=1}^{m} 2^{m+1-k} x_{k,j+1} \ \forall j \in \{1, \ldots, \ n-1\}. \tag{10}$$

Note that these constraints force the columns of x to be lexicographically decreasing. It is clearly not practical to add these constraints to any problem of reasonable size. However, even if it were, these constraints would not improve the LP relaxation at the root node.

Theorem 4.3 *For any subproblem of* (9) *formed by using modified orbital branching with a minimum row-index branching rule, the solution to the LP relaxation does not change when constraints* (10) *are added to the formulation.*

Proof: A well known result from [7] states that for a linear program with symmetry group $\mathcal{G}$, there exists an optimal solution with $x_i = x_{\pi(i)}$ for all $\pi \in \mathcal{G}$. Because all column permutations are symmetries, there will always be optimal LP solution at the root node with $x_{i,j} = x_{i+1,j}$ for all appropriate i and j. This solution is not removed by the inequalities described in (10). □

Note, however, that constraints (10) may be strengthened by considering the integrality of the variables in order to generate tighter cuts. Theorem 4.3 does not apply to these integrality-based constraints.

4.3 Relaxing the Branching Rule

Unfortunately, Theorem 4.1 requires the minimum row-index branching strategy. Example 4.1 illustrates how isomorphic solutions can remain if the minimum row-index branching rule is not used.

Example 4.1 *Figure 3 shows a branching tree for a problem where x is a 2×2 0/1 matrix whose symmetry contains the permutation of the columns of x. The*

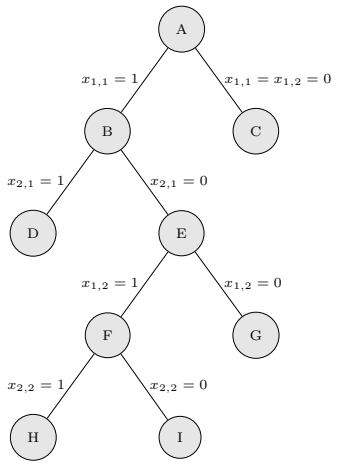

Figure 3: Symmetry remaining after modified orbital branching

solution represented by subproblem H is

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \tag{11}$$

which does not have lexicographically decreasing columns. This solution is equivalent to

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \tag{12}$$

which is feasible in node D.

Why does modified orbital branching fail to prune the solution at node H from the feasible region? The problem starts at node B. There is no symmetry in node B, as the first column of x contains one fixed variable and the second column does not. However, symmetry is present in $\mathcal{F}_B \cup \{x \in \mathcal{F} | x_{1,2} = 1\}$. Fixing $x_{1,2}$ to one at node B (as is done with the minimum row-index branching rule) would reincorporate the column symmetry into the subproblem. In this example, however, this is not done. Further branching also does not reincorporate this symmetry, and as a result, the search explores more solutions than is necessary.

Figure 4 shows the branch-and-bound tree when the minimum row-index branching rule is used. Note that the column symmetry is reincorporated into subproblem D', leading to a nontrivial orbital branch. As a result of this branch, solution (11) *is removed from the search space.*

Enforcing a fixed branching rule does hinder the performance of the solver because different strategies to choose branching variables can have a large impact on the time required to solve the problem. Fortunately, a few tricks can be implemented to add more flexibility in branching. Similar to the strategy used

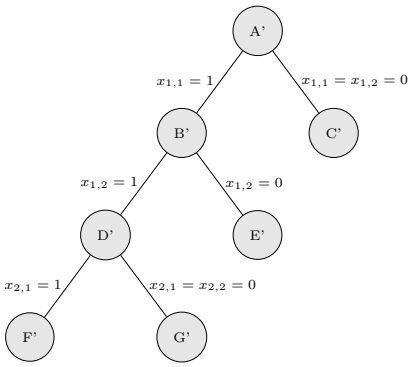

Figure 4: No symmetry remaining with minimum row-index rule

in [14], row indices can be permuted to favour branching. For example, suppose the current subproblems contained solutions of the type

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \\ x_{3,1} & x_{3,2} \\ x_{4,1} & x_{4,2} \end{pmatrix}. \tag{13}$$

The minimum row-index branching rule requires that either $x_{3,1}$ or $x_{3,2}$ be chosen for branching. However, by reindexing the rows, we can create a new subproblem with solutions of the type

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \\ x'_{3,1} = x_{4,1} & x'_{3,2} = x_{4,2} \\ x'_{4,1} = x_{3,1} & x'_{4,2} = x_{3,2} \end{pmatrix} \tag{14}$$

and now either $x'_{3,1}$ or $x'_{3,2}$ (representing $x_{4,1}$ and $x_{4,2}$) can be chosen for branching. Note that a row cannot be reindexed after any variable in that row has been fixed by branching. Similar to isomorphism pruning [15], this leads to a reordering of the rows for each subproblem in the branch-and-bound tree. This reordering can be kept track of by using a *rank vector*. The rank vector $R^a \in \mathbb{R}^m$ for subproblem a denotes the order in which rows were branched on. $R^a[i] = r < m + 1$ designates that row i was the rth row branched on, whereas $R^a[i] = m + 1$ indicates that no variable in row i has been fixed by branching. The *minimum row-rank* branching rule states that if there is a row with rank less then $m + 1$ that contains a free variable, a variable from that row must be chosen for branching (only one such row will exist). The resulting solutions using the minimum row-rank branching rule may not have lexicographical solutions, but the space explored will not contain any isomorphic solutions. A corollary to Theorem 4.1 is:

Corollary 4.1 $|\mathcal{C}| = |\mathcal{F} \cap \mathcal{P}_O(m, n)|$.

Even more flexibility in branching can be obtained by recognizing when symmetry will never be reincorporated. This is the case in the partial solution shown in (14). Independent of the unknown variables, column two will always be lexicographically smaller than column one. This is easily seen by recognizing that the first difference between the solutions in the columns are at row two, where column one has a "1" and column two has a "0". In this case, variables can be branched on in any order and still guarantee that only nonisomorphic solutions are explored. We need to test if and when this is the case.

Suppose we wish to branch on orbit $O_{i,j} = \{x_{i,j},\ x_{i,j+1},\ \ldots,\ x_{i,j+k}\}$ at subproblem a with rank vector R^a. We say that $O_{i,j}$ is *left closed* if $j = 1$ or column j (reindexed by R^a) is guaranteed to be lexicographically smaller than column $j-1$.

Left closure is easily determined by finding the first (after reindexing) row i' with $x_{i',j-1}$ fixed to a different value than $x_{i',j}$. If $x_{i',j-1} \in F_1^a$ and $x_{i',j} \in \mathcal{F}_0^a$, then $O_{i,j}$ is left closed. Similarly, $O_{i,j}$ is right closed if either $j = n_k$ or $O_{i,j+k+1}$ is left closed (the jth column of the matrix will always be lexicographically larger than the j+k+1st column). Orbit $O_{i,j}$ is *closed* if it is both left closed and right closed.

Example 4.2 *Consider the subproblem with the following fixed variables*

$$\left(\begin{array}{c|c|c|c|c|c} 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & x_{3,4} & 0 & 0 \\ 0 & x_{4,2} & x_{4,3} & x_{4,4} & x_{4,5} & 0 \\ 1 & x_{5,2} & x_{5,3} & x_{5,4} & x_{5,5} & x_{5,6} \\ x_{6,1} & x_{6,2} & x_{6,3} & x_{6,4} & x_{6,5} & x_{6,6} \\ x_{7,1} & x_{7,2} & x_{7,3} & x_{7,4} & x_{7,5} & x_{7,6} \end{array}\right). \tag{15}$$

The symmetries in this subproblem permute columns two and three as well as five and six. Let O_1, be the orbit representing column 1, O_2 be the orbit representing columns 2 and 3, O_4 be the orbit representing column 4, and O_5 be the orbit representing column 5 and O_6 be the orbit representing column 6.

O_1 is trivially left closed (because it contains the leftmost column). It is also right closed because column 2 can never be lexicographically larger than column 1 (since $x_{2,1} = 1$ and $x_{2,2} = 0$).O_2 is left closed (for the same reasons O_1 is right closed), but not right closed. This is because $x_{3,3} = 1$ but $x_{3,4}$ is currently free. If $x_{3,4}$ were fixed to one, then column four would join the column orbit O_2. O_4 is neither left nor right closed. O_5 is right closed, implying that O_6 is right closed. Because O_6 contains the right-most column, it is also right closed (making it closed).

The minimum row-index branching rule would require that $x_{3,3}$ be chosen for branching.

If orbit $O_{i,j}$ is closed at subproblem a, then subproblem a can be reformulated to account for the symmetry removed as a result of the branching decisions. Instead of one 0/1 matrix representing the variables, three 0/1 matrices can

used. Let $M(1, j-1)$ be the $m \times (j-1)$ matrix representing the first $j-1$ columns, $M(j, j+k)$ be the $m \times (j+k)$ matrix representing the the columns j through $i+k$, and $M(j+k+1, n_k)$ be the $m \times (n_k - j - k - 1)$ matrix representing the last $n_k - j - k - 1$ columns of the original matrix. By definition of "closed", every column in $M(1, j-1)$ must be lexicographically larger (after reindexing) than any column in both $M(j, j+k)$ and $M(j+k+1, n_k)$. Similarly, every column in $M(j, j+k)$ must be lexicographically larger than any column in $M(j+k+1, n_k)$. Because of this reformulation, any row of $M(i, i+j)$ can be branched on using modified orbital branching (while reindexing the rows if necessary). Thus, row i' can be chosen for branching even if there is a row with smaller rank containing a free variable in either $M(1, j-1)$ or $M(j+k+1, n_k)$.

Example 4.3 *Referring back to* (15) *in the previous example, only columns* O_1 *and* O_6 *are closed. As a result, we can partition the 0/1 matrix into 3 different matrices:*

$$M(1,1) = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ x_{6,1} \\ x_{7,1} \end{pmatrix}, M(2,5) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & x_{3,3} & 0 \\ x_{4,2} & x_{4,3} & x_{4,4} & x_{4,5} \\ x_{5,2} & x_{5,3} & x_{5,4} & x_{5,5} \\ x_{6,2} & x_{6,3} & x_{6,4} & x_{6,5} \\ x_{7,2} & x_{7,3} & x_{7,4} & x_{7,5} \end{pmatrix}, \text{ and } M(6,6) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ x_{5,6} \\ x_{6,6} \\ x_{7,6} \end{pmatrix}.$$

Now there are several options for branching variables. In $M(1,1)$*, there is no row containing a fixed variable and a free variable, so any free variable can be chosen for branching (and updating the rank vector accordingly). This is also the case for variables in* $M(6,6)$*. In* $M(2,5)$*, however, there is a row containing fixed and free variables. As a result,* $x_{3,3}$ *is the only variable in* $M(2,5)$ *that can be branched on. Note, however, that fixing (by branching)* $x_{3,3}$ *to either 0 or 1 will add symmetry, as column 4 will join* O_2 *in the 1 case and* O_5 *in the 0 case.*

Algorithm 1 summarizes how to choose the appropriate branching orbit for orbitopes.

Recall that the packing (partioning) orbitope is the collection of all lexicographically minimal 0/1 matrices with at most (exactly) one 1 per column. It is easy to see that every orbit in every subproblem will be closed, meaning that isomorphic solutions can be removed from the search without requiring any branching restrictions.

4.4 Practical Implementation

Algorithm 1 describes how to use modified orbital branching to ensure that the branching process explores only non-isomorphic solutions. One implementation difficulty, however, is the use of the rank vector. The rank vector describes the branching decisions made on the path from the root node to the current problem in the branch-and-bound tree. Some MILP solvers do not keep track of what

Algorithm 1 Selecting Branching Orbit at Subproblem $a = (F_0^a, F_1^a)$

INPUT: F_1^a, F_0^a, R^a, and candidate branching orbit $O_{i,j}$.
OUTPUT: Branchable orbit
Is $O_{i,j}$ is left closed?
  If Yes
    Is $O_{i,j}$ is right closed?
      If Yes
        Branch on $O_{i,j}$
      If No
        Let r be the smallest ranked index where only one of $x_{r-1,j}$ and $x_{r-1,j+k+1}$ is in N^a.
        If $x_{r,j} \in F_1^a$
          Branch on $O_{r,j+k+1}$
        If $x_{r,j+k+1} \in F_0^a$
          Branch on $O_{r,j}$
  If No
    Let r be the smallest ranked index where only one of $x_{r-1,j-1}$ and $x_{r-1,j}$ is in N^a
      If $x_{r,j-1} \in F_1^a$
        Branch on $O_{r,j}$
      If $x_{r,j} \in F_0^a$
        Branch on $O_{r,j-1}$

the actual tree looks like, and storing the rank vector for each subproblem may consume a lot of memory. Fortunately, the rank vector is not necessary in practice.

The rank vector is used to determine two things. First, it is needed to determine if a given orbit is closed. Suppose that all variable fixings occur only through branching (no reduced cost fixing or other logical implications). Using Algorithm 1, it is easy to determine if a given orbit $O_{i,j}$ is closed (both left and right) without using the rank vector.

Theorem 4.4 *If there is a row i' such that $x_{i',j-1} \in F_1^a$ and $x_{i',j} \in F_0^a$, then $O_{i,j}$ must be left closed. Similarly, if there is a row i' such that $x_{i',j} \in F_1^a$ and $x_{i',j+k+1} \in F_0^a$, then $O_{i,j}$ must be right closed.*

Proof: Let i' be such that $x_{i',j-1} \in F_1^a$ and $x_{i',j} \in F_0^a$. Aiming at a contradiction, assume that $O_{i,j}$ is not left closed. Because $O_{i,j}$ is not left closed, there must be a row r with rank smaller than i' with either

- $x_{r,j-1}, x_{r,j} \in N^a$,
- $x_{r,j-1} \in \mathcal{F}_1^a$ and $x_{r,j} \in N^a$,
- $x_{r,j-1} N^a$ and $x_{r,j} \in F_0^a$.

In any of these cases, there is a free variable in a column with rank less then i', meaning that neither $x_{i',j-1}$ nor $x_{i',j}$ could have been fixed by branching.

If there is a row i' such that $x_{i',j} \in F_1^a$ and $x_{i',j+k+1} \in F_0^a$, then $O_{i,j}$ must be right closed, as $O_{i,j+k+1}$ is left closed. □

The rank vector is also used to find a branchable orbit if $O_{i,j}$ is not closed. In this case, another branchable orbit must be found. Let us first consider the case where $O_{i,j}$ is neither left nor right closed. Supposing that all variable fixings occur only through branching, Algorthm 1 will return the same branchable orbit if the term "let r be the smallest ranked index" is replaced with "let r be *any* index".

Theorem 4.5 *Algorithm 1 will return the same branchable orbit if the term "let r be the smallest ranked index" is replaced with "let r be any index".*

Proof: If $O_{i,j}$ is not left closed, then there can be only one row where only one of $x_{r-1,j-1}$ and $x_{r-1,j}$ are in N^a. This is a direct consequence of the minimum row-index branching. □

Theorems 4.4 and 4.5 rely on the fact that variables are only fixed by branching. In reality, a commercial solver will attempt to fix variables by a variety of methods, such as reduced-cost fixing or strong-branch fixing. These fixing make it difficult to determine the appropriate branching orbit. For example, two consecutive columns may have many rows where they take different values. Fortunately, though, the structure of the modified branching algorithm can be studied in order to return good branchable orbits. Recall from the algorithm that if $O(i,j)$ is not left closed, then the row index is chosen by looking for a

row with either $x_{i,j-1} \in F_1^a$ and $x_{i,j} \in N^a$ or $x_{i,j-1} \in N^a$ and $x_{i,j} \in F_0^a$. If no such row exists, then additional variables can be fixed in a way that makes $O(i,j)$ a larger orbit. If there are two or more such rows, then any row can be arbitrarily chosen.

5 Orbital Branching and the Unit Commitment Problem

To demonstrate the effectiveness of modified orbital branching, we apply it to the unit commitment (UC) problem. The UC problem is an important problem in the power systems community. Given a set of power generators and set of electricity demands, the UC problem minimizes the total production cost of the power generators subject to the constraints that

1. the demand is met, and
2. the generators operate within their physical limits, i.e.,
   (a) the power output level of a generator may not change too rapidly (ramping constraints), and
   (b) when a generator is turned on (off), it must stay on (off) for a minimum amount of time (minimum up / downtime constraints).

Different MILP formulations for the UC problem have been proposed, see e.g. [1, 3, 2].

The variables in UC can be expressed as full orbitopes. Real-life instances of the UC problem are typically very large and difficult. UC problems are particularly difficult when there are many identical generators, resulting in many symmetries.

The MILP formulations were initially solved using Lagrangian relaxation. However, even without branch-and-bound, symmetry still causes numerical difficulties in Lagrangian relaxation methods. The paper [20] attempted to address this issue by developing a surrogate subgradient method to update second level prices. At present, most system operators use MILP solvers whose performance may benefit from effective symmetry-breaking methods.

One way to remove symmetry from the UC problem's formulation is to aggregate all identical generators into a single generator. This is typically done for combined cycle plants that can have two or three identical combustion turbines. While this might be effective in reducing the number of variables, aggregating generator variables may be very difficult, and some of the physical requirements may be difficult to enforce. If the UC solution only returned the number of generators operating at each time period as well as the total power produced by those generators, it might be difficult to determine which generators produce how much power at each time period. A comparison between aggregating combined cycle generators and modeling them as individual units can be found in [12].

We assume demand is known a priori. Some physical limits of the generators include minimum up and down time and ramping rates. Efforts to improve MILP modeling include improving the linear relaxations of the nonlinear objective function [4, 5], examining the minimum up/downtime polytope [18], and tightening ramping constraints [16].

We use the three-binaries-per-generator-hour formulation based on [1]. This model contains three sets of binary variables at every time period: those representing the on/off status of each generator at the time period, those representing if each generator is started up in the time period, and those representing if each generator is shut down in the time period.

Since the startup/shutdown status can be easily determined if the on/off status is known, in our discussion we focus only on those variables indicating if the generator is on or off. This is done for notational convenience. It is common in the power systems literature to relax the integrality constraints of the startup and shutdown variables.

Suppose that there are K different classes of generators, where n_k denotes the number of generators of type k. We let x_{it}^k be the on/off status of the ith generator of type k at time t. Because all generators of type k are identical, the set $\mathcal{O}(k,t) = \{x_{1k}^t,\ x_{2k}^t, \ldots,\ x_{n_k,t}^k\}$ is an orbit with respect to $\mathcal{G}$ for all t in T. Moreover, $\texttt{Proj}_{\mathcal{O}(k,t)}(\mathcal{G}) \cong S^{n_k}$, so modified orbital branching can be applied when orbit $\mathcal{O}(k,t)$ is chosen for branching.

We randomly generated instances with 45 to 75 generators using eight different types of generators and solved them to 0.01% optimality. These generators are based on data used in [16]. The following algorithms were tested:

- **Default CPLEX:** CPLEX's default algorithm. This includes methods such as dynamic search as well as multithreading.
- **Branch & Cut:** CPLEX with advanced features turned off (mimicking what happens when callbacks are used); CPLEX's symmetry-breaking procedure is used.
- **OB:** Original orbital branching implemented using callbacks.
- **Modified OB:** Modified orbital branching from Section 3 implemented using callbacks.
- **Modified OB Lex:** Modified orbital branching implemented using callbacks with the (relaxed) branching rule to guarantee only non-isomorphic solutions are explored.

All versions of orbital branching were implemented in CPLEX 12 using the branch callback feature. Branching decisions are determined by CPLEX. After CPLEX chooses a branching variable, modified orbital branching augments it by searching for the appropriate orbit. In the case of modified orbital branching, the orbit is used as an input to Algorithm 1, and not necessarily the orbit used to branch. One would expect that incorporating the orbit information into determining the branching variable would further improve solution times.

Unfortunately, using callback functions disables other CPLEX features, notably dynamic search. For this reason, in addition to comparing classical orbital branching with the versions of modified orbital branching, we also give results based on CPLEX's default setting (dynamic search plus additional features) and CPLEX with features disabled (using traditional branch-and-cut).

The computational results are reported in Table 1. Instances not solved within two hours are denoted by "-".

Instance Number	Number of Generators	CPLEX Only									
		Default CPLEX		Branch & Cut		OB		Modified OB		Modified OB Lex	
		Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes
1	45	120.13	5896	81.76	654	1123.83	28386	243.13	2061	48.68	140
2	45	349.45	39838	-	-	-	125999	4528.86	60283	6497.75	70300
3	45	190.48	4845	344.63	4994	952.76	22374	196.57	2000	814.72	10480
4	48	165.69	6044	-	-	3091.62	55149	287.96	1922	294.34	2392
5	49	334.96	5584	5750.05	101690	1750.61	30076	551.48	5147	218.95	1160
6	51	706.57	47307	-	-	-	95883	5379.3	57893	3960.36	31170
7	52	128.67	4608	341.21	5530	4002.4	67290	244.89	2400	733.30	6870
8	52	64.99	2409	59.15	140	12.55	0	12.41	0	12.72	0
9	56	329.68	4959	1251.25	20100	547.66	7300	64.77	72	143.18	360
10	56	1129.28	46383	-	-	6747.10	105600	2245.25	21692	-	-
11	56	290.25	5412	513.64	7460	1013.57	16390	351.13	2992	125.56	351
12	58	156.38	2839	1036.12	21190	2358.29	32867	75.15	186	309.11	2092
13	58	41.62	1341	61.14	294	63.20	263	69.47	270	30.63	50
14	59	385.56	5404	2112.99	26000	2949.28	36740	119.07	150	418.08	850
15	59	436.69	6356	1229.78	13600	2871.40	39516	681.76	4342	711.45	2471
16	59	486.96	6594	6634.58	138500	714.33	13900	277.26	2094	370.70	3139
17	61	222.40	4952	462.60	6945	971.25	17532	194.80	750	938.85	6960
18	62	21.62	0	23.85	0	16.79	0	17.18	0	19.75	0
19	62	298.46	5694	6896.48	100300	1174	17000	2386.77	19365	2245.85	16530
20	65	195.73	4928	92.71	183	599.31	9600	45.39	50	357.85	2493
21	66	309.38	5219	-	-	4132.92	46619	189.49	479	368.53	2048
22	68	51.86	639	22.16	0	18.84	0	18.33	0	17.95	0
23	71	154.04	3322	104.15	223	115.67	731	263.16	2255	64.25	165
24	75	36.66	174	75.54	281	93.67	416	54.95	130	71.94	230
25	79	93.71	335	288.46	1496	414.23	4070	710.89	5577	92.73	210

Table 1: Computational results on UC instances

The results show that modified orbital branching performs in general significantly better than the original orbital branching, sometimes by as much as one order of magnitude. Default CPLEX (with dynamic search) seems to perform better than traditional orbital branching, while the traditional branch-and-cut version of CPLEX is noticeably weaker. In fact, the difference between default CPLEX and CPLEX using the traditional branch-and-cut is striking. It begs the question of how effective modified orbital branching would be on these instances if it were incorporated into default CPLEX.

In order to better understand the impact of modified orbital branching consider the partial solution

$$x^i = \begin{pmatrix} 1 & ? & ? & ? & ? \\ ? & ? & ? & 1 & ? \\ ? & 1 & ? & ? & ? \\ ? & 1 & 1 & ? & ? \end{pmatrix}.$$

In the subproblem, all of the column permutations have been removed from the symmetry group. But suppose that the corresponding LP solution is

$$x^i_{LP} = \begin{pmatrix} 1 & .95 & 1 & ? & ? \\ ? & ? & ? & 1 & ? \\ .97 & 1 & 1 & ? & ? \\ .95 & 1 & 1 & ? & ? \end{pmatrix}. \tag{16}$$

Technically, there is no symmetry found in this subproblem. However, it is likely that the optimal solution to this problem has each of $x_{1,2}$, $x_{3,1}$, and $x_{4,1}$ equal to one. If each of these variables had been fixed, then all permutations of the first three columns of x would be in the subproblem's symmetry group, and those permutations could be used to strengthen the branching disjunction.

We let CPLEX choose our branching candidate, then augment that branching decision using orbital branching. One problem with this approach is that if $x_{i,j}$ is chosen to branch on at a node, then it is unlikely that $x_{i,k}$ will be chosen at a child node. This is especially true in cases similar to (16). Variables that take a value close to one in the LP solution (and especially if they are equal to one) will likely not be chosen for branching because doing so will not improve the bound. However, from a symmetry point of view, those variables need to be fixed to create larger symmetry groups and strengthen the subsequent branches. To exploit symmetry early in the branch-and-bound tree, it is important to branch on variables in the same row as previously fixed variables, but this is not a good branching strategy from a general MILP point of view. Using modified orbital branching makes it more likely that several variables in each row are fixed, so there is a better chance that this symmetry will be recognized.

Interestingly, there is not a significant difference between Modified OB and Modified OB Lex. One possible explanation for this is that the loss of branching flexibilty balances out with the full removal of isomorphic solutions. Another explanation might be found in the structure of the UC problem. Because of the

minimum up and downtimes, branching on one variable can have a significant effect on several other variables. For instance, if a generator was fixed to be on at time t then fixed to be off at time $t+1$, then the generator must be off for several more time periods to satisfy the minimum downtime constraint.

6 Conclusion

This paper explores how the specific structure of the symmetry group of a mixed-integer linear programming problem can be used to strengthen orbital branching. This strengthening, called modified orbital branching, can have a considerable impact on the overall time needed to obtain a global optimal solution. An important class of problems to which modified orbital branching applies is the orbitopes (with all column permutations in the symmetry group). Modified orbital branching can be used to enumerate all nonisomorphic solutions to these problems in polynomial time (when the number of rows in the orbitope is fixed).

Acknowledgements: The research of the second and third authors was partially supported by NSERC, the Natural Sciences and Engineering Research Council of Canada.

References

[1] Arroyo, J.M., Conejo, A.J.: Optimal response of a thermal unit to an electricity spot market. IEEE Transactions on Power Systems **15**(3), 1098–1104 (2000)

[2] Carrion, M., Arroyo, J.: A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem. IEEE Transactions on Power Systems **21**(3), 1371 –1378 (2006)

[3] Chang, G., Tsai, Y., Lai, C.: A practical mixed integer linear programming based approach for unit commitment. PES General Meeting pp. 221 – 225 (2004)

[4] Frangioni, A., Gentile, C.: Perspective cuts for a class of convex 0-1 mixed integer programs. Mathematical Programming **106** (2006)

[5] Frangioni, A., Gentile, C., Lacalandra, F.: Tighter approximated milp formulations for unit commitment problems. IEEE Transactions on Power Systems **24**(1) (200)

[6] Friedman, E.J.: Fundamental domains for integer programs with symmetries. In: Dress, A.W.M., Xu, Y., Zhu, B. (eds.) Combinatorial Optimization and Applications, *Lecture Notes in Computer Science*, vol. 4616, pp. 146–153. Springer (2007)

[7] Gattermann, K., Parrilo, P.: Symmetry groups, semidefinite programs, and sums of squares. Journal of Pure and Applied Algebra **192**, 95–128 (2004)

[8] Jeroslow, R.: Trivial integer programs unsolvable by branch-and-bound. Mathematical Programming **6**, 105–109 (1974)

[9] Kaibel, V., Loos, A.: Branched polyhedral systems. In: IPCO 2010: The Fourteenth Conference on Integer Programming and Combinatorial Optimization, *Lecture Notes in Computer Science*, vol. 6080, pp. 177–190. Springer (2010)

[10] Kaibel, V., Peinhardt, M., Pfetsch, M.E.: Orbitopal fixing. Discrete Optimization **8**(4), 595 – 610 (2011)

[11] Kaibel, V., Pfetsch, M.: Packing and partitioning orbitopes. Mathematical Programming **114**, 1–36 (2008)

[12] Liu, C., Shahidehpour, M., Li, Z., Fotuhi-Firuzabad, M.: Component and mode models for the short-term scheduling of combined-cycle units. IEEE Transactions on Power Systems **24**(2), 976 –990 (2009)

[13] Margot, F.: Pruning by isomorphism in branch-and-cut. Mathematical Programming **94**, 71–90 (2002)

[14] Margot, F.: Exploiting orbits in symmetric ILP. Mathematical Programming, Series B **98**, 3–21 (2003)

[15] Margot, F.: Symmetry in integer linear programming. In: Jünger, M., Liebling, T.M., Naddef, D., Nemhauser, G.L., Pulleyblank, W.R., Reinelt, G., Rinaldi, G., Wolsey, L.A. (eds.) 50 Years of Integer Programming 1958-2008, pp. 647–686. Springer Berlin Heidelberg (2010)

[16] Ostrowski, J., Anjos, M.F., Vannelli, A.: Tight mixed integer linear programming formulations for the unit commitment problem. IEEE Transactions on Power Systems **27**(1), 39 –46 (2012)

[17] Ostrowski, J., Linderoth, J., Rossi, F., Smriglio, S.: Orbital branching. Mathematical Programming **126**(1), 147–178 (2009)

[18] Rajan, D., Takriti, S.: Minimum up/down polytopes of the unit commitment problem with start-up costs. Tech. rep., IBM Research Report (2005)

[19] Sherali, H.D., Smith, J.C.: Improving zero-one model representations via symmetry considerations. Management Science **47**(10), 1396–1407 (2001)

[20] Zhai, Q., Guan, X., Cui, J.: Unit commitment with identical units successive subproblem solving method based on Lagrangian relaxation. IEEE Transactions on Power Systems **17**(4), 1250 – 1257 (2002)