Computer Sciences Department

Solving Large Steiner Triple Covering Problems

Jim Ostrowski
Jeff Linderoth
Fabrizio Rossi
Stefano Smriglio

Technical Report #1663

September 2009

UNIVERSITY OF WISCONSIN MADISON

Solving Large Steiner Triple Covering Problems

James Ostrowski

Department of Management Sciences,
University of Waterloo
200 University Avenue West, Waterloo, Ontario, Canada N2L 3G1, Canada

jostrow@engmail.uwaterloo.ca

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

September 22, 2009

Abstract

Computing the 1-width of the incidence matrix of a Steiner Triple System gives rise to small set covering instances that provide a computational challenge for integer programming techniques. One major source of difficulty for instances of this family is their highly symmetric structure, which impairs the performance of most branch-and-bound algorithms. The largest instance in the family that has been solved corresponds to a Steiner Tripe System of order 81. We present optimal solutions to the set covering problems associated with systems of orders 135 and 243. The solutions are obtained by a tailored implementation of *constraint orbital branching*, a method for branching on general disjunctions designed to exploit symmetry in integer programs.

1 Introduction

A Steiner Triple System of order 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 [7] showed that a Steiner Triple System of order v exists if and only if $v \equiv 1$ or 3 mod 6. A covering of a Steiner Triple System 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 the system is the size of its smallest covering. The problem of computing the incidence width of a Steiner Triple System is known as *Steiner Triple Covering Problem*. Fulkerson, Nemhauser, and Trotter [4] suggested the following integer program for the Steiner Triple Covering Problem

$$z_v \stackrel{\text{def}}{=} \min_{x \in \{0,1\}^v} \{e_v^T x \mid A_v x \geq 1\}, \qquad \text{(STS}(v)\text{)}$$

where $A_v \in \{0,1\}^{|\mathcal{B}| \times v}$ is the incidence matrix of the Steiner Triple System and e_v is a vector of ones of size v. The authors 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 Ratliff, as reported by Avis [1].

Feo and Resende [3] introduced instances STS(81) and STS(243). The instance STS(81) was first solved to optimality by Mannino and Sassano [9] 14 years ago, and it remains the largest solved instance in this family. STS(81) is also easily solved by the isomorphism pruning method of Margot [10] and the orbital branching method of Ostrowski *et al.* [14], but neither of these methods seem capable of solving larger STS(v) instances. Karmarkar, Ramakrishnan, and Resende [6] introduced the instance STS(135). Odijk and van Maaren [12] have reported the best known solutions to both STS(135) and STS(243), having values 103 and 198 respectively. In this paper we prove that these values are indeed the optimal ones. We accomplish this task by a tailored application of *constraint orbital branching*, a branching method illustrated in [15] designed for highly-symmetric integer programs.

The subsequent paper is divided into six sections. In Section 2, we describe the form of the STS instances. The instances are highly symmetric, a concept we formalize in Section 3. In Section 4, we describe a tailored implementation of the constraint orbital branching method to these instances. To prune the nodes of the branching trees, we rely on an enumerative technique in combination with branch and bound, a method described in Section 5. Finally, in Section 6 the computational achievements are presented, and conclusions are offered in Section 7.

2 Steiner Triple Instances

The instance STS(27) was created from STS(9), and STS(45) was created from STS(15) using a "tripling" procedure described by Hall [5]. 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 $\mathcal{B}_v$. In the Hall construction, the elements of STS($3v$) are the pairs $\{(i, j) \mid i \in \{1, 2, \ldots, v\}, j \in \{1, 2, 3\}\}$, and the blocks $\mathcal{B}_{3v}$ are created in the following manner:

1. $\{(a, k), (b, k), (c, k)\} \in \mathcal{B}_{3v} \quad \forall \{a, b, c\} \in \mathcal{B}_v, \ \forall k \in \{1, 2, 3\}$,
2. $\{((i, 1), (i, 2), (i, 3)\} \in \mathcal{B}_{3v} \quad \forall i \in \{1, \ldots, v\}$,
3. $\{(a, i), (b, j), (c, k)\} \in \mathcal{B}_{3v} \quad \forall \ i, \ j, \ k \ \in \{1, 2, 3\}, \ i \neq j \neq k$.

By construction, small instances from the family are embedded into larger instances, as evidenced by the block diagonal structure of the integer program:

$$\min_{x^i \in \{0,1\}^v, i=\{1,2,3\}} e_v^T x^1 + e_v^T x^2 + e_v^T x^3 \begin{pmatrix} A_v & & \\ & A_v & \\ & & A_v \\ I & I & I \\ D_1 & D_2 & D_3 \end{pmatrix} \begin{pmatrix} x^1 \\ x^2 \\ x^3 \end{pmatrix} \geq e \qquad \text{(STS}(3v)\text{)}$$

where A_v is the incidence matrix of STS(v) and the matrices D_i (of size $6 \cdot |\mathcal{B}_v| \times v$) have exactly one "1" in every row. The embedded nature of the instances can be used to generate valid inequalities and strengthen the formulation STS($3v$). For example, since the first v columns of a solution to STS($3v$) must also be a cover for STS(v), the inequality $\lambda_v x \geq z_v$, with $\lambda_v \stackrel{\text{def}}{=} [e_v, 0_{2v}]$ is valid for STS($3v$). Similarly, since the Hall construction process is recursive, the inequality $\sigma_v x \geq z_v$, where $\sigma_v \stackrel{\text{def}}{=} [e_v, 0_{8v}]$ is valid for STS($9v$).

3 Symmetry in Integer Programs

To describe symmetry in integer programs requires some definitions from algebra. The set Π^n is the set of all permutations of $I^n = \{1, \ldots, n\}$. This set (along with the binary operation of composition) forms the *complete symmetric group* of I^n. Any subgroup of the complete symmetric group is a *permutation group*. For a permutation group Γ acting on set of points $\mathcal{Z}$ and 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, or $\text{orb}(\Gamma, z) \stackrel{\text{def}}{=} \{\pi(z) \mid \pi \in \Gamma\}$. The stabilizer of a set S with respect to a 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 $\text{stab}(S, \Gamma)$ is a subgroup of Γ.

The Steiner Triple Covering Problems are strongly characterized by their symmetric structure. If we let $\mathcal{F}$ be the incidence vectors of coverings (or feasible solutions), the *symmetry group* $\mathcal{G}$ of STS(v) is the set of permutations of the variables that maps each feasible solution onto a feasible solution of the same value. Since in the original formulation of STS(v) all objective coefficients are equal, the symmetry group is

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

Solutions x and $\pi(x)$, for any $\pi \in \mathcal{G}$, are called *isomorphic*.

Computing the symmetry group $\mathcal{G}$ of an integer program is NP-hard and typically more difficult than solving the instance itself. As a result, practical methods aimed at exploiting symmetries are forced to use a subgroup of the symmetry group that is found by examining the problem formulation. Let (A, b) be a formulation of STS(v), i.e. $\mathcal{F} = \{x \in \{0,1\}^v, Ax \geq b\}$. Given a permutation $\pi \in I^v$ and a permutation $\sigma \in I^m$, let $A(\pi, \sigma) = P_\sigma A P_\pi$ be the matrix obtained by permuting the columns of A by π and the rows of A by σ, for permutation matrices P_σ and P_π. Applying the permutation to the right hand side vector gives $\sigma(b) = P_\sigma b$. The *formulation group* $\mathcal{G}(A, b)$ of STS(v) is the set of permutations

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

Computing the formulation group $\mathcal{G}(A, b)$ can be accomplished by using software (such as `nauty` [11] or `saucy` [2]) designed computing the isomorphism group of a related graph. More details of the construction are available in the papers [13, 8]. The formulation group $\mathcal{G}(A, b)$ is a subgroup of $\mathcal{G}$. In fact, there may be many formulations (A', b') of the problem, and the symmetries present in any formulation group may be used by symmetry-exploiting methods. Since adding valid inequalities may reduce the size of the formulation group, in our computations we *exclude* valid inequalities like $\lambda_v x \geq z_v$ in the formulation group evaluation, while we *include* such inequalities in evaluating bounds on the optimal solution value.

Given a valid inequality $a^T x \leq b$ for an integer program with a symmetry group $\mathcal{G}$, all symmetrically equivalent forms $d^T x \leq b$, with $d \in \text{orb}(\mathcal{G}, a)$, are also valid inequalities for the integer program. The inequality $\lambda_v x \geq z_v$ is valid for STS($3v$), and by construction, the vectors $\mu_1 = [0_v, e_v, 0_v]$ and $\mu_2 = [0_{2v}, e_v]$ are elements of $\text{orb}(\mathcal{G}(A_{3v}, \lambda))$. Thus, the symmetrically equivalent inequalities $\mu_1 x \geq z_v$ and $\mu_2 x \geq z_v$ may be added to strengthen the formulation. Similarly, many valid inequalities may be constructed from permutations of $\sigma_t = [e_t, 0_{8t}]$, for $t = v/3$.

4 Constraint Orbital Branching

Constraint orbital branching, introduced by Ostrowski *et al.* [15], is a branching method designed to exploit symmetry in integer programs. Given a constraint $a^T x \leq b$ with $(a, b) \in \mathbb{Z}^{n+1}$, the method is based on the fact that either an equivalent form of $a^T x \leq b$ holds for one of the members of $\text{orb}(\mathcal{G}, a)$, or the inequality $a^T x \geq b + 1$ holds for all of them. This result is summarized in Theorem 1.

Theorem 1 *If x is a feasible solution to an integer program with symmetry group $\mathcal{G}$, and $a^T x \leq b$, then there exists a feasible solution $y \in \text{orb}(x, \mathcal{G})$ that satisfies $d^T y \leq b$, for $d \in \text{orb}(a, \mathcal{G})$,*

Proof Let $\pi \in \mathcal{G}$ be any permutation mapping a to d, with associated permutation matrix P_π.

Then

$$a^T x = a^T P_\pi^T P_\pi x = (P_\pi a)^T (P_\pi x) = d^T P_\pi x \leq b.$$

Letting $y = P_\pi x = \pi(x)$ completes the proof. $\Diamond$

The application of Theorem 1 leads to the following constraint orbital branching disjunction:

$$\left(a^T x \leq b\right) \vee \left(\bigwedge_{d \in \mathrm{orb}(\mathcal{G}, a)} d^T x \geq b + 1 \right). \tag{1}$$

To solve STS($3v$), we apply the disjunction (1) on constraints obtained from the optimal solution to smaller embedded STS instances. Specifically, we branch on the constraints $\lambda_v x \leq k$, where $k \in \{z_v, z_v + 1, \ldots\}$ or on the constraints $\sigma_{v/3} x \leq q$, where $q \in \{z_{v/3}, z_{v/3} + 1, \ldots\}$.

Figure 1 shows the branching tree obtained by applying this methodology to solve STS(135). The constraints added on the right branch of the tree (like $\mu^T x \geq 31 \ \forall \mu \in \mathrm{orb}(\mathcal{G}, \lambda)$) greatly improve the lower bound obtained by solving the linear programming relaxation. In fact, the nodes D, F, and G in Figure 1 are pruned by bound, as the value of the linear programming relaxation at these nodes exceeds 102, and a solution of value 103 is known.

Figure 1: Branching Tree for Solution of STS(135)

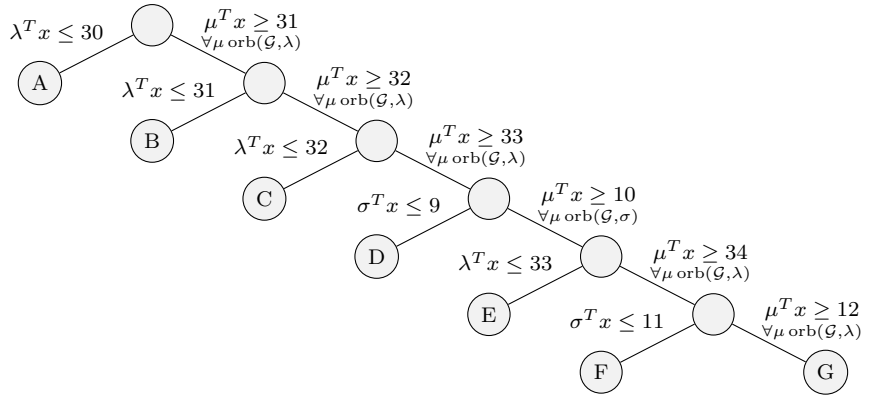


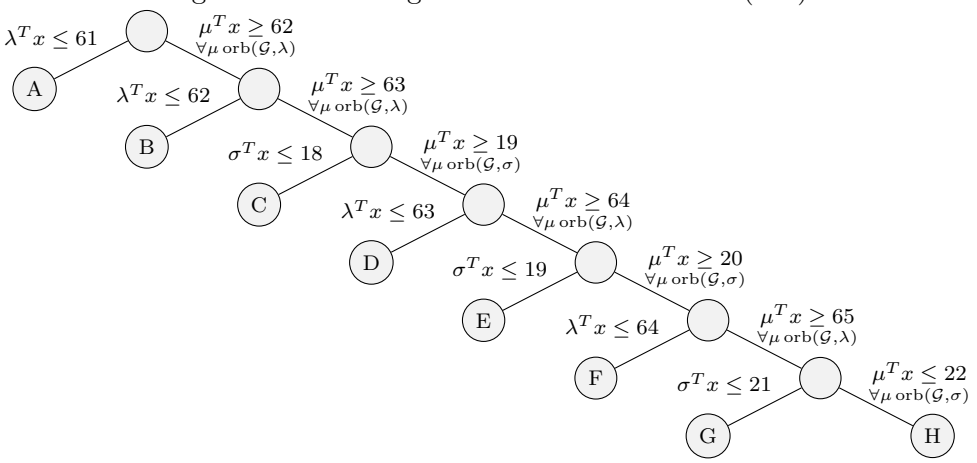
Figure 2 shows the branching tree for solving STS(243). In this case, the nodes C, E, and H in Figure 2 are pruned by bound.

5 Enumerating Solutions

Based on the bounds obtained after applying the orbital branching disjunction, in order to solve STS(135), we must only devise a procedure for processing the nodes A, B, C, and E of the branching tree of Figure 1. Likewise, to solve STS(243), the only the nodes A, B, D, F, and G of the branching tree of Figure 2 need be processed.

One mechanism for processing the nodes would be to solve them with a black-box or commercial IP solver. Processing all nodes in this fashion did not appear to be computationally

Figure 2: Branching Tree for Solution of STS(243)



feasible. An alternative for processing nodes is based on enumeration. By the construction of STS($3v$), for any feasible $x^* = [x^{1^*}, x^{2^*}, x^{3^*}]$ with $\lambda_v x^* \le k$, the first v components of the solution, $x^{1^*} \in \{0,1\}^v$, must be a feasible solution to STS(v) with cardinality at most k. Note that in each branching tree, the first branch enforces the constraint with $k = z_v$, so for x^* to be feasible, x^{1^*} must be an optimal solution to STS(v). The branching tree continues by incrementing k in $\lambda_v x \le k$, or in some cases by branching on $\sigma_{v/3} x \le q$.

Based on this construction, one technique for processing a node that is defined by the branching inequality $\lambda_v x^* \le k$ begins by enumerating *all* solutions $\{y^1, y^2, \ldots y^T\}$ to STS(v) of value k. Then, then for each solution $k = 1, \ldots, T$, fix the first v variables of STS($3v$) to y^k and solve the smaller IP. Clearly, all feasible solutions to STS($3v$) exist in the feasible region of at least one of the integer programs whose first v components are fixed in this manner. A key point about this enumerative procedure is that it suffices only to enumerate all *non-isomorphic* solutions to STS(v) of value k, as summarized in Theorem 2.

Theorem 2 *Let $\mathcal{G}_{3v}$ be the symmetry group for STS(3v). For any two solutions to STS(v) x_1 and y_1, if there is a $\pi \in \mathrm{stab}(\mathcal{G}_{3v}, \{1, \ldots, v\})$ with $\pi(x_1) = y_1$, then π maps every feasible solution to the subproblem formed by x_v to a feasible solution to the subproblem formed by y_v. In other words, if the subproblem formed by x_v contains an optimal solution, so will the subproblem formed by y_v.*

Proof: Let $x = [x_1, x_2, x_3]$ be any solution to the subproblem generated by x_1. Permuting x by π gives $\pi(x) = [\pi(x_1), \pi(x_2), \pi(x_3)] = [y_1, \pi(x_2), \pi(x_3)]$. So, $\pi(x)$ is a feasible solution the subproblem generated by y_1. $\Diamond$

Theorem 2 remains true for any subgroup of the symmetry group $\mathcal{G}_{3v}$. Note that the stabilizer of the symmetry group of STS($3v$) is used, not the symmetry group of STS(v). However, in this case, it is easy to verify that the symmetry found in STS(v) is equivalent to $\mathrm{stab}(\mathcal{G}_{3v}, \{1, \ldots, v\})$,

so our procedure relies on enumerating solutions of STS(v) that are nonisomorphic with respect to its symmetry group (or a formulation group of STS(v)).

Enumerating all the non-isomorphic solutions to STS(v) of value at most k is computationally viable for values of k not significantly greater than z_v. The enumeration procedure was done with an extension of the branch-and-bound-based, isomorphism pruning algorithm of Margot [10]. In this variant, branching and pruning operations are performed until *all* variables are fixed. Nodes may not be pruned by integrality, only by bound, infeasibility, or isomorphism. All unpruned leaf nodes of the resulting tree are feasible solutions to the integer program whose objective value is k.

6 Solution to STS(135) and STS(243)

In this section, results of the computation proving the optimality of the cardinality 103 covering of STS(135) and the optimality of the cardinality 198 covering of STS(243) are presented.

6.1 **STS**(135)

The best solution known to STS(135) has value 103, and was reported by Odijk and van Maaren [12]. In the STS(135) branching tree shown in Figure 1, the nodes D, F, and G all have a lower bound obtained by solving the LP relaxation of value at least 103. To process nodes A, B, C, and E, the enumerative procedure outlined in Section 5 was undertaken. First, the flexible isomorphism pruning code of Ostrowski, Linderoth, and Margot [13] was run (on a 2.4GHz Intel Xeon processor) to enumerate all solutions to STS(45) of value at most 33. This required 662 seconds, 86428 nodes of the enumeration tree, and produced 2080 solutions. Next, the code was run again to enumerate all solutions to STS(45) of value 33 that also obeyed the appropriate permutations of $\sigma_{15}x \geq 10$. (These constraints enforce that each of the three blocks of 15 in STS(45) must have at least 10 ones). This required 4894 seconds, 693692 nodes, and produced 16,849 solutions. To solve the $2080 + 16849 = 18929$ integer programs necessary to prove the optimality of the solution of value 103 to STS(135), the orbital branching code of Ostrowski *et al.* [16] was used. An upper bound of value 103.1 was used to prune the branch and bound tree. Note that since the objective function may take only integer values for feasible solutions, a bound of 102 could have been used, but 103.1 was used to ensure that the methodology and software was able to reproduce the best known solution. The integer programs were solved on a pool of heterogenous workstations managed by the Condor resource management system at the University of Wisconsin. Table 1 contains a summary of the computation, listing for each node of the branch and bound tree of Figure 1, the number of non-isomorphic solutions generated (also the number of integer programs that need to be solved to process that node), the total number of nodes required, and the total CPU time required to solve all of the integer programs. All told, slightly more than ten million CPU seconds (or roughly 126 CPU days), were required

for the computation. However, since the integer programs were solved in parallel, the total wall clock time was 20 hours and 19 minutes. One solution of value 103 was found, and this solution was isomorphic to the one reported by Odjik and van Maaren. This computation establishes that the optimal solution to STS(135) has value 103.

Table 1: Statistics for STS(135) IP Computations

Node	**# Sol**	**Nodes**	**CPU Time**
A	1	10041	13m 37s
B	56	2738242	2d 2h 26m 8s
C	2023	40634479	30d 9h 41m 40s
D		$z_{root} = 105$	
E	16849	114346449	93d 16h 52m 0s
F		$z_{root} = 103$	
G		$z_{root} = 108$	

6.2 **STS**(243)

Odijk and van Maaren [12] have reported a solution of value 198 to the instance STS(243), and based on the structure of the solution, they conjecture the solution to be optimal. We are able to confirm that the optimal solution does have value 198. In the STS(243) branching tree shown in Figure 2, the nodes C, E, and H are all pruned by bound. To process nodes F and G, the orbital branching code of Ostrowski *et al.* was run on the integer programs *without* enumerating solutions to STS(81) and fixing variables in STS(243). A bound of 198.1 was used for pruning the branch and bound tree, and processing both nodes required just over two CPU hours, as detailed in Table 2.

To process nodes A, B, D, the enumerative procedure described in Section 5 was employed. All non-isomorphic solutions to STS(81) of value at most 63 were enumerated. This required 1021 seconds and 2420 nodes of the enumeration tree. Only 4 such solutions were found. To solve the 4 integer programs to process these nodes, the flexible isomorphism pruning code of Ostrowski, Linderoth, and Margot was used [13], and an upper bound of 198.1 was used for pruning. A summary of the computation is given in Table 2. All 4 integer programs were solved on a Intel Core 2 CPU, clocked at 2.4 GHz. The total CPU time required for the entire comutation, including enumeration was just over 51 hours.

Two solutions of value 198 were found, but they were both isomorphic to the solution reported by Odijk and van Maaren. Thus, the optimal solution to STS(243) has value 198. It is interesting that (likely due to the high quality of the solution of value 198), that significantly less CPU effort was required to solve STS(243) than the smaller instance STS(135).

Table 2: Statistics for STS(243) IP Computations

Node	# Sol	**Nodes**	**CPU Time**
A	1	33575	17h 47m 31s
B	1	46145	1d 4h 10m 25s
C		$z_{root} = 198$	
D	2	2428	2h 38m 58s
E		$z_{root} = 199$	
F	N/A	379	11m 27s
G	N/A	59	11m 51s
H		$z_{root} = 198$	

7 Conclusions

We have been able to prove the optimality of solutions for two new Steiner Triple Covering Problem, with systems of order 135 and 243. In both cases, the solution reported by Odijk and van Maaren was found to be an optimal solution. The instances were solved by a combination of symmetry-exploiting branching methodology, the enumeration of solutions to embedded subproblems, and parallel computing. We continue to attempt to solve the next larger instances in family, of sizes 405 and 729, and we suspect these instances will prove a challenge to the integer programming community for some time.

Acknowledgment

The work of the second author was supported by National Science Foundation (NSF) under grant DMI-0522796 and by the US Department of Energy under grant DE-FG02-09ER25869

References

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

[2] P. T. Darga, M. H. Liffiton, K. A. Sakallah, and I. L. Markov. Exploiting structure in symmetry generation for CNF. In *Proceedings of the 41st Design Automation Conference*, pages 530–534, 2004.

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

[4] 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.

[5] M. Hall. *Combinatorial Theory.* Blaisdell Company, 1967.

[6] 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.

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

[8] L. Liberti. Reformulations in mathematical programming: Symmetry. *Mathematical Programming*, 2009. To appear.

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

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

[11] B. D. McKay. *Nauty User's Guide (Version 1.5).* Australian National University, Canberra, 2002.

[12] 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.

[13] J. Ostrowski, J. Linderoth, and F. Margot. Flexible isomorphism pruning. Working paper.

[14] 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.

[15] J. Ostrowski, J. Linderoth, F. Rossi, and S. Smriglio. Constraint orbital branching. In *IPCO 2008: The Thirteenth Conference on Integer Programming and Combinatorial Optimization*, volume 5035 of *Lecture Notes in Computer Science*, pages 225–239. Springer, 2008.

[16] J. Ostrowski, J. Linderoth, F. Rossi, and S. Smriglio. Orbital branching. *Mathematical Programming*, 2009. To appear.