Constraint Orbital Branching

JAMES OSTROWSKI

Department of Industrial and Systems Engineering,
Lehigh University
200 W. Packer Ave. Bethlehem, PA 18015, USA

jao204@lehigh.edu

JEFF LINDEROTH

Department of Industrial and Systems Engineering,
University of Wisconsin-Madison
3226 Mechanical Engineering Building, 1513 University Avenue, Madison, WI 53706, USA

linderoth@wisc.edu

FABRIZIO ROSSI, STEFANO SMRIGLIO

Dipartimento di Informatica,
Università di L'Aquila
Via Vetoio I-67010 Coppito (AQ), Italy

rossi@di.univaq.it · smriglio@di.univaq.it

November 15, 2007

Abstract

Orbital branching is a method for branching on variables in integer programming that reduces the likelihood of evaluating redundant, isomorphic nodes in the branch-and-bound procedure. In this work, the orbital branching methodology is extended so that the branching disjunction can be based on an arbitrary constraint. Many important families of integer programs are structured such that small instances from the family are embedded in larger instances. This structural information can be exploited to define a group of strong constraints on which to base the orbital branching disjunction. The symmetric nature of the problems is further exploited by enumerating non-isomorphic solutions to instances of the small family and using these solutions to create a collection of typically easy-to-solve integer programs. The solution of each integer program in the collection is equivalent to solving the original large instance. The effectiveness of this methodology is demonstrated by computing the optimal incidence width of Steiner Triple Systems and minimum cardinality covering designs.

Keywords: Integer programming; symmetry; branch-and-bound algorithms; Steiner Triple Systems; Covering Designs

1 Introduction

Symmetry has long been considered an obstacle to solving integer programs. Recently, there has been significant work on combating symmetry in integer programs. A technique used by a variety of authors is to add inequalities that exclude symmetric feasible solutions [Macambira et al., 2004; Rothberg, 2000; Sherali and Smith, 2001]. Kaibel and Pfetsch [2007] formalize many of these arguments by defining and studying the properties of a polyhedron known as an *orbitope*, the convex hull of lexicographically maximal solutions with respect to a symmetry group. Kaibel et al. [2007] then use the properties of orbitopes to remove symmetry in partitioning problems. Another technique for combating symmetry is to recognize pairs of nodes of the enumeration tree that will result in symmetric feasible solutions. One of the two nodes may safely be pruned without excluding all optimal solutions from the search. This *isomorphism-free backtracking* procedure has long been used in the combinatorics community, e.g. [Read, 1998; Butler and Lam, 1985; McKay, 1998] and was introduced in the integer programming community with the name *isomorphism pruning* by Margot [2002]. Ostrowski et al. [2007] introduce a technique related to isomorphism pruning, called *orbital branching*. The fundamental idea behind orbital branching is to select a branching variable that is equivalent to other variables with respect to the symmetry remaining in the problem. In this work, we extend the work of Ostrowski et al. [2007] to the case of branching on disjunctions formed by inequalities—*constraint orbital branching*.

Exploiting the symmetry in the problem when branching on more general disjunctions of this form can often be significantly strengthened by exploiting certain types of embedded subproblem structure. Specifically, if the disjunction on which the branching is based is such that relatively few non-isomorphic feasible solutions may satisfy one side of the disjunction, then portions of potential feasible solutions may be enumerated. The original problem instance is then partitioned into smaller, more tractable problem instances. As an added benefit, the smaller instances can then easily be solved in parallel. A similar technique has been recently employed in an ad-hoc fashion by Linderoth et al. [2007] in a continuing effort to solve an integer programming formulation for the football pool problem. This work poses a general framework for solving difficult, symmetric integer programs in this fashion.

The power of the constraint orbital branching is demonstrated by solving to optimality for the first time a well-known integer program to compute the incidence width of a Steiner Triple System with 135 elements. Previously, the largest instance in this family that was solved contained 81 elements [Mannino and Sassano, 1995]. The generality of the constraint orbital branching procedure is further illustrated by an application to the construction of minimum cardinality covering designs. In this case, the previously best known bounds from the literature are easily reproduced.

The remainder of this section contains some mathematical preliminaries, and the subsequent paper is divided into four sections. In Section 2, the constraint orbital branching method is presented and proved to be a valid branching methodology. Section 3 discusses properties of good disjunctions for the constraint orbital branching method. Section 4 describes our computational experience with the constraint orbital branching method, and conclusions are given in Section 5.

1.1 Preliminaries

The primary focus of this work is on set covering problems of the form

$$\min_{x \in \mathcal{F}} \{e^T x\}, \text{ with } \mathcal{F} \stackrel{\text{def}}{=} \{x \in \{0,1\}^n \mid Ax \geq e\}, \tag{1}$$

where $A \in \{0,1\}^{m \times n}$ and e is a vector of ones of conformal size. The restriction of our work to set covering problems is mainly for notation convenience, but also of practical significance, since many important set covering problems contain a great deal of symmetry.

Before describing constraint orbital branching, we first define some notation. Let Π^n be the set of all permutations of $I^n = \{1, \ldots, n\}$, so that Π^n is the symmetric group of I^n. For a vector $\lambda \in \mathbb{R}^n$, the permutation $\pi \in \Pi^n$ acts on λ by permuting its coordinates, a transformation that we denote as

$$\pi(\lambda) = (\lambda_{\pi_1}, \lambda_{\pi_2}, \ldots \lambda_{\pi_n}).$$

Throughout this paper we display permutations 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. Some 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 will omit 1-element cycles from our display.

Since all objective function coefficients in (1) are identical, permuting the coordinates of a solution does not change its objective value, i.e. $e^T x = e^T(\pi(x)) \forall x \in \mathcal{F}$. The *symmetry group* G of (1) is the set of permutations of the variables that maps each feasible solution onto a feasible solution of the same value. In this case,

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

Typically, the symmetry group $\mathcal{G}$ of feasible solutions is not known. However, by closely examining the structure of the problem, many of the permutations making up the group can be found, and this subgroup of the original group $\mathcal{G}$ can be employed in its place. Specifically, given a permutation $\pi \in \Pi^n$ and a permutation $\sigma \in \Pi^m$, let $A(\pi, \sigma)$ be the matrix obtained by permuting the columns of A by π and the rows of A by σ, i.e. $A(\pi, \sigma) = P_\sigma A P_\pi$, where P_σ and P_π are permutation matrices. Consider the set of permutations

$$\mathcal{G}(A) \stackrel{\text{def}}{=} \{\pi \in \Pi^n \mid \exists \sigma \in \Pi^m \text{ such that } A(\pi, \sigma) = A\}.$$

For any $\pi \in \mathcal{G}(A)$, if $\hat{x} \in \mathcal{F}$, then $\pi(\hat{x}) \in \mathcal{F}$, so $\mathcal{G}(A)$ forms a subgroup of $\mathcal{G}$, and the group $\mathcal{G}(A)$ is the group used in our computations. The group $\mathcal{G}(A)$ can act on an arbitrary set of points $\mathcal{Z}$, but in our work, it acts on either $\mathbb{R}^n$ or $\{0,1\}^n$.

For a point $z \in \mathcal{Z}$, 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, i.e.,

$$\text{orb}(\Gamma, z) \stackrel{\text{def}}{=} \{z' \in \mathcal{Z} \mid \exists \pi \in \Gamma \text{ such that } z' = \pi(z)\} = \{\pi(z) \mid \pi \in \Gamma\}.$$

The stabilizer of a set $S \subseteq I^n$ in Γ is the set of permutations in Γ that send S to itself.

$$\text{stab}(S, \Gamma) = \{\pi \in \Gamma \mid \pi(S) = S\}.$$

The stabilizer of S is a subgroup of Γ.

At node a, the set of feasible solutions to (1) is denoted by $\mathcal{F}(a)$, and the value of an optimal solution for the subtree rooted at node a is denoted by $z^*(a)$. Two subproblems a and b are *isomorphic* if $x \in \mathcal{F}(a) \Rightarrow \exists \pi \in \mathcal{G}$ with $\pi(x) \in \mathcal{F}(b)$.

2 Constraint Orbital Branching

Constraint orbital branching is based on the following simple observations. If $\lambda^T x \leq \lambda_0$ is a valid inequality for (1) and $\pi \in \mathcal{G}$, then $\pi(\lambda)^T x \leq \lambda_0$ is also a valid inequality for (1). In constraint orbital branching, given an integer vector $(\lambda, \lambda_0) \in \mathbb{Z}^{n+1}$, we will branch on a base disjunction of the form

$$(\lambda^T x \leq \lambda_0) \vee (\lambda^T x \geq \lambda_0 + 1),$$

simultaneously considering all symmetrically equivalent forms of $\lambda x \leq \lambda_0$. Specifically, the branching disjunction is

$$\left(\bigvee_{\mu \in \mathrm{orb}(\mathcal{G},\lambda)} \mu^T x \leq \lambda_0 \right) \vee \left(\bigwedge_{\mu \in \mathrm{orb}(\mathcal{G},\lambda)} \mu^T x \geq \lambda_0 + 1 \right).$$

Further, by exploiting the symmetry in the problem, one need only consider one representative problem for the left portion of this disjunction. That is, either the "equivalent" form of $\lambda x \leq \lambda_0$ holds for one of the members of $\mathrm{orb}(\mathcal{G}, \lambda)$, or the inequality $\lambda x \geq \lambda_0 + 1$ holds for all of them. This is obviously a feasible division of the search space. Theorem 1 demonstrates that for any vectors $\mu_i, \mu_j \in \mathrm{orb}(\mathcal{G}, \lambda)$, the subproblem formed by adding the inequality $\mu_i^T x \leq \mu_0$ is equivalent to the subproblem formed by adding the inequality $\mu_j^T x \leq \mu_0$. Therefore, we need to keep only one of these representative subproblems, pruning the $|\mathrm{orb}(\mathcal{G}, \lambda)| - 1$ equivalent subproblems. The orbital branching (on variables) method of Ostrowski et al. [2007], is a special case of constraint orbital branching for $(\lambda, \lambda_0) = (e_k, 0)$.

Theorem 1 *Let a be a generic subproblem and $\mu_i, \mu_j \in \mathrm{orb}(\mathcal{G}, \lambda)$. Denote by b the subproblem formed by adding the inequality $\mu_i^T x \leq \mu_0$ to a and by c the subproblem formed by adding the inequality $\mu_j^T x \leq \mu_0$ to a. Then, $z^*(b) = z^*(c)$.*

Proof. Let x^* be an optimal solution of b. WLOG, we can assume that $z^*(b) \leq z^*(c)$. Since μ_i and μ_j are in the same orbit, there exists a permutation $\sigma \in \mathcal{G}$ such that $\sigma(\mu_i) = \mu_j$. By definition of $\mathcal{G}$, $\sigma(x^*)$ is a feasible solution to the subproblem with objective value $z^*(b)$. For any permutation matrix P we have that $P^T P = I$. Since x^* is in b, $\mu_i^T x^* \leq \mu_0$. We can rewrite this as $\mu_i^T P_\sigma^T P_\sigma x^* \leq \mu_0$, or $(P_\sigma \mu_i)^T P_\sigma x^* \leq \mu_0$. This implies that $\mu_j P_\sigma x^* \leq \mu_0$, so $\sigma(x^*)$ is in c. This implies that $z^*(c) \leq z^*(b)$. By our assumption, $z^*(c) = z^*(b)$. □

The basic *constraint orbital branching* is formalized in Algorithm 1.

Algorithm 1 Constraint Orbital Branching

Input:	Subproblem a.
Output:	Two child subproblems b and c.
Step 1.	Choose a vector of integers λ of size n and an integer λ_0
Step 2.	Compute the orbit of λ, $\mathcal{O} = \{\mu_1, \ldots, \mu_p\}$.
Step 3.	Choose arbitrary $\mu_k \in O$. Return subproblems b with $\mathcal{F}(b) = \mathcal{F}(a) \cap \{x \in \{0,1\}^n : \mu_k^T x \leq \lambda_0\}$ and c with $\mathcal{F}(c) = \mathcal{F}(a) \cap \{x \in \{0,1\}^n : \mu_i^T x \geq \lambda_0 + 1, i = 1, \ldots, p\}$

As for standard branching on constraints, the critical choice in Algorithm 1 is in choosing the inequality (λ, λ_0) [Karamanov and Cornuéjols, 2005]. When dealing with symmetric problems, the embedded subproblem structure can be exploited to find strong branching disjunctions, as described in the next section.

3 Strong Branching Disjunctions, Subproblem Structure, and Enumeration

Many important families of symmetric integer programs are structured such that small instances from the family are embedded in larger instances. In this case the problem often shows a block-diagonal structure with identical blocks and some linking constraints, like expressed in Figure 1.

The subproblem $z = \min_{x\in\{0,1\}^n}\{e^T x|\ Ax \geq e\}$, denoted by P, is often computationally manageable and can be solved to optimality in reasonable time. Constraint orbital branching allows us to exploit its optimal value z. The first step consists in choosing an index $i \in \{1, \ldots, r\}$ and enforcing the constraint $e^T x^i \geq z$, which, obviously, does not cut off any optimal solution of the whole problem. Then, the new constraint is used as branching disjunction by letting $\lambda = [0_n, \ldots, \lambda_i, \ldots 0_n]$, $\lambda_i = e_n$ and $\lambda_0 = z$. The resulting child subproblems have interesting properties.

$$\min e^T x^1 + e^T x^2 + \ldots + e^T x^r$$

s.t.

$$\begin{pmatrix} A & & & \\ & A & & \\ & & \ddots & \\ & & & A \\ D_1 & D_2 & \ldots & D_r \end{pmatrix} \begin{pmatrix} x^1 \\ x^2 \\ \vdots \\ x^r \end{pmatrix} \geq e$$

$$x^i \in \{0,1\}^n, i = 1, \ldots, r$$

Figure 1: Block Diagonal IP

Left subproblem In the left child, the constraint $e_n^T x^i \leq z$ is added. Since also $e_n^T x^i \geq z$ holds, this is equivalent to $e_n^T x^i = z$. Therefore, the feasible sub-vectors x^i for the left subproblem coincides with the set of the solutions of P with objective value equal to z. Denoting by $\{x_1^*, x_2^*, \ldots, x_l^*\}$ the set of such (optimal) solutions, one can solve the left subproblem by dividing it into l subproblems, each associated with a solution x_j^*, for $j = 1, \ldots, l$. Precisely, each child j is generated by fixing $x^i = x_j^*$. This yields two relevant benefits. First, the resulting integer programs are easier than the original. Second, these are completely independent and can be solved in parallel. Of course, this option is viable only if the number of optimal solutions of P is reasonably small. Otherwise, one can select an index $k \neq i$ and choose $e_n^T x^k \geq z$ as a branching disjunction. In §4.1 we show how to address this "branching or enumerating" decision for well-known difficult set covering problems.

However, a more insightful observation can lessen the number of subproblems to be evaluated as children of the left subproblem. Suppose to know a symmetry group $\mathcal{G}(P) \subseteq \Pi^n$ with the property that any two solutions in P which are isomorphic with respect to $\mathcal{G}(P)$ generate subproblems in the original problem which are isomorphic. If such a group exists, then one can limit the search in the left subproblem only to the children corresponding to solutions x_j^* non-isomorphic with respect to $\mathcal{G}(P)$.

The group $\mathcal{G}(P)$ is created as follows. Let $I = \{i \cdot n + 1,\ \ldots,\ (i+1)n\}$ be the column indices representing the elements of x^i. First, compute the group $\mathrm{stab}(I, \mathcal{G})$. Note that this group is in $\Pi^{r\times n}$, but we are only interested in how each $\pi \in \mathrm{stab}(I, \mathcal{G})$ permutes the n elements in I. For this reason, we *project* $\mathrm{stab}(I, \mathcal{G})$ onto I. Every permutation $\pi \in \mathrm{stab}(I, \mathcal{G})$ can be expressed as a product of two smaller permutations, $\phi \in \Pi^I$ and $\gamma \in \Pi^{n-I}$, where ϕ permutes the elements in I and γ permutes the elements not in I. We can write this as $\pi = (\phi, \gamma)$. The projection of $stab(I, \mathcal{G})$ onto I, $\mathcal{G} \downarrow_I$, contains all ϕ such that there exists a γ with $(\phi, \gamma) \in \mathrm{stab}(I, \mathcal{G})$. Note that permutations not in $\mathrm{stab}(I, \mathcal{G})$ cannot be projected in this way, so it is unambiguous to describe this set as $\mathcal{G} \downarrow_I$.

Theorem 2 *The projection of $\mathcal{G}$ onto I, $\mathcal{G} \downarrow_I$, is a subset of $\mathcal{G}(P)$.*

Proof. Let $\phi \in \mathcal{G} \downarrow_I$. Let x be any optimal solution of P. By definition, x and $\phi(x)$ are isomorphic with respect to $\mathcal{G} \downarrow_I$. Consider the subproblems formed by setting $x^i = x$ (subproblem a) and $x^i = \phi(x)$ (subproblem b). By definition, there is a $\gamma \in \Pi^{n-I}$ with $\pi = (\phi, \gamma) \in \mathcal{G}$.

Let x^* be any integer feasible solution in a. By definition of permutation, we know that $\pi(x^*)$ is feasible at the root node. Also π sends x^i to $\phi(x^i)$. Since b differs from the root node only by the constraint $x^i = \phi(x^i)$, we have that $\pi(x^*)$ is in b. To conclude, any solutions to P which are isomorphic with respect to $\mathcal{G} \downarrow_I$ will generate subproblems which are isomorphic.

□

Corollary 1 *The left subproblem can be decomposed into a set of restricted subproblems associated with the optimal solutions to P which are non-isomorphic with respect to $\mathcal{G}\downarrow_I$.*

In practice, non-isomorphic optimal solutions of symmetric problems often represent a small portion of all the optimal solutions. In this cases, enumerating the left subproblem becomes computationally very efficient, as shown in the case studies of Section 4.1.

Right subproblem In the right branch, the constraints $\mu^T x \geq \lambda_0 + 1$, for all $\mu \in \mathrm{orb}(\mathcal{G}, \lambda)$, are added. If $|\mathrm{orb}(\mathcal{G}, \lambda)|$ is fairly large, then the LP bound is considerably increased.

The whole branching process can be iterated at the right child. In fact, the constraint $e_n^T x^i \geq z+1$ can be exploited as branching disjunction. In this case all the solution to P with value $z+1$ should be enumerated to solve the new left branch.

Example: Consider the graph $G = (V, E)$ of Figure 2 and the associated vertex cover problem

$$\min_{x\in\{0,1\}^{|V|}} \left\{e^T x \mid x_i + x_j \geq 1 \quad \forall (i,j) \in E\right\}.$$

Its optimal solution has value 10 and it is supposed to be known. The coefficient matrix A shows a block diagonal structure with three blocks, corresponding to the incidence matrices of the 5-holes induced by vertices $\{1,\ldots,5\}$, $\{6,\ldots,10\}$ and $\{11,\ldots,15\}$ respectively. Therefore, the i-th subproblem, $i \in \{0,1,2\}$, has the form

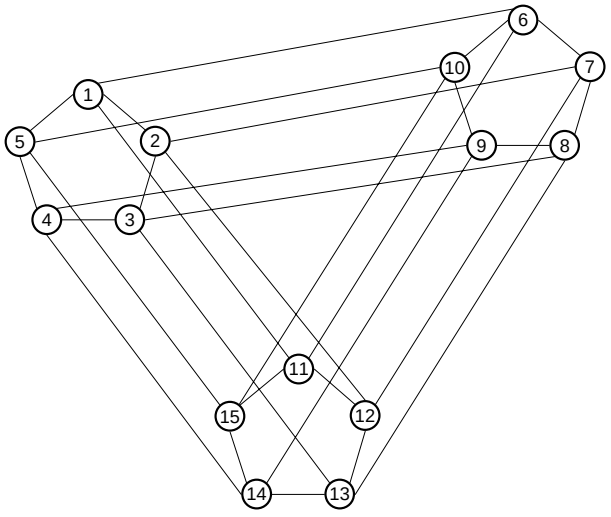


Figure 2: Example Graph

$$P: \min\ x_{5i+1} + x_{5i+2} + x_{5i+3} + x_{5i+4} + x_{5i+5}$$

s.t.

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{5i+1} \\ x_{5i+2} \\ x_{5i+3} \\ x_{5i+4} \\ x_{5i+5} \end{pmatrix} \geq e$$

$$x \in \{0,1\}^5$$

The group $\mathcal{G}(A)$ contains 60 permutations in Π^{15} and is generated by the following permutations:

$$\pi^1 = (2,\ 5)(3,\ 4)(7,\ 10)(8,\ 9)(12,\ 15)(13,\ 14) \qquad \pi^2 = (6,\ 11)(7,\ 12)(8,\ 13)(9,\ 14)(10,\ 15)$$
$$\pi^3 = (1,\ 2)(3,\ 5)(6,\ 7)(8,\ 10)(11,\ 12)(13,\ 15) \qquad \pi^4 = (1,\ 6)(2,\ 7)(3,\ 8)(4,\ 9)(5,\ 10)$$

$\mathcal{G}(P)$ can be created by projecting $\mathcal{G}(A)$ on the variables of the first block (i.e., $x_1,\ldots,x_5$). It consists of 10 permutations in Π^5 which are generated by $(2,\ 5)(3,\ 4)$, and $(1,\ 2)(3,\ 5)$.
The optimal solution to P has value 3 and there is only one non-isomorphic cover of size 3 (for instance,

$x_1 = 1$, $x_2 = 1$ and $x_4 = 1$). At the root node we branch on the disjunction $\lambda = (1, 1, 1, 1, 1, 0, \ldots, 0)$, $\lambda_0 = 3$. Then, in the left subproblem the constraint $x_1 + x_2 + x_3 + x_4 + x_5 \leq 3$ is added, while in the right subproblem the constraints $x_1 + x_2 + x_3 + x_4 + x_5 \geq 4$, $x_6 + x_7 + x_8 + x_9 + x_{10} \geq 4$ and $x_{11} + x_{12} + x_{13} + x_{14} + x_{15} \geq 4$ are enforced.

Since P has a unique non-isomorphic optimal solution, searching the left child amounts to solve only one subproblem with $x_1 = 1$, $x_2 = 1$, $x_3 = 0$, $x_4 = 1$ and $x_5 = 0$. Its optimal value is 10 and the subproblem can be fathomed. On the right branch, the lower bound increases to 12 and also that subproblem can be fathomed.

If a classical variable-branching dichotomy is applied, it results in a much larger enumeration tree (15 subproblems vs. 3).

In the general case of unstructured problems, finding effective branching disjunctions may be difficult. Nevertheless, branching on a constraint (λ, λ_0) such that the number of non-isomorphic optimal solutions to the left subproblem is fairly small still gives good results, as shown in Section 4.3.

4 Case Studies

4.1 Steiner Triple Systems

A Steiner Triple System of order v, denoted by STS(v), consists of a set S with v elements, and a collection $\mathcal{B}$ of triples of S with the property that every pair of elements in S appears together in a unique triple of $\mathcal{B}$. Kirkman [1847] showed that STS(v) exists if and only if $v \equiv 1$ or $3 \mod 6$. A covering of a STS is a subset C of the elements of S such that $C \cap T \neq \emptyset$ for each triple $T \in \mathcal{B}$. The *incidence width* of a STS is its smallest-size covering. Fulkerson et al. [1974] suggested the following integer program to compute the incidence width of a STS(v):

$$\min_{x \in \{0,1\}^v} \{e^T x \mid A_v x \geq 1\},$$

where $A_v \in \{0,1\}^{|\mathcal{B}| \times v}$ is the incidence matrix of the STS(v). Fulkerson et al. [1974] created instances based on STS of orders $v \in \{9, 15, 27, 45\}$, and posed these instances as a challenge to the integer programming community. The instance STS(45) was not solved until five years later by H. Ratliff, as reported by Avis [1980].

The instance of STS(27) was created from STS(9) and STS(45) was created from STS(15) using a "tripling" procedure described in Hall [1967]. We present the construction here, since the symmetry induced by the construction is exploited by our method in order to solve larger instances in this family. For ease of notation, let the elements in STS(v) be $\{1, 2, \ldots v\}$, with triples $\mathcal{B}_v$. The elements of STS$(3v)$ are the pairs $\{(i, j) \mid i \in \{1, 2, \ldots, v\}, j \in \{1, 2, 3\}\}$, and its collection of triples is denoted as $\mathcal{B}_{3v}$. Given STS(v), the Hall construction creates the blocks of STS$(3v)$ in the following manner:

- $\{(a, k), (b, k), (c, k)\} \in \mathcal{B}_{3v} \quad \forall \{a, b, c\} \in \mathcal{B}_v,\ \forall k \in \{1, 2, 3\}$,
- $\{((i, 1), (i, 2), (i, 3)\} \in \mathcal{B}_{3v} \quad \forall i \in \{1, \ldots, v\}$,
- $\{(a, \pi_1), (b, \pi_2), (c, \pi_3)\} \in \mathcal{B}_{3v} \quad \forall \{a, b, c\} \in \mathcal{B}_v,\ \forall \pi \in \Pi^3$.

Feo and Resende [1989] introduced two new instances STS(81) and STS(243) created using this construction. STS(81) was first solved by Mannino and Sassano [1995] 12 years ago, and it remains the largest problem instance in this family to be solved. STS(81) is also easily solved by the isomorphism pruning method of Margot [2002] and the orbital branching method of Ostrowski et al. [2007], but neither of these methods seem capable of solving larger STS(v) instances. Karmarkar et al. [1991] introduced the instance STS(135) which is built by applying the tripling procedure to the STS(45) instance of Fulkerson

et al. [1974]. Odijk and van Maaren [1998] have reported the best known solutions to both STS(135) and STS(243), having values 103 and 198 respectively. Using the constraint orbital branching method, we have been able to solve STS(135) to optimality, establishing that 103 is indeed the incidence width.

The incidence matrix, A_{3v}, for an instance of STS$(3v)$ generated by the Hall construction has the form shown in Figure 3, where A_v is the incidence matrix of STS(v) and the matrices D_i have exactly one "1" in every row. Note that A_{3v} has the block-diagonal structure discussed in Section 3, so it is a natural candidate on which to apply the constraint orbital branching methodology.

$$A_{3v} = \begin{bmatrix} A_v & 0 & 0 \\ 0 & A_v & 0 \\ 0 & 0 & A_v \\ I & I & I \\ D_1 & D_2 & D_3 \end{bmatrix},$$

Figure 3: Incidence Matrix of A_{3v}

Furthermore, the symmetry group Γ of the instance STS$(3v)$ created in this manner has a structure that can be exploited. Specifically for STS(135), let $\lambda \in \mathbb{R}^{135}$ be the vector $\lambda = (e_{45}, 0_{90})^T$ in which the first 45 components of the vector are 1, and the last 90 components are 0. It is not difficult to see that the following 12 vectors $\mu_1, \ldots \mu_{12}$ all share an orbit with the point λ. (This fact can also be verified using a computational algebra package such as GAP [2004]).

	$1-15$	$16-30$	$31-45$	$46-60$	$61-75$	$76-90$	$91-105$	$106-120$	$121-135$
μ_1	e	e	e	0	0	0	0	0	0
μ_2	0	0	0	e	e	e	0	0	0
μ_3	0	0	0	0	0	0	e	e	e
μ_4	e	0	0	e	0	0	e	0	0
μ_5	e	0	0	0	e	0	0	0	e
μ_6	e	0	0	0	0	e	0	e	0
μ_7	0	e	0	e	0	0	0	e	0
μ_8	0	e	0	0	e	0	0	e	0
μ_9	0	e	0	0	0	e	e	0	0
μ_{10}	0	0	e	e	0	0	0	e	0
μ_{11}	0	0	e	0	e	0	e	0	0
μ_{12}	0	0	e	0	0	e	0	0	e

As described for the general case in Section 3, to create an effective constraint orbital branching dichotomy, we will use this orbit and also the fact that branching on the disjunction

$$(\lambda x \leq K) \vee (\mu^T x \geq K+1) \; \forall \mu \in \mathrm{orb}(G, \lambda)$$

allows us to enumerate coverings for STS$(v/3)$ in order to solve the left-branch of the dichotomy.

4.2 Computational Results

In this section, results of the computation proving the optimality of the cardinality 103 covering of STS(135) are presented. The optimal solution to STS(45) has value 30. Figure 4 shows the branching tree used by the constraint orbital branching method for solving STS(135). The node E in Figure 4 is pruned by bound, as the solution of the linear programming relaxation at this node is 103.

A variant of the (variable) orbital branching algorithm of Ostrowski et al. [2007] can be used to obtain a superset of all non-isomorphic solutions to an integer program whose objective value is better than a prescribed value K. The method works in a fashion similar to that proposed by Danna et al. [2007]. Specifically, branching and pruning operations are performed until *all* variables are fixed (nodes may not be pruned by integrality). All leaf nodes of the resulting tree are feasible solutions to the integer program whose

Figure 4: Branching Tree for Solution of STS(135)

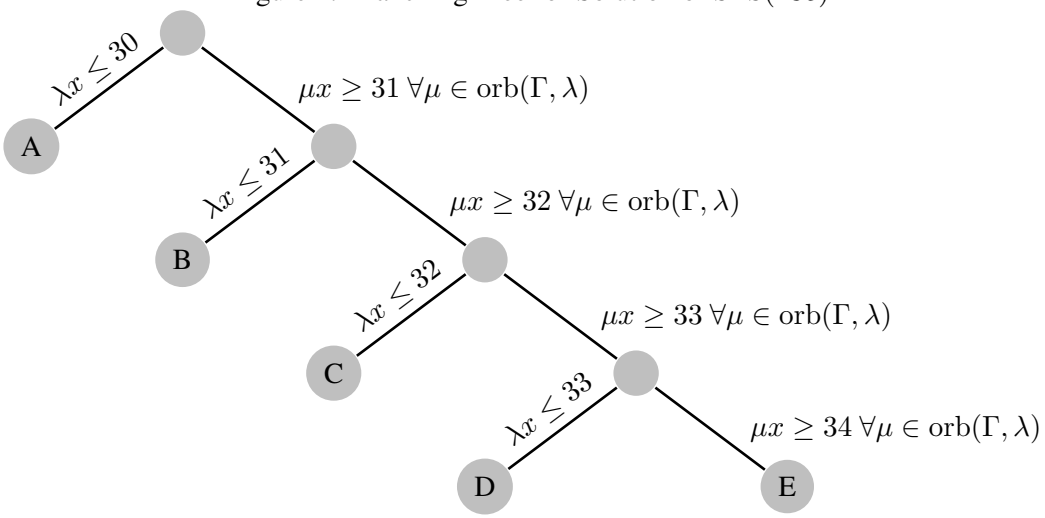


Table 1: Computational Statistics for Solution of STS(135)

(a) Solutions of value K for STS(45)

(K)	# Sol
30	2
31	246
32	9497
33	61539
	71,284

(b) Statistics for STS(135) IP Computations

K	Total CPU Time (sec)	Simplex Iterations	Nodes
30	538.01	2,501,377	164,720
31	90790.94	255,251,657	13,560,519
32	2918630.95	8,375,501,861	306,945,725
33	6243966.98	25,321,634,244	718,899,460
	9.16×10^6	3.36×10^{10}	1.04×10^9

objective value is $\leq K$. Using this algorithm, a superset of all non-isomorphic solutions to STS(45) of value 33 or less was enumerated. The enumeration required 10CPU hours on a 1.8GHz AMD Opteron Processor and resulted in 71,284 solutions. The number of solutions for each value of K is shown in Table 1(a).

For each of the 71,284 enumerated solutions to STS(45), the first 45 variables of the STS(135) integer programming instance for that particular node were fixed. For example, the node B contains the inequalities $\mu x \geq 31\ \forall \mu \in \mathrm{orb}(\Gamma, \lambda)$, and the bound of the linear programming relaxation is 93. In order to establish the optimality of the covering of cardinality 103 for STS(135), each of these 71,284 90-variable integer programs must be solved to establish that no solution of value smaller than 103 exists. The integer programs are completely independent, so it is natural to consider solving them on a distributed computing platform. The instances were solved on a collection of over 800 computers running the Windows Operating System at Lehigh University. The computational grid was created using the Condor High Throughput Computing software [Livny et al., 1997], so the computations were run on processors that would have otherwise been idle. The commercial package CPLEX (v10.2) was used to solve all the instances, and an initial upper bound of value 103.1 was provided to CPLEX as part of the input to all instances. Table 1(b) shows the aggregated statistics for the computation. The total CPU time required to solve all 71,284 instances was roughly 106 CPU days, and the wall clock time required was less than two days. The best solution found during the search had value 103[1], thus establishing that the incidence-width of STS(135) is 103.

[1]In fact, two solutions of value 103 were found, but they were isomorphic

4.3 Covering Designs

A (v, k, t)-*covering design* is a family of subsets of size k, chosen from a ground set V of cardinality $|V| = v$, such that every subset of size t chosen from V is contained in one of the members of the family of subsets of size k. The number of members in the family of k-subsets is the covering design's *size*. The covering number $C(v, k, t)$ is the minimum size of such a covering. Let $\mathcal{K}$ be the collection of all k-sets of V, and let $\mathcal{T}$ be the collection of all t-sets of V. An integer program to compute a (v, k, t)-covering design can be written as

$$\min_{x \in \{0,1\}^{|\mathcal{K}|}} \{e^T x \mid Bx \geq e\}, \tag{2}$$

where $B \in \{0, 1\}^{|\mathcal{T}| \times |\mathcal{K}|}$ has element $b_{ij} = 1$ if and only if t-set i is contained in k-set j, and the decision variable $x_j = 1$ if and only if the jth k-set is chosen in the covering design.

Numerous theorem exist that give bounds on the size of the covering number $C(v, k, t)$. An important theorem that we need to generate a useful branching disjunction for the constraint orbital branching method is due to Schönheim [1964]. For some subset of the ground set elements $U \subseteq V$, let $\mathcal{K}(U)$ be the collection of all the k-sets of V that contain U. Margot [2003a] shows that the following inequality, which he calls a *Schönheim* inequality, is valid, provided that $|U| = u$ is such that $1 \leq u \leq t - 1$:

$$\sum_{i \in \mathcal{K}(U)} x_i \geq C(v - u, k - u, t - u). \tag{3}$$

The *Schönheim* inequalities substantially increase the value of the linear programming relaxation of (2).

A second important observation is that the symmetry group G for (2) is such the characteristic vectors of all u-sets belong to the same orbit: if $|U'| = |U|$, then $\chi_{\mathcal{K}(U')} \in \mathrm{orb}(G, \chi_{\mathcal{K}(U)})$. These two observations taken together indicate that the Schönheim inequalities (3) may be a good candidate for constraint orbital branching. On the left branch, the constraint

$$\sum_{i \in \mathcal{K}(U)} x_i \leq C(v - u, k - u, t - u)$$

is enforced. To solve this node, all non-isomorphic solutions to the $(v - u, k - u, t - u)$-covering design problem may be enumerated. For each of these solutions, an integer program in which the corresponding variables in the (v, k, t)-covering design problem are fixed may be solved.

On the right branch of the constraint-orbital branching method, the constraints

$$\sum_{i \in \mathcal{K}(U')} x_i \geq C(v - u, k - u, t - u) + 1 \qquad \forall U' \in \mathrm{orb}(G, \chi_{\mathcal{K}(U)})$$

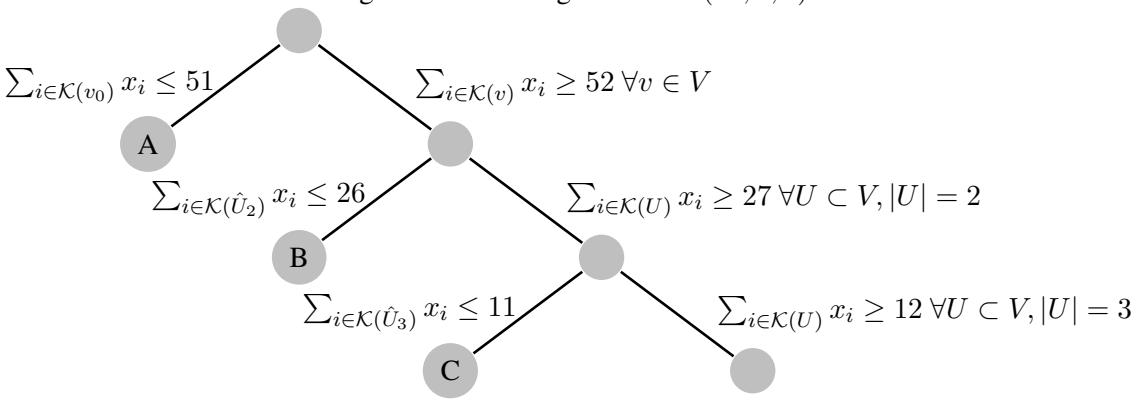
may be imposed. These inequalities can significantly improve the linear programming relaxation.

4.4 Computational Results

We will demonstrate the applicability of constraint orbital branching using the Schönheim inequalities by an application to the $(11, 6, 5)$-covering design problem. Nurmela and Östergård [1993] report an upper bound of $C(11, 6, 5) \leq 100$, and Applegate et al. [2003] were able to show that $C(11, 6, 5) \geq 96$. Using the constraint orbital branching technique, we are also easily able to obtain the bound $C(11, 6, 5) \geq 96$, and ongoing computations are aimed at further sharpening the bound. The covering design numbers $C(10, 5, 4) = 51$, $C(9, 4, 3) = 25$, and $C(8, 3, 2) = 11$ are all known [Gordon et al., 1995], and this knowledge is used in the branching scheme.

The branching tree used for the $(11, 6, 5)$-covering design computations is shown in Figure 5. In the figure, node D is pruned by bound, as the value of its linear programming relaxation is > 100. The nodes A, B, and C will be solved by enumerating solutions to a (v, k, t)-covering design problem of appropriate size. For node A, $(10, 5, 4)$-covering designs of size 51 are enumerated; for node B, $(9, 4, 3)$-covering designs of size ≤ 26 are enumerated; and for node C, $(8, 3, 2)$-covering designs of size ≤ 11 are enumerated. Table 2 shows the number of solutions at each node, as well as the value of the linear programming relaxation $z(\rho)$ of the parent node. The size 51 $(10, 5, 4)$-covering designs are taken from the paper of Margot [2003b], and the other covering designs are enumerated using the variant of the orbital branching method outlined in Section 4.2.

Figure 5: Branching Tree for $C(11, 6, 5)$



Since the value of the linear programming relaxation of the parent of node B is 95.33, if none of the 40 integer programs created by fixing the size 51 $(10, 5, 4)$-covering design solutions at node A of Figure 5 has a solution of value 95, then immediately, a lower bound of $C(11, 6, 5) \geq 96$ is proved. The computation to improve the lower bound for each of the 40 IPs to 95.1 required only 8789 nodes and 10757.5 CPU seconds on a single 2.60GHz Intel Pentium 4 CPU. More extensive computations are currently underway on a Condor-provided computational grid in order to further improve this bound.

It is interesting to note that an attempt to improve the lower bound of $C(11, 6, 5)$ by a straightforward application of the variable orbital branching method of Ostrowski et al. [2007] was unable to improve the bound higher than 94, even after running several days and eventually exhausting a 2GB memory limit. An exhaustive comparison with variable orbital branching will be reported in a journal version of the paper. However, the results on specific classes of problems show that the generality of constraint orbital branching does appear to be useful to solve larger symmetric problems.

Table 2: Node Characteristics

Node	# Sol	$z(\rho)$
A	40	93.5
B	782,238	95.33
C	11	99

5 Conclusions

In this work, we generalized a previous work for branching on orbits of variables (orbital branching) to branching on orbits of constraints (constraint orbital branching). Constraint orbital branching can be especially powerful if the problem structure is exploited to identify a strong constraint on which to base the disjunction and by enumerating all partial solutions that might satisfy the constraint. Using this methodology, we are for the first time able to establish the optimality of the cardinality 103 covering for STS(135).

Acknowledgment

The authors would like to thank François Margot for his insightful comments about this work and Helen Linderoth for hand-keying all 40 non-isomorphic solutions to $C(10, 5, 4)$ into text files. Author Linderoth would like to acknowledge support from the US National Science Foundation (NSF) under grant DMI-0522796, by the US Department of Energy under grant DE-FG02-05ER25694, and by IBM, through the faculty partnership program. The solution of the STS135 instance could not have been achieved were it not or the generous donation of "unlimited" CPLEX licenses by Rosemary Berger and Lloyd Clarke of ILOG. The authors dedicate this paper to the memory of their good friend Lloyd.

References

D. Applegate, E. Rains, and N. Sloane. On asymmetric coverings and covering numbers. *Journal of Combinatorial Designs*, 11:218–228, 2003.

D. Avis. A note on some computationally difficult set covering problems. *Mathematical Programming*, 8: 138–145, 1980.

G. Butler and W. H. Lam. A general backtrack algorithm for the isomorphism problem of combinatorial objects. *Journal of Symbolic Computation*, 1:363–381, 1985.

E. Danna, M. Fenelon, Z. Gu, and R. Wunderling. Generating multiple solutions for mixed integer programming problems. In M. Fischetti and D. Williamson, editors, *IPCO 2007: The Twelfth Conference on Integer Programming and Combinatorial Optimization*, pages 280–294. Springer, 2007.

T. A. Feo and G. C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Reswearch Letters*, 8:67–71, 1989.

D. R. Fulkerson, G. L. Nemhauser, and L. E. Trotter. Two computationally difficult set covering problems that arise in computing the 1-width of incidence matrices of Steiner triples. *Mathematical Programming Study*, 2:72–81, 1974.

GAP—Groups, Algorithms, and Programming, Version 4.4. The GAP Group, 2004. `http://www.gap-system.org`.

D. Gordon, G. Kuperberg, and O. Patashnik. New constructions for covering designs. *Journal of Combinatorial Designs*, 3:269–284, 1995.

M. Hall. *Combinatorial Theory*. Blaisdell Company, 1967.

V. Kaibel and M. Pfetsch. Packing and partitioning orbitopes. *Mathematical Programming*, 2007. To appear.

V. Kaibel, M. Peinhardt, and M.E. Pfetsch. Orbitopal fixing. In *IPCO 2007: The Twelfth Conference on Integer Programming and Combinatorial Optimization*. Springer, 2007. To appear.

M. Karamanov and G. Cornuéjols. Branching on general disjunctions. *submitted*, 2005.

N. Karmarkar, K. Ramakrishnan, and M.Resende. An interior point algorithm to solve computationally difficult set covering problems. *Mathematical Programming, Series B*, 52:597–618, 1991.

T. P. Kirkman. On a problem in combinations. *Cambridge and Dublin Mathematics Journal*, 2:191–204, 1847.

J. Linderoth, F. Margot, and G. Thain. Improving bounds on the football pool problem via symmetry reduction and high-throughput computing. Submitted, 2007.

M. Livny, J. Basney, R. Raman, and T. Tannenbaum. Mechanisms for high throughput computing. *SPEEDUP*, 11, 1997.

E. M. Macambira, N. Maculan, and C. C. de Souza. Reducing symmetry of the SONET ring assignment problem using hierarchical inequalities. Technical Report ES-636/04, Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro, 2004.

Carlo Mannino and Antonio Sassano. Solving hard set covering problems. *Operations Research Letters*, 18:1–5, 1995.

F. Margot. Exploiting orbits in symmetric ILP. *Mathematical Programming, Series B*, 98:3–21, 2003a.

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

F. Margot. Small covering designs by branch-and-cut. *Mathematical Programming*, 94:207–220, 2003b.

D. McKay. Isomorph-free exhaustive generation. *Journal of Algorithms*, 26:306–324, 1998.

K. J. Nurmela and P. Östergård. Upper bounds for covering designs by simulated annealing. *Congressus Numerantium*, 96:93–111, 1993.

Michiel A. Odijk and Hans van Maaren. Improved solutions to the Steiner triple covering problem. *Information Processing Letters*, 65(2):67–69, 29 January 1998.

J. Ostrowski, J. Linderoth, F. Rossi, and S. Smriglio. Orbital branching. In M. Fischetti and D. Williamson, editors, *IPCO 2007: The Twelfth Conference on Integer Programming and Combinatorial Optimization*, pages 104–118. Springer, 2007.

R. C. Read. Every one a winner or how to avoid isomorphism search when cataloguing combinatorial configurations. *Annals of Discrete Mathematics*, 2:107–120, 1998.

E. Rothberg. Using cuts to remove symmetry. Presented at the 17^{th} International Symposium on Mathematical Programming, 2000.

J. Schönheim. On coverings. *Pacific Journal of Mathematics*, 14:1405–1411, 1964.

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