

# **A Group Theoretic Tabu Search Algorithm for Set Covering Problems**

**G. Kinney**

**Graduate Program in Operations Research and Industrial Engineering  
The University of Texas at Austin  
Austin, Texas 78712  
Ph. 512-471-1336, Fax 512-232-1489, Email gkinney@mail.utexas.edu**

**J. W. Barnes**

**Graduate Program in Operations Research and Industrial Engineering  
The University of Texas at Austin  
Austin, Texas 78712  
Ph. 512-471-3083, Fax 512-232-1489, Email gkinney@mail.utexas.edu**

**B. Colletti**

**n  
n  
n  
n  
n**

## **Abstract**

We develop a Group Theoretic Tabu Search (GTTS) algorithm for solving the unicost Set Covering Problem (SCP). We solve a linear programming (LP) relaxation of the problem and use the LP optimum to construct a quality solution profile. We use group theory to partition the solution space into orbits based on this profile. We tested our algorithm on 65 benchmark problems and compared the results against the previous best known and solutions obtained by CPLEX 9.0. GTTS discovered 46 new best known solutions. GTTS converged *significantly* faster than CPLEX for all problem sets.

Keywords - Tabu Search, Group Theory, Set Covering Problem, LP Relaxation

# A Group Theoretic Tabu Search Algorithm for Set Covering Problems

## Abstract

We develop a Group Theoretic Tabu Search (GTTS) algorithm for solving the unicast Set Covering Problem (SCP). We solve a linear programming (LP) relaxation of the problem and use the LP optimum to construct a quality solution profile. We use group theory to partition the solution space into orbits based on this profile. We tested our algorithm on 65 benchmark problems and compared the results against the previous best known and solutions obtained by CPLEX 9.0. GTTS discovered 46 new best known solutions. GTTS converged *significantly* faster than CPLEX for all problem sets.

Keywords - Tabu Search, Group Theory, Set Covering Problem, LP Relaxation

## 1. Introduction

### 1.1. Problem Definition and Historical Background

The Set Covering problem (SCP) is a well-known combinatorial optimization problem. Given a 0-1 incidence matrix  $A$  with  $m$  rows and  $n$  columns, the problem is to select the minimum weight subset of columns while ensuring every row is covered. Formally, the problem is:

$$\begin{array}{ll} \text{Minimize} & z = \sum_{j=1}^n w_j x_j \\ \text{Subject to} & \sum_{j=1}^n a_{ij} x_j \geq 1 \quad i = 1, \dots, m \\ & x_j \in \{0, 1\} \quad j = 1, \dots, n \end{array} \quad (\text{P1})$$

If  $a_{ij} = 1$ , column  $j$  covers row  $i$  and  $w_j$  is the weight or cost of column  $j$ ,  $a_j$ . If  $a_j$  is selected to be in the subset,  $x_j$  is set to 1 and  $w_j$  is added to the cost of the solution. When  $w_j = 1$  for all  $j$ , the problem is the *unicost* SCP [28]. The *cardinality* of any P1 solution,  $\mathbf{x}$ , is the number of  $x_j=1$ , which, in the case of the unicast SCP, is the value of the objective function,  $z$ . The linear programming (LP) relaxation of P1 where the  $x_j$  can be non-integer is denoted by  $\overline{\text{P1}}$  which has optimal solution,  $\mathbf{x}_{LP}^*$ , and optimal objective function,  $z_{LP}^*$ .

The SCP has many practical applications including crew scheduling [1, 10, 15, 16], emergency facility location [19, 33] and political redistricting [21]. The SCP is known to be NP-Hard and both exact [1, 2, 6, 19, 21, 33] and heuristic [7, 9, 10, 11, 28] approaches have been proposed for it. Several of these have made extensive use of Lagrangian relaxation [1, 2, 6, 7, 10] and column dominance; neither are effective for unicast SCPs.

Grossmann and Wool [28] explore the performance of nine different heuristic algorithms on the unicast SCP. Most algorithms tested were LP rounding-based and construction-based approaches. A neural network algorithm was also included in the study. A randomized version of the greedy construction algorithm [11, 31] produced the best results.

## 1.2. Tabu Search

Tabu Search (TS) [22] is a local search method requiring a *starting solution*. A *move* modifies the current incumbent solution and all solutions that can be reached from the incumbent comprise the incumbent's *neighborhood*. The new solution is chosen from the neighborhood based on a merit function and the details of the algorithm. The new solution becomes the incumbent and the procedure repeats. The final solution is the best solution found during the search. Local search provides a distinct advantage in solving the unicast SCP by allowing the search to be based on the cardinality of the solution.

In TS, we prohibit recently visited solutions from being revisited for *tabu tenure* iterations. At each iteration the best *non-tabu* neighboring solution is selected. The tabu data structure allows escape from local optima to continue the search. TS has proven to be quite effective in solving complex optimization problems.

Reactive Tabu Search (RTS) was developed by Battiti and Tecchiolli [5]. In early TS implementations, the length of the tabu tenure was fixed or randomly chosen. In RTS, the tabu tenure is based on the occurrence of repeated visits to the same solutions. When the algorithm detects repetitions, the tabu tenure is increased to encourage the algorithm to diversify into a region of the solution space not yet explored.

## 1.3. Group Theory

Although not commonly acknowledged, many aspects of metaheuristics can be defined in terms of group theory. Until recently group theory has received little notice in heuristic operations research literature. This is unfortunate because group theory provides the necessary tools to define the structure of our solution space and take advantage of that structure during the search [3, 4, 12, 13, 14, 17, 18, 34]. Group theory was used frequently in classical methods in the late 60s through early 80s and has recently been resurrected [23, 24, 25, 26, 27, 32]. Gomory's fractional cuts are known to be elements of a finite abelian group [32]. Appendix A provides a brief introduction of group theory concepts and definitions.

Colletti [12] gives a comprehensive treatment of group theory in the context of metaheuristics. Using the Traveling Salesperson Problem (TSP) and TS, he classifies and defines current move definitions in terms of group theory and develops composite move strategies that would be difficult to generate using other methods. He provides efficient methods for escaping from chaotic attractors during the search and for generating search neighborhoods. This GTTS approach has been successfully applied to complex problems such as aerial fleet refueling [4, 34] and military theater distribution [17, 18].

## 1.4. LP Relaxation

The LP relaxation of an IP is used extensively in exact methods, but with the exception of the LP bound, is not commonly used in metaheuristics. For the SCP, the  $\overline{\text{P1}}$  solution provides valuable information for a direct search approach. The choice of the non-basic variables,  $\mathbf{x}_N$ , the basic variables,  $\mathbf{x}_B$ , and the reduced costs of the  $\mathbf{x}_N$ ,  $\overline{\mathbf{c}}_N$ , can provide valuable insight into the characteristics of high-quality IP solutions. This information can be used to create a *solution profile* which, using group theory, directs the search. Some of the ideas used are based on the insights of Gomory's Corner Polyhedra theory [25]; although, the approach is quite different.

## 2. Methodology

### 2.1. Overview

The GTTS approach is presented in detail later in the paper. However, for the purposes of clarity and ease of understanding, we provide a global overview. here.

First, CPLEX solves  $\overline{\text{P1}}$ . The associated optimal basic variables, the  $x_{Bi}$ , and the  $\overline{\mathbf{c}}_N$  are used to assign the  $x_j$  to *clusters*. A TS *starting solution*,  $\mathbf{x}_0$ , is generated by setting  $\mathbf{x}_B = \mathbf{0}$  and then *selecting* a subset of the  $x_{Bi}$  to raise to value 1 so that all rows are covered in P1. A RTS procedure is used to explore partitions of the solutions space, *orbits*, with cardinality between that of  $\mathbf{x}_0$  and  $\mathbf{x}_{LP}^*$ . A RTS procedure is also used to find *quality* orbits. The algorithm terminates when a GTTS solution value equals the P1 lower bound,  $z_{LB} = \lceil z_{LP}^* \rceil$  or when the allotted time has expired.

### 2.2. Clustering Variables Based on the LP Relaxation

The CPLEX solution to  $\overline{\text{P1}}$  details  $z_{LP}^*$ ,  $\mathbf{x}_B$ ,  $\overline{\mathbf{c}}_N$ , and which  $\mathbf{x}_N$  are at their upper (1) or lower (0) bounds. All  $x_{Bi}$  join the same cluster. The upper bound variables,  $x_{NUj}$ , and lower bound variables,  $x_{NLj}$ , cluster separately based on their reduced costs,  $\overline{c}_{NUj}$  and  $\overline{c}_{NLj}$ . For unicast  $\overline{\text{P1}}$ , all  $\overline{c}_{NLj} \leq 1$ . If  $\overline{c}_{NLj} = 1$ ,  $\mathbf{a}_j$  does not cover any of rows associated with the binding constraints at optimality. A  $\overline{c}_{NUj} = -1$  implies that if  $\mathbf{a}_j$  is removed from the solution, we must *select* at least two new columns to render the LP solution feasible. While it is possible for  $\overline{c}_{NUj}$  to be less than -1, it is quite unusual.

When we change the  $\mathbf{x}_B$  during the search, we are implicitly changing the basis and  $\overline{\mathbf{c}}_N$ . As a result, we cannot use  $\overline{\mathbf{c}}_N$  to explicitly determine which  $x_{Nj}$  will change values in the near optimal solutions. However, we can use  $\overline{\mathbf{c}}_N$  as a heuristic indicator of such changes. Therefore, we place the  $x_{Nj}$  in the same cluster if their  $\overline{c}_{Nj}$  are equal to the nearest 0.1 digit.

As illustrated in Figure 1, the clusters are created in the following order: upper bound variables, basic variables, lower bound variables.

We initially set all  $x_{NUj} = 1$ , all  $x_{NLj} = 0$  and all  $x_{Bi} = 0$  making the solution integer feasible. Integer feasibility is then *maintained* throughout the algorithm. Similar to the cardinality  $z$ , we define the cardinality,  $k_i$ , of a cluster  $i$  as the number of variables set to value 1 in the cluster. For example, a solution to a 22 variable problem could be

$$[1111|1101|10100|001|000000]$$

where  $z = 10$  and the five clusters have the cluster cardinality vector  $\mathbf{k} = (4 \ 3 \ 2 \ 1 \ 0)$ .

### 2.3. Partitioning the Solution Space into Orbits

The solution and cluster cardinality partition the solution space. By performing “swap moves” *within* the clusters, where the values of two variables in the same cluster are swapped, we preserve the current cardinality. This partitions the solution space first by total solution cardinality,  $z$ , and second by cluster cardinality,  $\mathbf{k}$ .

In group theoretic terms, the partitions are *orbits* and the swap moves are *group actions* (as discussed in Appendix A). Let  $n_i$  be the number of variables in cluster  $i$  and  $r$  be the number of clusters, then the group acting upon the set of binary vectors size  $n$  is  $S_{n_1} \times \dots \times S_{n_r}$ , where  $\sum_{i=1}^r n_i = n$ . Each within cluster swap move,  $(i \ j)$  with variable  $i$  and variable  $j$  assigned to the *same* cluster, is an element of this group. The group action is defined as moving the value of variable  $i$  to variable  $j$  and moving the value of variable  $j$  to variable  $i$ . For example,  $(2 \ 5)$  would swap the values of  $x_2$  and  $x_5$ .

The orbit can be uniquely identified by  $\mathbf{k}$ . Initially all orbits are *open*. A list of orbits visited is maintained and an orbit is *closed* to further search after it has been searched for MAX\_ORBIT\_ITER iterations or has been visited MAX\_ORBIT\_VISITS times. An orbit hash function [35],  $\Omega(\mathbf{k})$ , is used to facilitate access to orbit information. Orbit hash values are calculated by generating a random number,  $\tau_i$ , for each cluster  $i$ . The orbit’s hash value is  $\Omega(\mathbf{k}) = \sum_{i=1}^r \tau_i k_i$ . Orbits can be further distinguished by the number of *free ones* and *free zeros* in the orbit. Free ones (zeroes) are the number of ones (zeroes) in clusters that are not all ones (zeroes). For example, the solution below has 6 free ones and 6 free zeroes.

$$[1111|1101|10100|001|000000]$$

Any orbit with  $\sum_{i=1}^r k_i$  less than  $z_{LB}$  contains only infeasible solutions and may be immediately closed. Once a feasible solution is found with cardinality  $z_{UB}$ , all orbits with  $\sum_{i=1}^r k_i \geq z_{UB}$  are dominated and may be closed. It is probable that orbits with large  $k_i$  in the upper bound clusters and small  $k_i$  in the lower bound clusters will contain good

solutions. This partitioning scheme permits the concentration of search effort within these good orbits.

#### 2.4. Inter- and Intra-Orbit Neighborhoods

Performing a swap move in the same cluster moves the search to another solution within the same orbit, i.e., swaps comprise an *intra-orbit neighborhood*. *Inter-orbit neighborhoods* allow the search to move to other orbits.

Any move that changes  $k$  changes the orbit.  $k_i$  is increased or decreased by “toggling” the value of a single cluster  $i$  variable. Both  $k_i$  and  $k_j$  may be changed by swapping values of variables in clusters  $i$  and  $j$ . Each of these types of moves are used at different points in the GTTS algorithm. When searching inter-orbit neighborhoods we consider only open orbits.

For all neighborhoods, the best move is defined in terms of *deficit*, the number of unsatisfied rows, and *surplus*, the total amount that rows are over-satisfied. A row is over-satisfied when it is covered more than once. A row covered by 3 columns has a surplus of 2. Once a feasible solution is found we decrease  $z$  by unselecting a column, therefore GTTS is usually searching for feasible solutions. The best move yields the smallest deficit. Deficit ties are broken by largest surplus and surplus ties are broken by order of evaluation (lexicographically).

A *complete* swap neighborhood (which ignores cluster membership) would be  $O(n^2)$ . For intra-orbit swaps, this effort is reduced by considering swap pairs only if they are in the same cluster. Further, if  $k_i = n_i$  or  $k_i = 0$  no swaps are considered in cluster  $i$ . However, even with this reduction a complete swap neighborhood is too costly for large problems. Both the intra-orbit and inter-orbit swap neighborhoods are based on a *conditional select move*. First, moves that unselect a non-tabu selected column in the current solution are evaluated. The best of such moves is chosen. Given that chosen column will be unselected, we next find the best non-tabu unselected column to select.

A complete *select neighborhood* is  $O(n)$ . This neighborhood’s size may be reduced by incorporating a heuristic to help diversify the search. A column selection is considered only when the current solution is infeasible. We first find the row that has been unsatisfied for the longest number of iterations, then we only select from the columns that will satisfy that row.

Since ties in surplus are broken lexicographically, the order in which the clusters are evaluated affects the performance of the algorithm. For select neighborhoods, we first examine the upper bound clusters then the basic cluster and finally the lower bound clusters. This favors increasing the cardinality of the upper bound clusters over the rest. Unselect neighborhoods are examined in the opposite order of select neighborhoods favoring decreasing the cardinality of the lower bound clusters over the rest.

#### 2.5. Tabu Lists and Tabu Tenure

Two types of tabu structures are used in our algorithm. The first is a broad-gauge structure which tracks the last iteration a column was selected or unselected. A selected

or unselected column's status cannot be changed again for tabu tenure iterations. The tabu tenure may increase or decrease when the algorithm detects cycling.

The second type of tabu structure is a fine-gauge structure which tracks each *individual solution*, noting when it was last visited, and how many times it has been visited. For efficiency, another hash function [35],  $\varphi(x)$ , is used. A random number,  $\rho_j$ , is generated for each  $\alpha_j$ . The solution's hash value is  $\varphi(x) = \sum_{j=1}^n \rho_j x_j$ . Solutions can be further distinguished by their deficit and surplus.

Each orbit maintains its own tabu structures. In addition, a tabu structure is used during the RTS procedure to find quality orbits. The default tabu tenure for select moves is SELECT\_TENURE and the default tabu tenure for unselect moves is UNSELECT\_TENURE.

The second tabu structure is used to detect cycling and to control the tabu tenures. If a solution is repeated, the tabu tenures are increased by a multiplicative factor (\*1.618). If MIN\_NEW\_SOLS consecutive new solutions are visited, the tabu tenures are returned to their default values. If MAX\_REPEATED\_SOLS solutions are repeated MAX\_REPEATS times, the search is presumed to be in a chaotic attractor basin and an *escape* is achieved by departing the current orbit or by increasing the tabu tenure if we are searching for an orbit.

## 2.6. Finding a Starting Solution

To obtain a starting solution, all  $x_{N_j}$  are fixed at their values in  $\mathbf{x}_{LP}^*$  and the  $x_{B_i}$  are set to zero. As illustrated in the pseudocode of Figure 2, basic columns are then selected until  $z_{LB}$  is reached. If that solution is not feasible, intra-orbit swap moves are performed within the basic cluster, until no improving move is available. If the solution is still not feasible, a select move is performed within the basic cluster and the process repeats until feasibility is achieved. After feasibility is achieved, any identified redundant columns, columns whose unselection will not destroy feasibility, are removed.

## 2.7. Primary Search Strategy

A pseudo-code of the *primary search* strategy is presented in Figure 3. After the initial solution is obtained, the GTTS still focuses on the basic variables. A non-basic variable's status is modified only to achieve feasibility. Since the *duality gap*,  $z_{UB} - z_{LB}$ , is known,  $\lceil (z_{UB} - z_{LB})/2 \rceil$  columns are unselected from the basic cluster. Next, a RTS procedure, based on the intra-orbit swap neighborhood, is applied to the resulting orbit until (1) a feasible solution is found, (2) MAX\_ORBIT\_ITER iterations have been performed, (3) MAX\_NI\_ORBIT\_ITER iterations have been performed without improving the best orbit solution, or (4) the time limit is reached. If a feasible solution is obtained, the duality gap is recalculated,  $\lceil (z_{UB} - z_{LB})/2 \rceil$  columns are unselected from the basic cluster and the process repeats.

If a feasible solution is not found, the search departs the current orbit and the orbit's best solution is instantiated as the current incumbent solution. Next, the current

inter-orbit swap neighborhood is evaluated in pursuit of a move leading to a feasible solution. If that search yields feasibility, the duality gap is recalculated and the process repeats; if not, the current select neighborhood is evaluated. If feasibility is achieved, the duality gap is recalculated and the process repeats. If both neighborhoods fail to find a feasible solution, a basic cluster column is selected and the new orbit is searched. If  $z_{UB}$  is reached without achieving feasibility, the search is expanded by selecting an additional column from the basic cluster.

## 2.8. Expanding the Search

A pseudo-code of the *expanded search* strategy is presented in Figure 4. After exploring the orbits near the optimal LP solution, GTTS expands the search to other areas of the solution space. A RTS procedure, based on both inter-orbit and intra-orbit swap neighborhoods, is used to find a good region for exploration. This RTS procedure continues until either a feasible solution is found or the time limit is reached.

If a feasible solution is found, we unselect a single column (from any cluster) and explore the resulting orbit. If a feasible solution is found while exploring the orbit, we unselect a column again and the process repeats. If a feasible solution is not found while exploring the orbit, we begin another RTS procedure at the new  $z$ . The process ends when the time limit is reached or a solution equal to the P1 lower bound is found.

## 3. Computational Results

### 3.1. Test Cases

The benchmark problems solved were obtained from Beasley's OR-Library [8]. Problem sets 4-6 originally appeared in [2], problem sets A-D appeared in [6] and problem sets NRE-NRH appeared in [7]. All problems were randomly generated based on the strategy of [2]. All of these problems were generated as weighted SCPs. Grossman and Wool [28] solved all but NRG and NRH as unicast SCPs. One problem set, E from [6], is unicast but is not solved here due to its trivial size (all algorithms reach an optimal solution in 0 seconds).

### 3.2. Test Procedures

All tests were performed on Dell Precision 530 Workstations running SuSE Linux with two 1.8GHz Pentium Xeon processors and 1GB of RAM. The machines are multi-user platforms. An attempt was made to find machines that were not too busy, but as each problem was run for at least an hour, CPU usage surely fluctuated during processing. Each problem was solved using CPLEX 9.0 and GTTS. The time limits used were 7200 seconds for CPLEX and 3600 seconds for GTTS. The GTTS algorithm was coded in C.

The previously published best known solutions for these problems were published in Grossman and Wool [28]. They performed their tests on an IBM RS6000 model 370 workstation with 128MB of RAM. They also coded their algorithms in C.

### 3.3. CPLEX 9.0

The algorithms tested by Grossman and Wool [28] are unsophisticated by today's standards. To provide a more modern benchmark, we compare our results to CPLEX version 9.0. CPLEX uses a very sophisticated branch and cut algorithm to solve MIPs. The algorithm is further aided by two heuristics. The first attempts to create a feasible solution from the fractional solution at the node. The second attempts to improve the incumbent integer solution through a neighborhood search [30].

CPLEX was also used to solve  $P1$  for the GTTS algorithm. The dual simplex LP solver was used for the smaller problem sets, 4-6 and A-D. The sifting LP solver was used for the larger problem sets, NRE-NRH. The default settings were used for all other parameters.

### 3.4. Results

Tables 1 and 2 contain the results of our tests as well as the problem details and previous best known solutions. The best solution found for each problem is highlighted in bold. GTTS found the best solution on 59 of the 65 problems. It outperformed CPLEX on 47 of 65. It significantly outperformed CPLEX on the larger problem sets NRG and NRH. Curiously, neither CPLEX nor GTTS were able to do as well as R-Gr [28] on the problem sets with higher density, NRE and NRF. None of the solutions were proven to be optimal.

Since GTTS and CPLEX were executed on the same platform, we can make an accurate comparison of the convergence properties. Table 3 shows the average CPU seconds for each problem set for GTTS and CPLEX. Since the algorithms achieved different quality solutions, the time to the best common solution is used for the comparison whenever possible. When no common quality solution exists, the next lower solution for GTTS is used. GTTS converged significantly faster than CPLEX for all problem sets.

Table 4 compares the convergence properties of GTTS to R-Gr [28]. R-Gr was executed on a slower computer and for 5 of the problems GTTS did not reach the same quality solution. As expected R-Gr is faster than GTTS; however, GTTS significantly outperforms R-Gr in terms of overall solution quality. If R-Gr was executed for a longer period of time or for more iterations, it would be unlikely to improve upon the solutions it has already found.

## 4. Conclusion and Future Research

### 4.1. Conclusions

The use of variable clustering and group theory allowed our algorithm to intensify the search in the areas of the solution space believed to contain good solutions. The orbits kept the search contained in these areas and the clusters worked as an enhanced candidate list, reducing the total number of moves in the neighborhood while still retaining the "good" moves. These techniques proved very effective for the unicost SCP discovering 46 new best known solutions to the benchmark problems. The major

contributions of this research are the use of the LP relaxation to cluster the variables and the use of group theory to partition the solution space.

#### **4.2. Other Clustering Techniques**

For the unicast SCP problem, the LP solution provided a good profile of quality IP solutions and provided a good basis for variable clustering. This is not likely to be true for all problems types. In addition, for problems where solving the LP relaxation is not reasonable other methods of clustering will need to be developed. Heuristic methods, such as benefit/cost ratio for knapsack problems, may provide a more efficient and accurate clustering.

Partitioning the solution space into orbits restricts the movement of the search. If the partitioning is done effectively, this restriction can be useful to intensify the search effort in good regions of the solution space. Choosing a poor or arbitrary clustering scheme could trap the search in bad regions of the solution space.

#### **4.3. Enhancements**

Two possible future enhancements to the above algorithm are the use of cuts to improve the LP relaxation and clustering and the use of *composite* moves to improve the search. The LP relaxation provides us with a good solution profile. Would adding cuts at the beginning of the algorithm strengthen the LP relaxation and improve the profile? What type of cuts would be the most effective? We only implemented simple swap moves in our move neighborhoods. It might be worthwhile to create composite moves, swapping in one or more clusters simultaneously.

The research documented here was supported by a grant for the Air Force Office of Scientific research.

## Appendix A – Introduction to Group Theory

### A.1. Groups

Given a set of elements  $G$  and a binary operation  $\oplus$  then  $(G, \oplus)$  is a group iff

1.  $\forall a, b \in G \Rightarrow a \oplus b \in G$  (Closure)
2.  $a \oplus (b \oplus c) = (a \oplus b) \oplus c \quad \forall a, b, c \in G$  (Associativity)
3.  $\exists e \in G$  s.t.  $a \oplus e = a \quad \forall a \in G$  (Identity)
4.  $\exists a^{-1} \in G$  s.t.  $a \oplus a^{-1} = e \quad \forall a \in G$  (Inverse)

If the binary operation for a group is commutative then it is called an *abelian* group [20, 29]. The group operation is typically called multiplication and the result called a product regardless of the real context of the operation. The symbol for the operation is often discarded (i.e.  $a \oplus b = ab$ ).

A well known group is the group of permutations called the symmetric group on  $n$  letters  $S(n)$ . A permutation is represented by a 2 by  $n$  matrix corresponding to a 1-1 and onto mapping of the integers  $\{1, 2, \dots, n\}$  where the integer in the top row is replaced in the order by its image in the bottom row. For example, if  $n=6$  then

$$p = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 1 & 2 & 4 & 6 & 5 \end{pmatrix} \quad q = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 1 & 5 & 3 & 2 & 4 \end{pmatrix}$$

$p$  and  $q$  are permutations. The set of all permutations of degree  $n$  along with the binary operation of function composition form  $S(n)$  the symmetric group on  $n$  letters.

The product of  $p$  and  $q$  is  $p \oplus q = pq = q(p(x))$  is function composition meaning that permutation  $p$  is performed on  $x$  followed by  $q$ . Permutations are often written in cycle notation for example  $p = (132)(4)(56) = (132)(56)$ . Single letter cycles or 1-cycles map a letter onto itself and are typically not shown. Cycles with 2 characters or 2-cycles are called transpositions. The letters moved by  $p$  are denoted  $\text{move}(p)$ .

### A.2. Subgroups

Let  $G$  be a group with  $H \subseteq G$ . If  $H$  is also a group under the operation  $\oplus$  of  $G$  then  $H$  is called a subgroup of  $G$  denoted  $H \leq G$ .  $H$  must be closed under  $\oplus$ , contain the identity element  $e$ , and if  $a \in H$  then  $a^{-1} \in H \quad \forall a \in H$ . The associativity is inherited from  $G$ , as is commutativity if  $G$  is abelian [16, 20].

Since the subgroup is closed if  $a \in H$  then we must have  $aa \in H$  and  $aaa \in H$  and so on. Let  $a \in G$  then  $H = \{a^n \mid n \in \mathbb{Z}\}$  is a subgroup of  $G$  and is the smallest subgroup that contains  $a$  (by definition  $a^0 = e$ ). We say that  $H$  is the cyclic group generated by  $a$  and  $H = \langle a \rangle$  [20]. The group  $K = \langle a, b, c \rangle$  is the smallest subgroup that contains  $a$ ,  $b$ , and  $c$ .

### A.3. Direct Product of Groups

Let  $G_1, G_2, \dots, G_n$  be a collection of groups. Let  $G$  be the Cartesian product of the  $G_i$ . Define the binary operation in  $G$  as  $(a_1, a_2, \dots, a_n)(b_1, b_2, \dots, b_n) = (a_1b_1, a_2b_2, \dots, a_nb_n)$  or componentwise multiplication.  $G$  also forms a group called the external direct product of the groups  $G_i$ . The identity of  $G$  is  $(e_1, e_2, \dots, e_n)$  and  $(a_1, a_2, \dots, a_n)^{-1} = (a_1^{-1}, a_2^{-1}, \dots, a_n^{-1})$  [20, 29]. If all of the  $G_i$  are abelian then the external direct product of  $G_i$  is also abelian.

### A.4. Group Action

Given a group  $G$  and a set  $T$ , the group action of  $G$  on  $T$ , denoted by  ${}_G T$ , is a remapping of  $T$  (accomplished by the operation  $\wedge$ ) such that  $\forall g \in G, s \in T$  we have  $s \wedge g = t \in T$  [12]. Additionally  ${}_G T$  must have the following properties:

1.  $t \wedge e = t \quad \forall t \in T$  (where  $e$  is the identity of  $G$ )
2.  $(t \wedge g) \wedge h = t \wedge (gh) \quad \forall t \in T$  and  $g, h \in G$

A group action partitions  $T$  into disjoint cells called orbits. For group action  ${}_G T$ , the orbit of  $t \in T$  is  $\text{Orbit}(G, t) = \{t \wedge g \mid g \in G\}$ . If  $s, t \in T$  are in the same orbit then  $\exists a \in G$  such that  $s \wedge a = t$ . In general these partitions do not contain the same number of elements [12].

## References

- [1] E. Balas, M.C. Carrera, A Dynamic Subgradient-Based Branch and Bound Procedure for Set Covering, *Ops. Res.* 44 (1996) 875-890.
- [2] E. Balas, A. Ho, Set Covering Algorithms Using Cutting Planes, Heuristics, and Subgradient Optimization: a Computational Study, *Math. Prog.* 12 (1980) 37-60.
- [3] J.W. Barnes, B.W. Colletti, D.L. Neuhay, Using Group Theory and Transition Matrices to Study a Class of Metaheuristic Neighborhoods, *Euro. Jour. of Op. Res.* 138 (2002) 531-544.
- [4] J.W. Barnes, V.D. Wiley, J.T. Moore, D.M. Ryer, Solving the Aerial Fleet Refueling Problem using Group Theoretic Tabu Search, *Math. and Comp. Modeling* 39 (2004) 617-640.
- [5] R. Battiti, G. Tecchiolli, The Reactive Tabu Search. *ORSA Jour. on Comp.* 6/2 (1994) 126-140.
- [6] J.E. Beasley, An Algorithm for Set Covering Problem, *Euro. Jour. of Op. Res.* 31 (1987) 85-93.
- [7] J.E. Beasley, A Lagrangian Heuristic for Set Covering Problems, *Naval Res. Log.* 37 (1990) 151-164.
- [8] J.E. Beasley, OR-library: Distributing Test Problems by Electronic Mail, *Jour. of Op. Res. Soc.* 41/11 (1990) 1069-1072 (<http://mscmga.ms.ic.ac.uk/info.html>).
- [9] J.E. Beasley, P.C. Chu, A Genetic Algorithm for the Set Covering Problem, *Euro. Jour. of Op. Res.* 94 (1996) 392-404.
- [10] S. Ceria, P. Nobile, A. Sassano, A Lagrangian-Based Heuristic for Large-Scale Set Covering Problems, *Math. Prog.* 81 (1998) 215-228.
- [11] V. Chvátal, A Greedy Heuristic for the Set Covering Problem, *Math. of Ops. Res.* 4/3 (1979) 233-235.
- [12] B.W. Colletti, Group Theory and Metaheuristics, Ph.D. Dissertation, Department of Operations Research and Industrial Engineering, The University of Texas at Austin, 1999.
- [13] B.W. Colletti, J. W. Barnes, Linearity in the Traveling Salesman Problem, *Applied Math. Letters* 13 (2000) 27-32.
- [14] B.W. Colletti, J.W. Barnes, Local Search Structure in the Symmetric Traveling Salesperson Problem Under a General Class of Rearrangement Neighborhoods, *Applied Math. Letters* 14 (2001) 105-108.
- [15] T.E. Combs, A Combined Adaptive Tabu Search and Set Partitioning Approach for the Crew Scheduling Problem with an Air Tanker Crew Application, Ph.D. Dissertation, Department of Operational Sciences, Air Force Institute of Technology, 2002.
- [16] T.E. Combs, J.T. Moore, A Hybrid Tabu Search/Set Partitioning Approach to Tanker Crew Scheduling, *Military Ops. Res. Soc. Jour.* 9 (2004) 43-57.
- [17] J.R. Crino, A Group Theoretic Tabu Search Methodology for Solving Theater Distribution Vehicle Routing and Scheduling Problems, Ph.D. Dissertation, Department of Operational Sciences, Air Force Institute of Technology, 2002.
- [18] J.R. Crino, J.T. Moore, J.W. Barnes, W.P. Nanry, Solving the Theater Distribution and Scheduling Problem using Group Theoretic Tabu Search, *Math. and Comp. Modeling* 39 (2004) 599-616.

- [19] M.S. Daskin, E. Stern, A Hierarchical Objective Set Covering Model for EMS Vehicle Deployment, *Trans. Sci.* 15 (1981) 137-152.
- [20] J.B. Fraleigh, *A First Course in Abstract Algebra*, Addison-Wesley, Reading MA, 1976.
- [21] R. Garfinkel, Optimal Political Districting by Implicit Enumeration Techniques, *Management Science* 16/8 (1970) 495-508.
- [22] F. Glover, M. Laguna, *Tabu Search*, Kluwer Academic Publishers, Boston MA, 1997.
- [23] R.E. Gomory, On the Relation Between Integer and Non-Integer Solutions to Linear Programs, *Proc. of the Nat. Acad. of Sci.* 53 (1965) 260-265.
- [24] R.E. Gomory, Faces of Integer Polyhedra, *Proc. of the Nat. Acad. of Sci.* 57 (1967) 16-18.
- [25] R.E. Gomory, Some Polyhedra Related to Combinatorial Problems, *Linear Algebra and Its Apps.* 2 (1969) 451-558.
- [26] R.E. Gomory and E.L. Johnson, The Group Problem and Subadditive Functions, in: T.C. Hu, S.M. Robinson (eds.), *Mathematical Programming*, Academic Press, New York, 1973 pp. 157-184.
- [27] R.E. Gomory, E.L. Johnson, An Approach to Integer Programming, *Math. Prog. Ser. B* 96 (2003) 181.
- [28] T. Grossman, A. Wool, Computational Experience with Approximation Algorithms for the Set Covering Problem, *Euro. Jour. of Op. Res.* 101 (1997) 81-92.
- [29] I.N. Herstein, *Topics in Algebra*, Xerox College Publishing, Waltham MA, 1975.
- [30] ILOG, *ILOG CPLEX 9.0 User's Manual*, ILOG, Mountain View, CA, 2003.
- [31] D.S. Johnson, Approximation Algorithms for Combinatorial Problems, *Jour. of Comp. and Sys. Sci.* 9 (1974) 256-278.
- [32] E.L. Johnson, *Integer Programming – Facets, Subadditivity, and Duality for Group and Semi-Group Problems*, SIAM Publications, Philadelphia PA, 1980.
- [33] C. Toregas, R. Swain, C. Reville, L. Bergman, The Location of Emergency Service Facilities, *Ops. Res.* 19, (1971) 1363-1373.
- [34] V. Wiley, *The Aerial Fleet Refueling Problem*, Ph.D. Dissertation, Department of Operations Research and Industrial Engineering, The University of Texas at Austin, 2001.
- [35] D.L. Woodruff, E. Zemel, Hashing Vectors for Tabu Search, *Annals of Ops. Res.* 41 (1993) 123-137.

## Figures

Upper bound						Basic	Lower bound					
...	...	...	-0.2	-0.1	-0.0	0.0	0.0	0.1	0.2	...	...	1.0

Figure 1 – Variable Clustering

<p><b><i>Generate Feasible Starting Solution</i></b></p> <p>Select columns from the <i>basic</i> cluster until the <math>z_{LB}</math> is reached</p> <p>If feasible</p> <p>    Terminate with optimal solution</p> <p>Else {</p> <p>    While not feasible {</p> <p>        Select a column from the <i>basic</i> cluster</p> <p>        While improving move found</p> <p>            Execute intra-orbit swap move</p> <p>    }</p> <p>    Attempt to unselect redundant columns</p> <p>}</p> <p>Save the solution and <math>z_{UB}</math></p>
---

Figure 2 – Starting Solution Algorithm

**Primary Search**

```
Search Orbit with a RTS procedure
  Perform intra-orbit swap moves until a feasible solution is found or
  termination criteria is met
If feasible {
  Attempt to unselect redundant columns
  Save the solution and  $z_{UB}$ 
  If  $z_{UB} = z_{LB}$ 
    Terminate with optimal solution
  Else {
    Remove  $(z_{UB} - z_{LB})/2$  columns from the basic cluster
    Goto Primary Search
  }
}
Else {
  If feasible inter-orbit swap move found {
    Execute swap
    Goto Primary Search
  }
  Else If  $z + 1 < z_{UB}$  {
    If feasible select column move found {
      Select the column
      Goto Primary Search
    }
    Else {
      Select a column from the basic cluster
      Goto Primary Search
    }
  }
  Else
    Goto Expanded Search
}
```

Figure 3 – Basic Search Algorithm

### ***Expanded Search***

```
Search for a feasible solution with a RTS procedure
  Perform intra-orbit and inter-orbit swap moves until a feasible solution is
  found or termination criteria is met
If feasible {
  Attempt to unselect redundant columns
  Save the solution and  $z_{UB}$ 
  If  $z_{UB} = z_{LB}$ 
    Terminate with optimal solution
  Else {
    Unselect a column
    While feasible solution found {
      Search Orbit with Reactive Tabu Search
      Perform intra-orbit swap moves until a feasible
      solution is found or termination criteria is met
      If feasible {
        Attempt to unselect redundant columns
        Save the solution and  $z_{UB}$ 
        If  $z_{UB} = z_{LB}$ 
          Terminate with optimal solution
        Unselect a column
      }
    }
    Goto Expanded Search
  }
}
```

Figure 4 – Expanded Search Algorithm

## Tables

Problem	Number of Rows	Number of Columns	Density	Previous Best Known	CPLEX 9 Best (seconds)	GTTS Best (seconds)
4.1	200	1000	2%	41	<u>38</u> (11)	<u>38</u> (938)
4.2	200	1000	2%	38	<u>37</u> (42)	<u>37</u> (5)
4.3	200	1000	2%	40*	<u>38</u> (28)	<u>38</u> (1)
4.4	200	1000	2%	41	39 (851)	<u>38</u> (272)
4.5	200	1000	2%	40	39 (64)	<u>38</u> (23)
4.6	200	1000	2%	40	38 (50)	<u>37</u> (3)
4.7	200	1000	2%	41	39 (211)	<u>38</u> (413)
4.8	200	1000	2%	40	<u>38</u> (131)	<u>38</u> (6)
4.9	200	1000	2%	40	<u>38</u> (835)	<u>38</u> (35)
4.10	200	1000	2%	41	<u>38</u> (1772)	<u>38</u> (161)
<b>Problem Set 4 Average</b>				40.2	38.2 (399.5)	37.8 (185.7)
5.1	200	2000	2%	<u>35</u>	<u>35</u> (824)	<u>35</u> (5)
5.2	200	2000	2%	<u>35</u>	<u>35</u> (137)	<u>35</u> (6)
5.3	200	2000	2%	36	35 (373)	<u>34</u> (39)
5.4	200	2000	2%	36	35 (122)	<u>34</u> (1182)
5.5	200	2000	2%	36	35 (2257)	<u>34</u> (12)
5.6	200	2000	2%	36	36 (29)	<u>34</u> (989)
5.7	200	2000	2%	35*	35 (33)	<u>34</u> (75)
5.8	200	2000	2%	37	35 (85)	<u>34</u> (74)
5.9	200	2000	2%	36	36 (113)	<u>35</u> (6)
5.10	200	2000	2%	36	35 (4739)	<u>34</u> (1873)
<b>Problem Set 5 Average</b>				35.8	35.2 (871.2)	34.3 (426.1)
6.1	200	1000	5%	<u>21</u>	22 (3)	<u>21</u> (5)
6.2	200	1000	5%	<u>21</u>	<u>21</u> (2)	<u>21</u> (6)
6.3	200	1000	5%	<u>21</u> *	22 (3)	<u>21</u> (10)
6.4	200	1000	5%	22	22 (40)	<u>21</u> (4)
6.5	200	1000	5%	22	22 (2)	<u>21</u> (25)
<b>Problem Set 6 Average</b>				21.4	21.8 (10)	21 (10)

Table 1 – Results for Problem Sets 4-6  
\* - Solution found by Neural Network algorithm [28]

Problem	Number of Rows	Number of Columns	Density	Previous Best Known	CPLEX 9 Best (seconds)	GTTS Best (seconds)
A.1	300	3000	2%	40	42 (19)	<u>39</u> (337)
A.2	300	3000	2%	41	41 (21)	<u>39</u> (79)
A.3	300	3000	2%	40	40 (398)	<u>39</u> (179)
A.4	300	3000	2%	40	42 (19)	<u>38</u> (1715)
A.5	300	3000	2%	40	39 (4598)	<u>38</u> (771)
<b>Problem Set A Average</b>				40.2	40.8 (1011)	38.6 (616.2)
B.1	300	3000	5%	23	23 (826)	<u>22</u> (719)
B.2	300	3000	5%	<u>22</u>	23 (134)	<u>22</u> (17)
B.3	300	3000	5%	<u>22</u>	23 (12)	<u>22</u> (698)
B.4	300	3000	5%	23	23 (77)	<u>22</u> (1910)
B.5	300	3000	5%	23	23 (2422)	<u>22</u> (46)
<b>Problem Set B Average</b>				22.6	23 (694.2)	22 (678)
C.1	400	4000	2%	45	48 (3251)	<u>43</u> (1524)
C.2	400	4000	2%	45	46 (4488)	<u>44</u> (197)
C.3	400	4000	2%	45	49 (41)	<u>43</u> (1029)
C.4	400	4000	2%	46	47 (3994)	<u>43</u> (1325)
C.5	400	4000	2%	45	48 (6006)	<u>44</u> (149)
<b>Problem Set C Average</b>				45.2	47.6 (3556)	43.4 (844.8)
D.1	400	4000	5%	26	27 (2823)	<u>25</u> (395)
D.2	400	4000	5%	<u>25</u> <sup>^</sup>	26 (32)	<u>25</u> (1890)
D.3	400	4000	5%	<u>25</u>	26 (445)	<u>25</u> (91)
D.4	400	4000	5%	26	26 (178)	<u>25</u> (226)
D.5	400	4000	5%	26	26 (3489)	<u>25</u> (200)
<b>Problem Set D Average</b>				25.8	26.2 (1393.4)	25 (560.4)
NRE.1	500	5000	10%	<u>17</u>	18 (77)	18 (38)
NRE.2	500	5000	10%	<u>17</u>	18 (77)	18 (27)
NRE.3	500	5000	10%	<u>17</u>	18 (455)	18 (32)
NRE.4	500	5000	10%	<u>17</u>	18 (71)	<u>17</u> (54)
NRE.5	500	5000	10%	<u>17</u>	<u>17</u> (586)	18 (176)
<b>Problem Set NRE Average</b>				17	17.8 (255.2)	17.8 (65.4)
NRF.1	500	5000	20%	<u>10</u>	11 (90)	11 (29)
NRF.2	500	5000	20%	<u>11</u>	<u>11</u> (94)	<u>11</u> (39)
NRF.3	500	5000	20%	<u>11</u>	<u>11</u> (132)	<u>11</u> (30)
NRF.4	500	5000	20%	<u>11</u>	<u>11</u> (1083)	<u>11</u> (22)
NRF.5	500	5000	20%	11	<u>10</u> (482)	11 (23)
<b>Problem Set NRF Average</b>				10.8	10.8 (376.2)	11 (28.6)
NRG.1	1000	10000	2%	-	74 (582)	<u>63</u> (1089)
NRG.2	1000	10000	2%	-	76 (175)	<u>61</u> (3401)
NRG.3	1000	10000	2%	-	75 (6426)	<u>62</u> (901)
NRG.4	1000	10000	2%	-	74 (3723)	<u>63</u> (1045)
NRG.5	1000	10000	2%	-	73 (577)	<u>63</u> (406)
<b>Problem Set NRG Average</b>				-	74.4 (2296.6)	62.4 (1368.4)
NRH.1	1000	10000	5%	-	40 (756)	<u>35</u> (2008)
NRH.2	1000	10000	5%	-	39 (5716)	<u>36</u> (297)
NRH.3	1000	10000	5%	-	39 (1527)	<u>36</u> (968)
NRH.4	1000	10000	5%	-	40 (748)	<u>35</u> (940)
NRH.5	1000	10000	5%	-	37 (5734)	<u>36</u> (454)
<b>Problem Set NRH Average</b>				-	39 (2896.2)	35.6 (933.4)

Table 2 – Results for Problem Sets A-D and NRE-NRH  
<sup>^</sup> - Solution found by Alternating Greedy algorithm [28]

<b>Problem Set</b>	<b>CPLEX 9 avg (sec)</b>	<b>GTTS avg (sec)</b>
4	38.2 (399.5)	38.2 (116.6)
5	35.2 (871.2)	35.2 (4.4)
6	21.8 (10)	21.8 (2.8)
A	40.8 (1011)	40.8 (21.2)
B	23 (694.2)	23 (26.8)
C	47.6 (3556)	47.4 (7.2)
D	26.2 (1393.4)	26 (46.2)
NRE	18 (150.6)	18 (57.8)
NRF	11 (296.2)	11 (27)
NRG	74.4 (2296.6)	68.6 (114.4)
NRH	39 (2896.2)	38.2 (175)

Table 3 – CPLEX vs GTTS Solve Time

<b>Problem Set</b>	<b>R-Gr avg (sec)</b>	<b>GTTS avg (sec)</b>
4	40.3 (1.6)	39.6 (0.8)
5	35.9 (3)	35.8 (2.6)
6	21.6 (2)	21.6 (3.4)
A	40.2 (6)	40 (7.8)
B	22.6 (8)	22.6 (167.4)
C	45.2 (10)	45.2 (20.8)
D	25.8 (14.2)	25.8 (54.8)
NRE	17 (38)	17.8 (65.4)
NRF	10.8 (71.2)	11 (27)

Table 4 – R-Gr vs GTTS Solve Time