OPTIMA 83

Mathematical Optimization Society Newsletter

Steve Wright

MOS Chair's Column

June 28, 2010. This is the first issue of Optima under our society's new name, *Mathematical Optimization Society*. After a vigorous and enlightening discussion, members approved the change by the decisive margin of 457 to 127, with an additional 38 voters registering an abstention. We are working through the many practical issues associated with the change, including a new web address mathopt.org, which will come online soon, and a new logo. (Content of the web site mathprog.org has already been updated at its current URL.) The society will be moving ahead with renewed vigor under the new, more descriptive name.

Andrea Lodi (*Optima* editor) and Alberto Caprara (co-editor) have asked to resign their offices, following publication of the next issue. We are most grateful for their excellent work during these past three years in reviving *Optima*, following a moribund period. Under their leadership, the format was revamped and the publication arrangements and layout were changed. I am delighted to announce that the other co-editor, Katya Scheinberg, has agreed to become the new editor. I thank Katya for her service to date and her willingness to continue in this new capacity. If you have any comments on *Optima*, or ideas on format and feature articles, I am sure that Katya would be happy to hear from you!

This is my last column as chair (though I will remain as vice-chair for two more years). It has been a privilege to serve during these past three years. Much has happened during that time: a wonderful ISMP 2009 in Chicago, successful IPCO and ICCOPT conferences and establishment of ICCOPT as a regular conference of the society, inauguration of *Mathematical Programming Computation*, overhaul and modernization of the constitution and by-laws, redesign and reimplementation of the society's web site, revival of *Optima*, incorporation of our membership directory in the online Combined Membership Listing at www.ams.org/cml/, establishment of the Committee on Stochastic Programming (COSP) as a technical section of the society, and of course the change of name to MOS. Such ongoing activities as *Mathematical Programming, Series A and B* and Optimization Online remain in good shape, and the bugs in the online archive of the journals at Springer Online have been ironed out. Further challenges and opportunities remain. I hope you will work constructively with incoming chair Philippe Toint (whose term begins on the first anniversary of ISMP 2009) and the other officers and councilors on the initiatives they undertake during the next term.

There are many ways that you can advance the mission of MOS – refereeing, editing, and publishing in the journals; organizing the conferences; serving on prize committees; running for office in MOS; contributing to *Optima* and Optimization Online. I urge you all to find a way to contribute your time and energy! MOS serves the profession best when its members are engaged and active.

MOS's all-volunteer leadership model is one of its most appealing traits, but as our responsibilities grow, it puts more and more of a burden on some officers. The new leadership should certainly consider different models for the professional assistance that MOS receives, to increase the effectiveness of officers and improve our service and efficient use of revenues. Also on the table in the months ahead will be a possible change to our membership model, tying attendance at ISMP more closely to membership in MOS during the three-year term following each symposium.

I close by wishing Philippe and the incoming team all the best. The future for MOS and optimization is bright. How lucky we are to be involved in the profession and the society at such an exciting time for the field!

Contents of Issue 83 / July 2010

James Ostrowski, Jeff Linderoth, Fabrizio Rossi and Stefano Smriglio

Solving Steiner Triple Covering Problems

1 Introduction

> First, we supply data for two integer programming (set covering) problems which we believe are computationally hard. Our experience indicates that optimal solutions to these problems are very tedious to compute and verify, even though they have far fewer variables than numerous solved problems in the literature.

Thus begins Fulkerson, Nemhauser, and Trotter's 1973 paper [9], introducing two pure binary integer programs, `sts27` and `sts45`, of respectively 27 and 45 variables. These instances are *Steiner Triple Covering Problems*, and at the time, Fulkerson, Nemhauser, and Trotter were able to solve only the instance `sts27`. The instance `sts45`

was not solved until many years later. Despite nearly 40 years of exponential growth in processing power, combined with even greater advances in integer programming methodology, small instances in the sts family still pose exceptional challenges to state-of-the-art integer programming software.

The purpose of this note is to briefly describe the manner in which we were able to prove the optimality to the two smallest unsolved instances in the family, sts135 and sts243. The procedure relied on symmetry-exploiting branching methodologies, enumeration, and the use of powerful high-throughput "cloud" computing platforms. This note provides only some background on the instances and an overview of the approach. The interested reader may turn to [24] for further details.

1.1 Background

The sts integer programs are set covering problems generated from *Steiner Triple Systems*. To understand a Steiner Triple System, consider the *Kirkman School Girl Problem*, as it was proposed by Kirkman in 1850 in the *Lady's and Gentleman's Diary*.

> Fifteen young ladies in a school walk out three abreast for seven days in succession; it is required to arrange them daily, so that no two shall walk twice abreast.

A Steiner Triple System (STS) on a set, X, of n elements is a collection, $\mathcal{B}$, of 3-sets (*triples*) such that for any two elements x and y in X, the pair x and y appears in exactly one triple in $\mathcal{B}$. A solution to the Kirkman School Girl problem is a Steiner Triple System, as each walking group consists of 3 girls, while each pair of girls are in exactly one walking group together. The School Girl problem comes with the additional restriction that the collection of $|\mathcal{B}| = 35$ triples be divided into seven sets of five triples, one for each day, such that each element, or girl, appears exactly once in the set of five triples for that day.

Finding Steiner Triple Systems has interested mathematicians, not to mention ladies and gentlemen, for years. In 1847, Kirkman showed that a Steiner Triple System exists for a set X if and only if either $|X| = 6k + 1$ or $|X| = 6k + 3$, for some positive integer k. While there is only one non-isomorphic Steiner Triple System with size 7 and 9, the number grows considerably after that. Remarkably, there are more than 10^{10} many nonisomorphic triple systems with size 19.

A Steiner Triple System for a set X can be represented by a $(0,1)$-matrix A that has a column for every element in X and a row for every triple in $\mathcal{B}$. The matrix element $a_{ij} = 1$ if and only if element j is a member of triple i, and $a_{ij} = 0$ otherwise.

The 1-width of a $(0,1)$ matrix A is the minimum number of columns that can be selected from A such that all rows have at least one '1' in the selected columns. In other words, the 1-width of a matrix $A \in \{0,1\}^{m\times n}$ is the solution value of the following set covering problem:

$$w(A) \stackrel{\text{def}}{=} \min_{x\in\{0,1\}^n} \{1_n^T x \mid Ax \geq 1_m\}, \tag{1}$$

where 1_ℓ is an ℓ-dimensional vector of ones. Fulkerson and Ryser [8] studied the 1-width of incidence matrices of Steiner Triple Systems and were able to show that the 1-width $w(A_n) \geq (n-1)/2$ for all Steiner Triple incidence matrices A_n. They conjectured at the time an upper bound on the incidence width of $w(A_n) \leq 2n/3 - 1$, a bound that holds for all Steiner Triple Systems of size $n \leq 15$.

Later, Ryser constructed a 45 element system and speculated that it had a 1-width of value 30, which would contradict their earlier upper bound conjecture. This 45 element system is precisely the instance sts45 that Fulkerson, Nemhauser, and Trotter introduced in in [9]. Unable to solve sts45, Fulkerson, Nemhauser, and Trotter demonstrated a Steiner Triple System of size 27, whose incidence matrix A_{27} had $w(A_{27}) = 18$, thus disproving the conjecture. This is the sts27 instance.

The large 1-width Steiner Triple Systems of sizes 27 and 45 were created by using a special tripling procedure on triples of sizes 9 and 15 respectively. This tripling procedure induces significant symmetry.

If $A_n \in \{0,1\}^{m\times n}$ is incidence matrix of a Steiner Triple System (STS) of order n, then the tripled STS incidence matrix is

$$A_{3n} = \begin{pmatrix} A_n & & \\ & A_n & \\ & & A_n \\ I & I & I \\ D_1 & D_2 & D_3 \end{pmatrix}, \tag{2}$$

where the matrices $D_i \in \{0,1\}^{6m\times v}$ have exactly one "1" in every row.

The tripling procedure applied to sts27, sts45, and sts81 generates problems with size 81, 135, and 243, respectively. The instance sts81 was first solved by Mannino and Sassano [14] 15 years ago, and it remained, until recently, the largest problem instance in this family to be solved. The best known solutions to sts135 and sts243 have values 103 and 198 respectively, and were both reported by Odijk and van Maaren [20].

Fulkerson, Nemhauser, and Trotter give three explanations why the sts instances may be especially challenging for integer programming algorithms. First, the instances are highly symmetric. Unless the symmetry is recognized and exploited, branch-and-bound algorithms will require a prohibitive amount of enumeration. Second, the optimal bases of the linear programming (LP) relaxations of the instances have large determinants, making cutting plane methods based on group-theoretic methods very difficult. Third, the instances have a large integrality gap. Specifically, the optimal value of the LP relaxation for an instance of size n is $n/3$, while the optimal solution has value approximately $2n/3$. In the remainder of the section, we will briefly provide some intuition and evidence for the first two of these challenges.

1.2 Symmetry

To explain why branch-and-bound may require a prohibitive amount of enumeration in the presence of symmetry, Fulkerson, Nemhauser, and Trotter reference an interesting problem described by Jeroslow [10]. A variant of Jeroslow's problem may be stated as

$$\min_{x\in\{0,1\}^{n+1}} \{x_{n+1} \mid 2x_1 + 2x_2 + \ldots + 2x_n + x_{n+1} = 2k+1\} \quad \text{where } k \in \mathbb{Z}_+,\ k < \frac{n}{2}. \tag{3}$$

Jeroslow shows that branch-and-bound techniques that use linear programming relaxations require solving at least $2^{\frac{k}{2}}$ subproblems. To see why symmetry is a major confounding issue in branch and bound's ability to solve (3), consider the specific instance

$$\min_{x\in\{0,1\}^5} \{x_5 \mid 2x_1 + 2x_2 + 2x_3 + 2x_4 + x_5 = 3\}. \tag{4}$$

Figure 1 shows a branch-and-bound tree for solving (3). (Only subproblems for which the LP relaxation is feasible are included in the figure). The LP relaxations of all non-leaf nodes in the tree have optimal objective value 0. The optimal solution to each of the leaf subproblems is also an optimal solution to the ILP, and hence each has an objective value of 1.

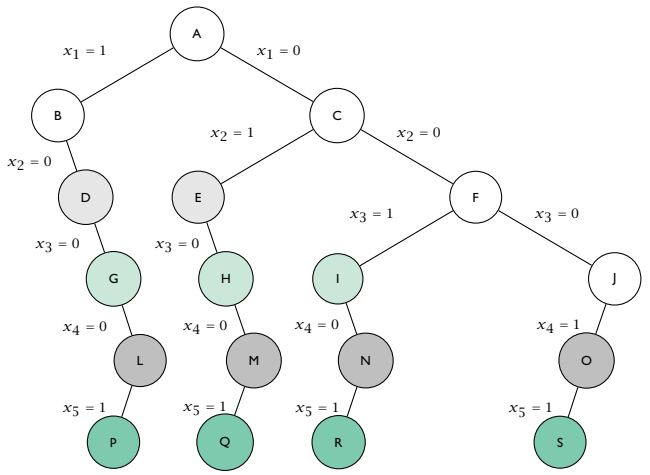

Figure 1. *Enumeration Tree for Jeroslow's problem*

Consider node D, the subproblem formed by fixing x_1 to zero and x_2 to one. We can rewrite the subproblem as

$$\min_{x \in \{0,1\}^5} \{x_5 \mid 2x_3 + 2x_4 + x_5 = 1\}. \tag{5}$$

Now consider node E, the subproblem formed by fixing x_1 to zero and x_2 to one. This subproblem can also be written as

$$\min_{x \in \{0,1\}^5} \{x_5 \mid 2x_3 + 2x_4 + x_5 = 1\}. \tag{6}$$

The subproblem at node D is *identical* to the subproblem at node E. Now it is easy to understand why traditional methods have difficulty with this type of problem, as neither node D nor E are pruned even though they are identical. It is obviously not necessary to consider *both* nodes D and E, but traditional methods provide no way of recognizing that the two nodes represent equivalent problems. Further inspection shows that there are many other nodes that represent identical subproblems (for instance, nodes G, H, and I represent the same subproblem, as do nodes L, M, N, and O).

In Jeroslow's problem, a solution is feasible (and optimal) if and only if x_{n+1} and k of the first n variables have value one (the rest zero). Thus, given a feasible solution, permuting any of the first n elements would yield a solution that is guaranteed to be feasible and has the same objective function value. In this sense, the first n variables are equivalent, inducing significant symmetry.

Effective symmetry-breaking techniques use symmetry to reduce the region that is searched. A first strategy consists in adding (non-valid) *symmetry-breaking inequalities* that keep in the feasible region a *representative solution* (i.e. an equivalent solution) for each feasible solution removed. For example, consider the set of constraints $x_1 \geq x_2 \geq \ldots \geq x_n$ in Jeroslow's problem. These constraints remove all but one solution from the feasible region. However, every solution removed is equivalent to the one remaining feasible solution. With these constraints in place, the integer program (IP) can be solved by a pure branch and bound algorithms with only 2 subproblems, compared to the minimum $2^{\frac{k}{2}}$ without symmetry-breaking.

For a general IP, the presence of large degrees of symmetry imply that there will be large classes of equivalent solutions. Because of these collections of equivalent solutions, it is likely that many of the subproblems solved in the branch-and-bound trees will be equivalent. Unless the equivalence is recognized, branch-and-bound trees may be thousands of times larger than what is necessary, making even the easiest problems impossible to solve.

In a series of papers, Margot [15, 16, 17] described an application of isomorphism-free backtracking (known in the combinatorics community) in the realm of IP. Isomorphism pruning examines the feasible region of each subproblem in the branch-and-bound tree and can recognize when a subproblem to be solved would be equivalent to another node in the search tree. A related idea, branching on *orbits* of variables, was explored by the authors of this paper [23].

To formally describe symmetry in IP, let Π^n be the set of all permutations of $\{1, 2, \ldots, n\}$. This set (along with the binary operation of composition) forms the *complete symmetric group*. Any subgroup of the complete symmetric group is a *permutation group*. For a permutation group Γ, the *orbit* of a point z under the action of the group is $\mathrm{orb}(\Gamma, z) \stackrel{\text{def}}{=} \{\pi(z) \mid \pi \in \Gamma\}$. The set $\mathrm{orb}(\Gamma, z)$ can be considered as the collection of "equivalent" elements to z, with respect to the permutations in Γ.

Let $A_n \in \{0,1\}^{m \times n}$ be the incidence matrix of a Steiner Triple System, and consider the collection of permutations

$$\Gamma = \{\pi \in \Pi^n \mid \exists \sigma \in \Pi^m \text{ such that } P_\sigma A_n P_\pi = A_n\},$$

where P_σ and P_π are the permutation matrices associated with σ and π respectively. If $\hat{x}$ is a feasible solution to the IP (1), then $\pi(\hat{x})$ is also a feasible solution of the same objective value. Thus, Γ is a group of permutations that identify equivalent solutions to (1). Solutions x and $\pi(x)$, for any $\pi \in \Gamma$, are called *isomorphic*.

Identifying the *symmetry group* Γ of (1) can be accomplished by using software such as `nauty` [19] or `saucy` [2] that computes the isomorphism group of a related graph. Liberti [12] and Ostrowski *et al.* [23, 12] describe the construction in more detail.

Applying symmetry-breaking techniques to the Jeroslow problem leads to branch-and-bound trees whose size is at most linear in n. Unfortunately, this dramatic speedup is not seen with the instances in the `sts` family, although symmetry breaking techniques can decrease computation times by several orders of magnitude. For example, the original solution by Mannino and Sassano of `sts81` did not use any symmetry-breaking techniques and required solving almost one billion subproblems (35 million for a processed problem). Margot's implementation of isomorphism pruning required solver fewer than 1,000 subproblems. However, symmetry breaking alone has not led to the solution of larger `sts` instances.

This has been only a cursory explanation of symmetry in integer programming. The reader interested in more details may see the the recent survey of Margot [18].

1.3 Cutting Planes

Cutting plane methods for general integer programs have been dramatically improved in the last decade and very effective practical implementations are now available in commercial solvers. Several types of general purpose cutting planes are known. Among them, Gomory *fractional* and *mixed integer* cuts, as well as *disjunctive cuts*, are often effective in closing the integrality gap, and help speed-up the solution of many integer programming problems. However, this is not the case of `sts` problems, as witnessed by several recent experimental findings which are summarized below.

Fischetti and Lodi [6] performed an exact optimization over the first *Chvàtal closure* of the natural $\{0,1\}$-LP formulation of `sts27` and `sts45`. This amounts to generating rank-1 Chvàtal-Gomory cuts until one such violated cut exists. Their experiments show that one gets no improvement with respect to the initial lower bound. The same negative result has been obtained in [1] by optimizing exactly over the *split closure*, that is, generating rank-1 *split cuts*, which are equivalent to Gomory mixed integer cuts.

Furthermore, Balas *et al.* [3] showed that a standard implementation of the Gomory's fractional cutting plane method (in which fractional Gomory cuts of any rank are generated) is able to close 50%

of the initial gap of sts15, but it fails to improve the initial bound when applied to sts27. Interestingly, they also demonstrated the interesting result that an advanced implementation of the method allowed to solve to optimality for the first time (with a pure cutting plane algorithm) sts15 and sts27.

As far as disjunctive cuts are concerned, they also prove to be ineffective when applied to Steiner triple covering problems. In fact, Fischetti and Lodi [7] showed that no improvement of the initial gap has been accomplished on sts27 with ten rounds of disjunctive cuts. This lack of improvement is quite striking if compared to what happens for most of the benchmark problems considered by the integer programming community.

In our initial attempts to solve sts135 and sts243, we thought an effective methodology may be to enumerate an isomorphism-free tree of subproblems, down to the point where the nodes did not contain any remaining symmetry. (The symmetry group Γ can be recomputed at each node and generally gets significantly smaller with each fixing of a variable). Then, we would generate a strong relaxation for each of the isomorphism-free subproblems via a closure procedure, and attempted to solve each of the strengthened subproblems.

We attempted this solution approach on sts135, running the modified orbital branching code [23] to write the nodal subproblem description to an MPS file and to fathom the node whenever a node with no symmetry remaining was reached. This generated a collection of 155 integer programs, and if we were able to solve each of these instances, then we could claim that we had solved sts135. We sent one of these instances to Anureet Saxena, the author of a code for computing the split closure of an integer program. Dr. Saxena kindly attempted to solve this instance for us by first tightening the relaxation via split cuts, and then giving the tightened instance to CPLEX. Generating split cuts improved the lower bound from the initial root relaxation value of 45 to the value 87.2 in 2.5 hours. CPLEX was then able to solve the strengthened instance in 14 days of CPU time, finding a solution of value 104, which was the optimal solution that could be found in that portion of the subtree. As we conjectured that the subproblem we gave to Dr. Saxena was one of the *easier* of the 155 instance we needed to solve, we abandoned this solution approach.

The lower bound after applying split cuts for the node given to Dr. Saxena was improved to 87.2. However, there is a trivial observation that can improve the lower bound at the *root node* to a value of 90. A quick examination of the matrix A_{3n} in (2) shows that the optimal cover must contain at least $3\times w(A_n)$ elements, as each of the three A_n must be independently covered. For example, since the first n columns of a solution to sts$(m = 3n)$ must also be a solution for sts(n), the inequality $\lambda_n^T x \geq w(A_n)$, with $\lambda_n \stackrel{\text{def}}{=} [1_n, 0_{m-n}]$ is valid for sts$(3n)$. Similarly, since the tripling construction process is recursive, the inequality $\lambda_n^T x \geq w(A_n)$ is valid for sts$(9n)$. This simple insight, and using this insight to generate strong branching disjunctions that exploit symmetry, is a specific case of the *constraint orbital branching* technique of the authors [22], which is briefly described in the next section.

2 Constraint Orbital Branching

Constraint orbital branching, 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 elements in the inequality's orbit ($\text{orb}(\Gamma, a)$), or the inequality $a^T x \geq b + 1$ holds for all elements in the orbit. This simple observation suggests the branching disjunction:

$$\left(a^T x \leq b\right) \vee \left(\bigwedge_{d \in \text{orb}(\Gamma, a)} d^T x \geq b + 1\right). \tag{7}$$

In the context of solving the sts135, branching on $(\lambda_{45}, w(A_{45}))$ implies that either there is a solution to sts135 having exactly 30 of the first 45 components of value 1, or for *each* of sets of indices $\{1, 45\}, \{46, 90\}, \{91, 135\}$ at least 31 components must be 1. This improves the lower bound on the "right" branch of the branching tree to $3 \times 31 = 93$.

To solve (1), we apply the disjunction (7) on constraints obtained from the optimal solution to smaller embedded sts instances. Specifically, we branch either on the constraints $\lambda_v^T x \leq k$, where $k \in \{w(A_v), w(A_v) + 1, \ldots\}$ or on the constraints $\lambda_{v/3}^T x \leq q$, where $q \in \{w(A_{v/3}), w(A_{v/3}) + 1, \ldots\}$.

Figure 2 shows the branching tree obtained by applying this methodology to sts135. The constraints added on the right branch of the tree (like $\mu^T x \geq 31 \ \forall \mu \in \text{orb}(\Gamma, \lambda)$) greatly improve the lower bound obtained by solving the linear programming relaxation. In fact, the nodes D, F, and G in Figure 2 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 3 shows the branching tree for solving sts243. In this case, the nodes C, E, and H are pruned by bound.

3 Solution by Enumeration

Based on the bounds obtained after applying the orbital branching disjunction, in order to solve sts135, we must only devise a procedure for processing the nodes A, B, C, and E of the branching tree of Figure 2. Likewise, to solve sts243, only the nodes A, B, D, F, and G of the branching tree of Figure 3 need be processed.

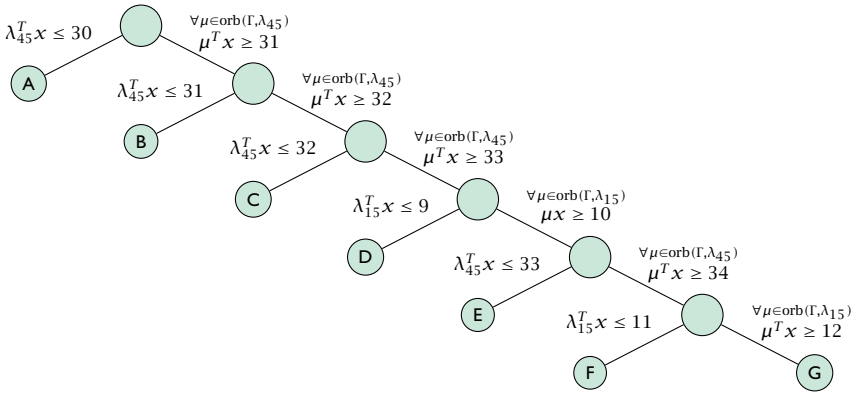

Figure 2. *Branching Tree for Solution of sts135*

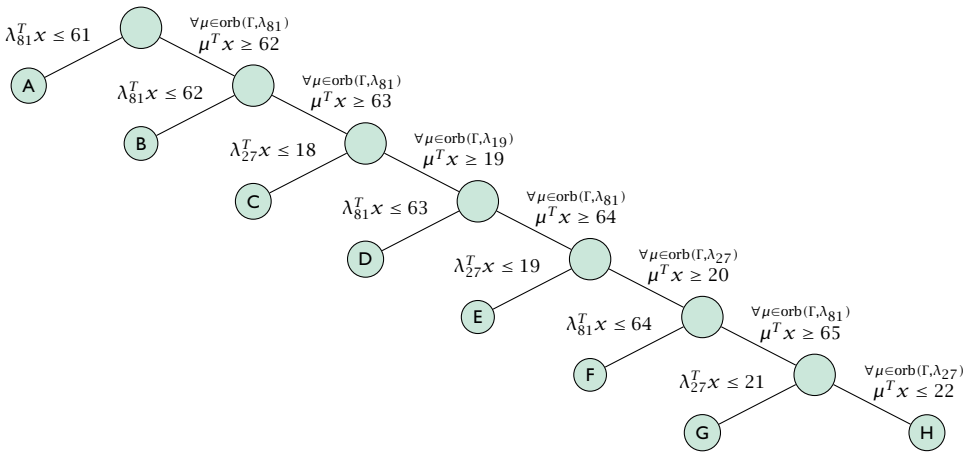

Figure 3. *Branching Tree for Solution of* `sts243`

One mechanism for processing the nodes would be to solve them with a black-box or commercial IP solver. Processing the nodes in this fashion did not appear to be computationally feasible. For instance, We ran CPLEX on the instance from node A in Figure 2 for many days, and there was still a quite significant optimality gap.

An alternative for processing the nodes is based on enumeration. By the construction of $\texttt{sts}(3v)$, for any feasible solution $x^* = [x^{1^*}, x^{2^*}, x^{3^*}]$ with $\lambda_v^T x^* \le k$, the first v components of the solution, $x^{1^*} \in \{0,1\}^v$, must be a feasible solution to $\texttt{sts}(v)$ with cardinality at most k. Note that in the branching trees used for solving `sts135` and `sts243`, the first branch enforces the constraint with $k = w(A_{45})$ and $k = w(A_{81})$, respectively. So, for x^* to be feasible in that branch, x^{1^*} must be an optimal solution to `sts45` or `sts81`. The branching tree continues by incrementing k in $\lambda_v^T x \le k$, or in some cases by branching on $\lambda_{v/3}^T x \le q$.

Based on this observation, one technique for processing a node in the tree defined by the branching inequality $\lambda_v^T x^* \le k$ is to enumerate *all* solutions $\{y^1, y^2, \ldots y^T\}$ to $\texttt{sts}(v)$ of value k. Then, for each solution $h = 1, \ldots, T$, fix the first v variables of $\texttt{sts}(3v)$ to y^h and solve the smaller IP. The best solution in the original node is the best solution found among the T smaller integer programs. This procedure is only likely to be effective if the smaller integer programs are significantly easier than the original integer program, and if the number of enumerated solutions T is not too large. Our preliminary computational experience with this method indicated that the reduced integer programs were often quite easily solved by commercial IP software or by our orbital branching code. Also, to reduce the size of T, we may again exploit symmetry, as it suffices only to enumerate all *non-isomorphic* solutions to $\texttt{sts}(v)$ of value k.

In our context, an advantage of this solution procedure – enumerating partial solutions, fixing variables based on these partial solutions, and solving the resulting subproblems – is that the smaller integer programs may be solved completely independently from one another. This makes the procedure an ideal candidate for the use of parallel computing. In general, with the advent of *massively* parallel processors, like the Jaguar system at Oak Ridge containing more than 200,000 cores, (structured) partial enumeration may give a mechanism for effectively using such high-performance modern computing architectures to solve integer programs. In our work, to provide the computational power for the solution of the `sts135` and `sts243` instances in this manner, we used CPU cycles from a distributed collection of workstations provided by the *Condor* software system [13]. Condor is a job scheduling software especially useful for completing jobs in a *high-throughput* manner; that is, jobs that require large amounts of processing over long periods of time, like the integer programs required to solve `sts135` and `sts243`. Condor has mechanisms to effectively harness idle CPU cycles from otherwise desktop workstations, effectively creating a "computational cloud" on which our integer programs were run. For a listing and description of the many features of Condor that make it a useful high-throughput computing toolkit, the reader is referred to the Condor web site (www.cs.wisc.edu/condor).

4 Solving sts135 and sts243

Tables 1 and 2 detail the historical progress made in finding high-quality feasible solutions for `sts135` and `sts243`.

Table 1. *Best Solutions Reported for* `sts135`

Author	Value	Date
Karmarkar *et al.* [11]	105	1991
Feo and Resende [5]	104	1995
Mannino and Sassano [14]	104	1995
Odijk and van Maaren [20]	103	1998

Table 2. *Best Solutions Reported for* `sts243`

Author	Value	Date
Feo and Resende [4]	204	1989
Feo and Resende [5]	203	1995
Mannino and Sassano [14]	202	1995
Odijk and van Maaren [20]	198	1998

4.1 sts135

In the `sts135` branching tree shown in Figure 2, the nodes D, F, and G all have a lower bound obtained by solving the LP relaxation of value at least 103, so only nodes A, B, C, and E need to be processed. The first step was to enumerate all solutions of `sts135` having solution value at most 33. This step was accomplished with a "flexible" variant of an isomorphism pruning code [21], requiring 662 seconds on a 2.4GHz Intel Xeon processor. Next, the code was run again to enumerate all solutions to `sts45` of value 33 that also obeyed the appropriate permutations of $\lambda_{15}^T x \ge 10$. (See node E in Figure 2). These constraints enforce that each of the three blocks of 15 variables in `sts45` must have at least 10 ones. This enumeration step required 4,894 seconds, 693,692 nodes, and produced 16,849 solutions. To solve the $2,080 + 16,849 = 18,929$ integer programs necessary to prove the optimality of the solution of value 103 to `sts135`, the orbital branching code of Ostrowski *et al.* [24] 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 heterogeneous workstations managed by Condor at the University of Wisconsin-Madison.

Table 3 contains a summary of the computation, listing for each node of the branch and bound tree of Figure 2, 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 evaluated in all the branch-and-bound trees, and the total CPU time required to solve all 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. The maximum CPU time required for any one individual IP instance was 7,139 seconds. One solution of value 103 was found, and this solution was isomorphic to the one reported by Odijk and van Maaren. This computation establishes that the optimal solution to sts135 has value 103.

Table 3. *Statistics for sts135 IP Computations*

Node	# Sol	Nodes	CPU Time
A	1	10,041	13m 37s
B	56	2,738,242	2d 2h 26m 8s
C	2,023	40,634,479	30d 9h 41m 40s
D		z_{root} = 105	
E	16,849	114,346,449	93d 16h 52m 0s
F		z_{root} = 103	
G		z_{root} = 108	

Table 4. *Statistics for sts243 IP Computations*

Node	# Sol	Nodes	CPU Time
A	1	33,575	17h 47m 31s
B	1	46,145	1d 4h 10m 25s
C		z_{root} = 198	
D	2	2,428	2h 38m 58s
E		z_{root} = 199	
F	N/A	379	11m 27s
G	N/A	59	11m 51s
H		z_{root} = 198	

4.2 sts243

Odijk and van Maaren [20] have reported a solution of value 198 to the instance sts243, 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 sts243 branching tree shown in Figure 3, the nodes C, E, and H are all pruned by bound. To process nodes F and G, we ran our orbital branching code on the integer programs *without* enumerating solutions to sts81 and fixing variables in sts243. A bound of 198.1 was used for pruning the branch and bound tree, and processing both nodes required just under 24 minutes, as detailed in Table 4.

To process nodes A, B, D, the enumerate-and-fix procedure described in Section 3 was employed. All non-isomorphic solutions to sts81 of value at most 63 were enumerated. This required 1,021 seconds and 2,420 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 [21], and an upper bound of 198.1 was used for pruning. A summary of the computation is given in Table 4. 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 computation, 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 sts243 has value 198. It is interesting that (likely due to the high quality of the solution of value 198) significantly less CPU effort was required to solve sts243 than the smaller instance sts135.

5 Solutions for sts729

Finding high quality solutions to Steiner Triple Covering Problems appears to also be quite difficult. As evidence, consider Tables 1 and 2 detailing the best solutions to sts135 and sts243 reported in the literature. Only the heuristic of Odijk and van Maaren, which is based on the GSAT heuristic of Selman *et al.* [25] was able to find the optimal solutions. However, the enumerate-and-fix procedure described in Section 3 used to prove the optimality of the Odijk and van Maaren's solutions to sts135 and sts243 was also able to (relatively) easily find the best known solutions as well. This suggests that the enumerate-and-fix procedure might also be effective as a heuristic.

Thus, we attempted to find a high quality solution to the sts729 instance obtained by tripling sts243. We ran CPLEX (v9) on the sts729 instance for more than two weeks, and the best solution found had value 653. This solution is likely far from optimal, as a solution of value 639 to sts729 can be obtained by taking the optimal solution of sts243 of value 198, doubling it, and then adding 1's in the last 243 positions.

To test the effectiveness of enumerate-and-fix as a heuristic procedure, we enumerated 35,861 unique solutions of sts243 having value at most 202. This enumeration process required over 10 million branch and bound nodes using the orbital branching code of Ostrowski *et al.* For 4,533 randomly chosen solutions, the first 243 variables of sts729 were fixed according to the solution, and CPLEX 12.1 was run for a maximum CPU time of 4 hours. All computations were done using the same Condor computational grid used for the solution of sts135 and sts243. The best solution obtained in this manner had value *619*, significantly better than could be found using CPLEX.

6 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 [20] 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 "high-throughput" computing.

We strongly believe that the use of challenge problems for the research community is an extremely important – both to measure the progress of the field and in generating "next generation" ideas that push the field in new directions. We have submitted instances sts405 and sts729 for consideration into MIPLIB2010 (http://miplib.zib.de/miplib2010/), and we hope these instances will spur the computational integer programming community to develop new solution methodologies.

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, 1513 University Avenue, Madison, WI 53706, USA. linderoth@wisc.edu
Fabrizio Rossi, Dipartimento di Informatica, Università di L'Aquila, Via Vetoio, I-67010 Coppito (AQ), Italy. fabrizio.rossi@univaq.it
Stefano Smriglio, Dipartimento di Informatica, Università di L'Aquila, Via Vetoio, I-67010 Coppito (AQ), Italy. stefano.smriglio@univaq.it

References

[1] E. Balas and A. Saxena. Optimizing over the split closure. *Math. Program. Ser. B*, 113(2):219–240, 2008.

[2] P. T. Darga, M. H. Liffiton, K. A. Sakallah, and I. L. Markov. Exploiting structure in symmetry detection for CNF. In *Design Automation Conference (DAC)*, pages 530–534, 2004.

[3] M. Fischetti, E. Balas and A. Zanette. Lexicography and degeneracy: can a pure cutting plane algorithm work? *Math. Program.*, to appear, 2010.

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

[5] T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[6] M. Fischetti and A. Lodi. Optimizing over the first chvátal closure. *Math. Program. Ser. B*, 110(1):3–20, 2007.

[7] M. Fischetti, A. Lodi, A. Tramontani. On the separation of disjunctive cuts. Mathematical Programming DOI: 10.1007/s10107-009-0300-y, 2010.

[8] D. Fulkerson and H. Ryser. Width sequences for special classes of (0-1)-matrices. *Canadian Journal of Mathematics*, 15:371–396, 1963.

[9] 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.

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

[11] 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.

[12] L. Liberti. Reformulations in mathematical programming: Symmetry. *Mathematical Programming*, 2010. To appear.

[13] M. J. Litzkow, M. Livny, and M. W. Mutka. Condor—A hunter of idle workstations. In *Proceedings of the 8th International Conference on Distributed Computing Systems*, pages 104–111, 1988.

[14] C. Mannino and A. Sassano. Solving hard set covering problems. *Operations Research Letters*, 18(1), July 13 1995.

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

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

[17] F. Margot. Symmetric ILP: Coloring and small integers. *Discrete Optimization*, 4:40–62, 2007.

[18] F. Margot. Symmetry in integer linear programming. In M. Jünger, T.M. Liebling, D. Naddef, G.L. Nemhauser, W.R. Pulleyblank, G. Reinelt, G. Rinaldi, and L.A. Wolsey, editors, *50 Years of Integer Programming 1958-2008*, pages 647–686. Springer-Verlag, 2009.

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

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

[21] J. Ostrowski, J. Linderoth, and F. Margot. Flexible isomorphism pruning. Unpublished Working Paper, 2010.

[22] 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.

[23] J. Ostrowski, J. Linderoth, F. Rossi, and S. Smriglio. Orbital branching. *Mathematical Programming*, 2010. To appear.

[24] J. Ostrowski, J. T. Linderoth, F. Rossi, and S. Smriglio. Solving large steiner triple covering problems. Technical Report 1663, Computer Sciences Department, University of Wisconsin-Madison, 2009.

[25] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI-92)*, pages 440–446, 1992.

Discussion Column

George Nemhauser

Vintage Steiner Triple System Paper Revisited

To accompany the paper by Ostrowski, Linderoth, Rossi, and Smriglio, Andrea Lodi has asked me to provide some background on the original Steiner Triple system paper by Ray Fulkerson, Les Trotter and me. This work was done at the School of Operations Research and Industrial Engineering at Cornell. Ray Fulkerson joined the department from the Rand Corporation in 1972 as a chaired professor. Ray had been part of the fantastic group of mathematicians along with Richard Bellman, George Dantzig, Les Ford, Phil Wolfe and others who pioneered the development of modern optimization including linear and nonlinear programming, dynamic programming and network flows. I believe that Ray was one of the last of these great men to leave Rand during the period when Rand was eliminating its basic research program in mathematics and operations research.

Soon after Ray arrived at Cornell, we had a discussion in which I told him that I was interested in computational integer programming. Ray told me that he and Herb Ryser had been studying a set covering problem regarding a conjecture about the 1-width of Steiner triple systems. He said, modest as always, that although he hadn't been keeping up with computational techniques in integer programming, he thought this was a particularly hard integer program for its size. I asked him to tell me about it. At this point he took a large piece of graph paper from his desk drawer that consisted of several 8x11 sheets pasted together. It contained a handwritten 330×45 0–1 matrix (only the ones were shown) with precisely 3 ones in each row. This is the A matrix for the set covering problem with an objective function of all ones, now known as Stein 45. The 1-width of the system is the optimal value.

I told Ray that I had a very bright graduate student by the name of Les Trotter and suggested that the three of us work on trying to solve Stein 45. As part of his Master's thesis, Les had developed a branch-and-bound algorithm with linear programming relaxations for solving binary integer programs. We also tried a pure cutting plane algorithm using Gomory cuts. Remember, in the mid 1970s, the idea, or at least the implementation, of combining cutting planes with search had not yet emerged.

I won't go into details here but we were not able to solve Stein 45, although we solved a smaller triple system Stein 27 (117 rows and 27 columns) that we constructed from Stein 45 and its solution resolved the conjecture. All of our computational success came from the branch-and-bound algorithm; the cutting plane algorithm could not solve a 15 element triple system with 35 rows and 15 columns. In our paper, we pointed out three main difficulties in solving these Stein instances: symmetry, very weak LP bounds, and very large determinants associated with the bases of the optimal solutions to the LP relaxations, which limit the effectiveness of Gomory cuts. Great progress has been made in the last two decades in tightening LP bounds and implementing Gomory cuts. However, symmetry remains a huge challenge. Recent works, such as the paper in this issue and earlier seminal work by Francois Margot, give new ideas for dealing with symmetry.

The Stein history has an interesting connection with my current university and department, Industrial and Systems Engineering at Georgia Tech. Besides my being a professor at Tech, Les Trotter did his Masters degree at Tech and the branch-and-bound algorithm we used came from his Masters thesis. Don Ratliff, now a colleague at Tech, was, I believe, the first person to solve Stein 45 about five years after we proposed it and soon after he arrived at Tech. Jeff Linderoth, one of the co-authors of the Stein paper in this issue, is a Tech PhD.

Finally, let me close with a story about a failed attempt to solve Stein 45. The optimal solution to the LP relaxations of a Stein instance has all of the variables equal to $\frac{1}{3}$ and hence an optimal value of $\frac{n}{3}$, where n is the number of variables. However, the optimal IP value is at least $\frac{n}{2} - 1$ and can even be larger than $\frac{2n}{3}$. Ryser speculated that the optimal value of Stein 45 was 30, and we believed it was correct. Moreover, we had a lower bound of 28, but we were reasonably sure that 28 was not feasible.

In those days, the use of integer programming algorithms that could find provably optimal solutions was extremely limited in practice. Practitioners and academics doing applied operations research primarily used heuristics and there was somewhat of a "cult" built up around people who had proprietary heuristics that supposedly performed much better than rigorous search or cutting plane algorithms. I happened to be at the University of Waterloo visiting Jack Edmonds when one of these heuristic gurus was also there. teaching a course on applied OR. I showed him Stein 45 and invited him to see how good a solution he could find because the best we could find was 30. A couple of hours later he told me that he could find a solution of value 27. When I told him that was impossible he said that surely he could find a solution of value 28. The next day I asked if he had found a solution of value 28 and he replied that he had lost interest since this was not a practical problem and there were no consulting fees to be had. So I suggested a small bet of $100 to make it interesting. He would win if he could find a solution of 28 in a week. He said he couldn't afford the time for such a small amount of money. So I said, how about $1000. He said that would be more interesting, but was unfair since he was a rich consultant and I was only a poor academic. Yes I said, I might be poor, but not dumb. So let's make it for $10,000 and you will have my certified check in an hour for that amount and you simply need to write personal check given your wealth. Suddenly, the braggadocio disappeared. Five years later Don Ratliff proved that 30 is indeed the optimal value.

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

George Nemhauser, Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta GA 30332 . george.nemhauser@isye.gatech.edu

Toulouse Global Optimization Workshop 2010 (TOGO10) Celebrating Pierre Hansen's 70th Birthday

31 August – 3 September 2010
ENSEEIHT, 2 rue Camichel, 31071 Toulouse, France

Global Optimization is a discipline aimed at finding global optima for hard nonconvex optimization problems. The workshop aims at bringing together researchers dealing with this topic. Depending on the number of submissions, we shall have either a single stream or two/three parallel streams. Accordingly, only a limited number of contributions will be accepted. A special issue of the Journal Of Global Optimization will contain (fully refereed) papers derived from works presented at the workshop (or on close topics).

Invited Speakers

Pierre Hansen (GERAD and HEC, Montréal, Canada), Emilio Carrizosa (Universidad de Sevilla, Spain), Charles Audet (École Polytechnique, Montréal, Canada), Pietro Belotti (Lehigh University, USA).

Registration Fee

300 EUR (200 EUR for students). This fee includes entry to all technical sessions, lunches, coffee breaks, welcome cocktail, a social dinner and one copy of the conference program and abstracts.

Information and Contact

http://www.lix.polytechnique.fr/togo10
gow.togo10@gmail.com

IMPRINT
Editor: Andrea Lodi, DEIS University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy. andrea.lodi@unibo.it ■ **Co-Editors:** Alberto Caprara, DEIS University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy. alberto.caprara@unibo.it ■ Katya Scheinberg, Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA. katyascheinberg@gmail.com ■ **Founding Editor:** Donald W. Hearn ■ Published by the Mathematical Optimization Society. ■ Design and typesetting by Christoph Eyrich, Berlin, Germany. optima@0x45.de ■ Printed by Oktoberdruck AG, Berlin, Germany.