A Morphing Procedure to Supplement a Simulated Annealing Heuristic for Cost- and Coverage-Correlated Set-Covering Problems

Michael J. Brusco

Florida State University

Larry W. Jacobs

Northern Illinois University

Gary M. Thompson

Cornell University

Abstract

We report on the use of a morphing procedure in a simulated annealing (SA) heuristic developed for set-covering problems (SCPs). Morphing enables the replacement of columns in solution with similar but more effective columns (morphs). We developed this procedure to solve minimum cardinality set-covering problems (MCSCPs) containing columns which exhibit high degrees of coverage correlation, and weighted set-covering problems (WSCPs) that exhibit high degrees of both cost correlation and coverage correlation. Such correlation structures are contained in a wide variety of real-world problems including many scheduling, design, and location applications. In a large computational study, we found that the morphing procedure does not degrade the performance of an SA heuristic for SCPs with low degrees of cost and coverage correlation (given a reasonable amount of computation time), and that it improves the performance of an SA heuristic for problems with high degrees of such correlations.

Keywords: set covering, heuristics

1. Introduction

The set-covering problem (SCP) forms the basis for a variety of practical problems, including location of postal relax boxes [10], bus stops [24], day care facilities [29], emergency warning sirens [18] and emergency service facilities [43,48]; scheduling of airline flight crews [25,28,49], bus crews [41], a workforce [12,13,42], and naval vessels [11,23]; metallurgical grade assignment [44]; ingot selection [47]; data extraction [19]; and problems in the automotive industry [33]. A general statement of the SCP is

$$\text{Minimize} \quad \sum_{j \in C} c_j x_j \tag{1}$$

$$\text{subject to} \quad \sum_{j \in C} a_{ij} x_j \geq 1 \quad \text{for } i \in R, \tag{2}$$

$$x_j \in \{0,1\} \quad for\ j \in C, \tag{3}$$

where

R = the set of rows to be covered;

C = the set of columns for covering the rows;

a_{ij} = 1 if column j covers row i, 0 otherwise;

c_j = the cost of column j; and

x_j = 1 if column j is in solution, 0 otherwise

The SCP is well known to be computationally difficult [34], and substantial interest has centered on understanding the facets of the set-covering polytope [4, 5] and on finding effective procedures for solving problems of practical size. In terms of optimal solution approaches for

SCPs, early research centered on both branch-and bound [21, 36] and cutting-plane methods [1, 40]. However, the most recent optimal solution approaches with applicability to relatively large SCPs have been based on sub-gradient optimization and Lagrangian relaxation [2, 3, 6, 9, 22, 27]. For example, using a Lagrangian-based procedure that employed dual heuristics, Fisher and Kedia [22] optimally solved SCPs with up to 200 rows and 2000 columns. Similarly, Beasley and Jörnsten [9] optimally solved SCPs with up to 400 rows and 4000 columns. Harche and Thompson [26] recently developed a "column-subtraction" algorithm that also optimally solved some large SCPs.

Despite advancements in the development of optimal methods, very large SCPs generally necessitate the use of heuristic methods. Many heuristic procedures for solving SCPs are based on the use of greedy rules for adding columns, one at a time, to form a solution [3, 16, 44–46]. While these rules are very efficient, they are generally not capable of providing near-optimal solutions to large SCPs. In a comparative study that utilized several sets of very large SCPs, Beasley [7] found that a Lagrangian based heuristic he developed substantially outperformed several greedy methods [3, 44]. Subsequently, a variety of very effective methods based on Lagrangian relaxation with sub-gradient optimization have been developed [2, 14, 15, 37]. For example, Caprara et al. [14] proposed a Langrangian-based heuristic which won first prize in the FASTER competition. This competition, sponsored by Ferrovie dello Stato SpA and the Italian Operational Research Society, was intended to promote the development of procedures for very large SCPs. Jacobs and Brusco [30] showed that greedy rules could be much more competitive with Lagrangian-based heuristics if they were incorporated within the context of simulated annealing (SA) algorithms. Shortly thereafter, Beasley and Chu [8] developed a solution

approach based on genetic algorithms that is also very competitive with Lagrangian-based heuristics.

Many comparative studies for evaluating heuristic methods have used test suites of SCPs that were obtained by randomly generating the c_j and a_{ij} coefficients. For this study, we additionally investigated SCPs wherein the magnitude of the c_j coefficients is correlated with the number of non-zeros in their corresponding column vectors (cost correlation) *and* where the column vectors are correlated with one another (coverage correlation). Cost correlation, originally described by Rushmeier and Nemhauser [39], suggests that, in application, the cost of a column is likely to be directly (or near directly) related to the level of activity associated with that column. As such, cost correlation is only relevant to non-unicost or weighted set-covering problems (WSCPs) in which the c_j's are not strictly unit-valued.

In this paper, we introduce an additional type of correlation, coverage correlation, which we believe to be very prevalent in many practical WSCP's, as well as unicost or minimum cardinality set-covering problems (MCSCPs). This new correlation measure is based on the premise that different columns in an SCP often have similar non-zero elements. The greater the extent to which columns cover the same rows, the greater the coverage correlation. Using the earlier definitions, the coverage-correlation index, *CCI*, is

$$CCI = \frac{1}{m} \left\{ \sum_{j \in C} \left(\max_{\{k \in C, k \neq j\}} V_j^{\top} V_k \right) / V1 V_j \right\} \tag{4}$$

where

n = the number of rows;

m = the number of columns;

V1 = an *n*-element row vector (1, 1, …, 1); and

V_j = the n-element column vector $(a_{1j}, a_{2j}, \ldots, a_{nj})$.

The *CCI* is an average, across all columns, of the maximum commonality each column has with any other column in terms of the rows they cover. To illustrate the presence of high coverage correlation in practical SCPs, consider the MCSCP described by Torregas et al. [43], a location problem, which has *CCI* values of 0.868 and 0.9498 when the maximum acceptable distance for covering a location is presumed to be 10 and 15 miles, respectively. In contrast, problems that we randomly generated using the procedure of Balas and Ho [3] had *CCI* values of 0.224 to 0.356.

To illustrate how both cost and coverage correlation might be present in non-unicost or weighted set-covering problems (WSCPs), consider a generalized set-covering representation of a workforce shift scheduling problem which allows both full-time and part-time employees who work eight and four-hour shifts, respectively. Assuming hourly planning intervals, a column representing a full-time (part-time) shift would contain eight (four) ones. If the objective were to minimize total labor hours, the column cost for a full-time (part-time) shift would be eight (four). Thus, in this example, the column cost is perfectly correlated with the number of ones in the column. Further, let column *j* define an eight-hour shift beginning at 8 a.m. and column *k* define an eight-hour shift beginning at 9 a.m. The shifts associated with columns *j* and *k* would both cover all work hours (rows) from 9 a.m. until 4 p.m. Thus, $a_{ij} = a_{ik} = 1$ for seven hours (rows). Thus, columns *j* and *k* are said to be highly correlated since seven of their eight work periods are in the same hours (rows). Cost and coverage correlation are also present in other types of WSCPs. For example, a column may represent the cities to include in a particular aircraft schedule and it is likely that many pairs of columns may differ by only a single city. Moreover,

the cost of a particular schedule may be correlated to the number of cities included in the schedule.

This paper proposes that "morphing" provides an effective supplement for SA-based heuristics when applied to problems associated with high cost and coverage correlation. Morphing enables the replacement of columns in solution with similar, but more effective, columns (morphs). Specifically, a morph of a column *j* is a column that has a high degree of correlation with column *j*. The morphing procedure involves generating a *morph list* for each column that appears in solution and investigating interchanges of columns in solution with their morphs (that are not in solution). Based on our assessment of morphing and the type of SCPs for which we presume it will be most effective, we offer three hypotheses:

H1a: For SCPs that do not exhibit a high degree of coverage correlation, supplementing an SA-based heuristic with the morphing procedure will increase the time required to identify the best solution.

H1b: For SCPs that do not exhibit a high degree of coverage correlation, supplementing an SA-based heuristic with the morphing procedure will not increase solution costs, provided that reasonable amounts of computation time are available to compensate for the effect identified in hypothesis H1a. We define "reasonable" for this purpose to be CPU times that are competitive with those of other successful heuristics for WSCPs.

H2: For MCSCPs that do exhibit high degrees of coverage correlation, supplementing an SA-based heuristic with the morphing procedure will increase its effectiveness.

H3: For WSCPs that do exhibit high degrees of cost and coverage correlation, supplementing an SA-based heuristic with the morphing procedure will increase its effectiveness.

The rationale for **H1a** is that the construction of morph lists requires computational effort (and thus CPU time), even when the resultant lists are empty (as may be the case for problems with low coverage correlation). However, if reasonable time is available for the heuristic to check for the presence of morphs, no deterioration in the solution cost should be observed; and so we arrive at **H1b**. Hypothesis **H2** stipulates that morphing can significantly improve the performance of an SA-based heuristic when high degrees of coverage correlation are present in MCSCPs. Hypothesis **H3** stipulates that morphing can also improve heuristic performance when *both* high cost and coverage correlation are present in WSCPs.

The experimental results presented herein confirm that for SCPs that do not exhibit high degrees of cost or coverage correlation, our local-search heuristic based on the SA algorithm is competitive with one of the best Lagrangian heuristics, regardless of whether the morphing procedure is incorporated in the SA heuristic. Moreover, the results demonstrate that for both MCSCPs and WSCPs exhibiting high degrees of coverage correlation, the morphing procedure substantially improves the performance of the SA heuristic.

Section 2 describes the SA-based heuristic and shows how the morphing procedure can be used to supplement this heuristic. Section 3 presents the results of an experimental study used to evaluate the impact of the morphing procedure. Section 4 concludes the paper and offers suggestions for future research.

2. Local search heuristics based on the simulated annealing (SA) algorithm

Simulated annealing, originally developed as a model for statistical mechanics [38], was independently extended to combinatorial optimization by Kirkpatrick et al. [35] and Cerny [16]. In this context, SA incorporates a neighborhood-search strategy to seek improved solutions. The SA algorithm exhaustively searches the neighborhood of an incumbent solution and allows an inferior solution to be accepted, with a probability based on the degree of inferiority and the amount of time the algorithm has expended. It is this acceptance of inferior solutions that allows SA to backtrack from local optima. Although SA may form the basis for an optimal solution approach, prior research has suggested that enormous computational effort might be expended in finding optimal solutions to problems of practical interest [31, 32]. The research presented herein uses SA in a heuristic context, with the objective of developing procedures that rapidly converge to good (though not necessarily optimal) solutions using a modest amount of computational effort.

2.1. Simulated annealing heuristic – no morphing (SAHNM)

The SA-based heuristic used in this study, SAHNM (simulated annealing heuristic, no morphing) is a more efficient and effective version of the one described by Jacobs and Brusco [30] (hereafter, SAJB). The SAHNM structure is based on SAJB as provided below:

Set $X_B = \varnothing \; and \; Z(X_B) = \infty$
Do $irep = 1, 10$
Set *temp, mtime, iloop, rtime*
GENERATE an initial feasible solution, *X*, with cost *Z(X)*
Begin Loop
    Do *j* = 1, *iloop*
        SEARCH for a neighbor *X'* of *X*
        SET *Z(X')* = the cost of *X*, δ = *Z(X')-Z(X)*, and $\overline{\omega}$ = random [0, 1]
        If $\delta \leq 0 \; or \overline{\omega} \; \leq e^{-\delta/temp}$
            Set *X*=*X'* and *Z(X)* = *Z(X')*

If $Z(X') < Z(X_B)$, set X_B=X' and $Z(X_B) = Z(X')$
End if
If *rtime* > *mtime* continue with next replicate (i.e., increment *irep*)
End do
Set temp = *temp* * *cool*
End loop
End do
Return X_B and $Z(X_B)$

Note that we define our procedure to consist of 10 replications of a simulated annealing algorithm for each WSCP. An initial feasible, incumbent solution ***X*** is obtained using the **GENERATE** subroutine. To create the initial solution, this subroutine uses a variation of a common greedy heuristic suggested by Balas and Ho [3]. The initial temperature (*temp*), cooling parameter (*cool*), maximum CPU time for each replication (*mtime*), temperature length (*iloop*), and current run time (*rtime*) are subsequently specified. The algorithm performs *iloop* iterations at a particular temperature as long as the actual CPU time of the replication, *rtime*, is less than *mtime*.

The **SEARCH** routine, which is used to search the WSCP solution space, is structured as

PERTURB – adjusts the incumbent solution ***X*** by removing columns

CONSTRUCT – builds a trial feasible solution, ***X'***

REDUND – eliminates redundant columns

The **PERTURB** subroutine removes columns from the incumbent solution, resulting in a partial solution to the WSCP. Each time this subroutine is called, one of two rules is used to drop columns from solution. The random drop (RD) randomly selects a column to be dropped from solution. The "least uniqueness" drop (LUD) rule drops the column that provides the smallest number of unique row coverages. While using LUD exclusively tends to result in cycling, we

found that randomly selecting LUR (RD) for two-thirds (one-third) of the calls to **PERTURB** is superior to SAJB's exclusive use of RD.

The **CONSTRUCT** subroutine is applied to the partial solution to generate a neighboring trial solution ***X'***. Each time the **CONSTRUCT** subroutine is entered, one of two different greedy heuristics, similar to those described by Balas and Ho [3] and Chvatal [17], are used to rebuild the solution. The first greedy heuristic (GH1), begins with the random selection of an uncovered row *r*. For each column j that covers row *r*, the sum of uncovered rows that it covers is computed and divided by its cost. The column with the maximum ratio of unique coverage to column cost is appended to the solution. The second greedy heuristic (GH2) computes, for each column j that is not in the current partial solution, the sum of uncovered rows that it would cover, divided by its cost (c_j). The column with the maximum such ratio is appended to the solution. Obviously, the second greedy heuristic is more computationally intensive than the first. Nevertheless, we have found that the more comprehensive search provided by GH2 represents an enhancement to SAJB that has been quite effective in generating lower cost solutions.

The **REDUND** subroutine is subsequently applied to eliminate redundant (unnecessary) columns from ***X'***. Specifically, columns in solution are examined, in decreasing order of cost, to determine if they can be removed without creating an infeasible solution. Upon completion of *iloop* iterations, the temperature is reduced by the cooling parameter, *cool*.

2.2.Simulated annealing heuristic – with morphing (SAHWM)

As an SA procedure converges to a good solution, the elements in solution are generally good choices from the complete set of available alternatives. We posit that, in certain circumstances, morphing offers a fast means of searching for an even lower-cost solution. The

goal of morphing is similar to that of Genetic Algorithms (GAs), since both approaches attempt to retain the good characteristics of a solution while searching for further improvements. GAs do this by combining "parent" solutions into "children" solutions, with the intent that some of the "children" will retain the desirable, complementary features of their "parents". Morphing is a column-exchange heuristic that is implemented using a morph list for each element that appears in solution. A morph list is simply a list of the solution elements most similar to a specific solution element. By replacing a solution element with one of these similar but perhaps superior elements (given the current solution), morphing can find improved solutions.

SAHWM (simulated annealing heuristic, with morphing), is identical to SAHNM except that it includes the supplementary morphing procedure. Specifically, SAHWM calls the routine **MORPH** after the columns are dropped in **PERTURB**, and after every fourth column is added to a partial solution during **CONSTRUCT**. During **MORPH**, each column in solution is scanned to see if replacing it with one of its morphs improves the value of

$$\frac{\text{cost of the partial solution}}{\text{number of rows covered in the partial solution}}$$

When first examining a column, SAHWM creates a unique morph list for that column. This list contains the columns most similar to the column undergoing examination. For our problems, we define the morph index m_{kj} to be our measure of the correlation of column *j* to column k, and measure it as

(5)

$$m_{kj} = V_j^{\top} V_k / V1 V_k,$$

where

$\boldsymbol{V1}$ = an n-element row vector (1, 1, … , 1);

$\boldsymbol{V_j}$ = the n-element column vector $(a_{1j}, a_{2j}, \ldots, a_{nj})$, and

$\boldsymbol{V_k}$ = the n-element of column vector $(a_{1k}, a_{2k}, \ldots, a_{nk})$.

In early experimentation, we observed a deterioration of performance in SAHWM when using morphs that did not exhibit a reasonably high level of correlation with the column. For this study, each column's morph list is comprised of the columns with the highest morph indices. We set $m_{kj} \geq 0.6$ as a necessary criterion for selecting column *j* as a morph of column *k*, and limited the morph list to 30 columns. While the most appropriate values for morph list size and morph index cutoff are likely to be problem dependent, we arbitrarily set these values in this study. Finally, the morph lists are passed from one replicate to the next during the solution procedure.

Since SAHWM only develops morph lists for columns that appear in solution, it requires less computational effort than would the a priori generation of morph lists for *all* columns. Moreover, since SAHWM can pass the morph lists from replicate to replicate, substantially less computational effort is expended in developing morph lists on subsequent replicates. This significantly increases the time available for the evaluation of alternate solutions on later replicates – thus increasing the likelihood of finding an improved solution.

3. An experimental study

We designed an experimental study, consisting of three parts, to address our hypotheses. For each part of the experimental study, SAHNM and SAHWM were written in FORTRAN and implemented on a Pentium-based (100 MHz) microcomputer.

3.1.Experimental study – part 1: low cost- and coverage-correlated WSCPs

In part 1 of the experimental study, we compared SAHWM to SAHNM across 20 large WSCPs (sets E, F, G, and H) originally described by Beasley [7]. Since Beasley's [7] test problems were randomly generated using the procedure suggested by Balas and Ho [3], they exhibit neither high cost correlation nor high coverage correlation. The optimal solutions to these test problems are unknown. To the best of our knowledge, the best known solution costs for these 20 problems are reported by Caprara et al. [14], whose Lagrangian procedure (hereafter CFT) identified the best known solution for all 20 problems. The solution costs associated with SAHNM and SAHWM were compared to those of CFT. Since we defined "reasonable" CPU time to be that which is competitive with other heuristics for SCPs, we also compared the CPU times of SAHNM and SAHWM to those of CFT in order to ascertain their competitiveness. We used *mtime* = 1 minute (4 minutes) for the E and F (G and H) test problems for both SAHWM and SAHNM.

Tables 1 and 2 present our results for the SAHNM and SAHWM heuristics, respectively. Since the CFT procedure is based on a single run, whereas the SAHNM and SAHWM solutions represent the best solution found over ten replicates, we used a transformation (similar to that used in [14]) of the CPU times accorded to our procedure to provide a comparison. Specifically, we selected the first replicate k in which the best solution was found and reported the transformed solution time as: *mtime* $\times (k-1) +$ (average time to identify the best solution during a replicate). The CPU times reported for CFT in [14] are adjusted based on relative machine performances [20].

Tables 1 and 2 show that SAHNM and SAHWM, respectively, required 28.64 and 31.80 seconds, on average, to find their best solution in each replicate. Moreover, the average time

required by SAHWM was higher than that required by SAHNM in 16 of the 20 problems. The average transformed times for SAHNM and SAHWM were 106.64 and 115.78 seconds, respectively; and the transformed time required for SAHWM was higher than that of SAHNM in 17 problems. These results offer strong support for **H1a**; i.e., morphing increases the time required to find solutions in environments characterized by low degrees of coverage correlation.

A comparison of tables 1 and 2 reveals that, for each of the 20 test problems, SAHNM and SAHWM yielded identical solution costs. Moreover, the results clearly support hypothesis **H1b**; i.e. morphing does not degrade the performance of the SA heuristic in an environment characterized by low degrees of coverage correlation, provided that a reasonable amount of time is available for SAHWM to determine the morph lists.

The SAHNM and SAHWM solution costs match those of CFT for 18 of the 20 test problems (the costs for problems G2 and H1 are one unit higher than CFT for these problems). Overall, the average cost of SAHNM and SAHWN's solutions on these problems differs from that of the CFT solution costs by only 0.1%. Moreover, the CPU time comparisons to CFT indicate that our heuristic generally required less computational effort. Collectively these results suggest that both SAHNM and SAHWM are very competitive with the best known procedure for large WSCPs.

3.2.Experimental study – part 2: High coverage-correlated MCSCPs

For the second part of the study, we developed a set of 30 coverage-correlated test problems, varying on two dimensions. One dimension was the size of the problem. The smaller (larger) problems had 400 rows and 4000 columns (800 rows and 8000 columns). The secondproblem dimension was the density – the percentage of elements in the problem A-matrix

that are nonzero. We used densities of 2, 6, and 10 percent. We define a problem category as a set of problems that have the same number of rows and columns and the same density. For each of the six problem categories, we generated five MCSCPs, yielding a total of 30 test problems. These problems are much larger than the MCSCPs that have been previously examined in the literature.

Insert Table 1 Here

Insert Table 2 Here

We generated the coverage-correlated columns as follows. First, we generated columns in clusters of 20. For the first column in a cluster (assume that it is column *j*), we generated an n-element vector $\boldsymbol{Y} = (y_1, y_2, \ldots, y_n)$, where y_i is an independent, uniformly= distributed [0, 1) random variate. Next, we used the vector $\boldsymbol{Y}$ to provide the *n*-element coverage vector $(a_{1j}, a_{2j}, \ldots, a_{nj})$, using the relationship a_{ij}= 1, if $y_i \leq$ problem density (measured as a proportion); a_{ij}= 0, otherwise. Define the set *COV* as the rows that the first column in the cluster covers, i.e., $COV = \{i \in R \mid a_{ij} = 1\}$; }; further, set *RIC* equal to the number of elements in *COV*, i.e., $RIC = \sum_{i \in R} a_{ij}$. For each of the remaining 19 columns in the cluster, the coverage is

initially set to those rows in *COV*, but *k* randomly-selected rows from *COV* are replaced with k randomly-selected new rows, where $k = \lfloor 0.1 * RIC + 0.5 \rfloor$.

For both SAHNM and SAHWM, we used *mtime* = 2 minutes and *mtime* = 8 minutes for the 400 × 4000 and 800 × 8000 MCSCPs, respectively. A summary of the computational results is presented in table 3. This table reports, for each problem category, the number of problems for which SAHNM and SAHWM identified the best heuristic solution. SAHWM found the best heuristic solution for all 30 test problems, while SAHNM was able to identify the best solution for only 10 of the 30 problems. Interestingly, the 10 problems for which SAHNM matched the best solutions of SAHWM were associated with densities of 6% or 10%. In other words, SAHWM always provided a lower cost solution than SAHNM for the 2% density MCSCPs. Table 3 also reveals that SAHWM yielded an improvement in solution cost over SAHNM in all problem categories, with an average improvement of 2.56%. These results provide strong support for hypothesis **H2**.

3.3.Experimental study – part 3: High cost- and coverage-correlated WSCPs

In part 3 of the experimental study, we generated 60 new large WSCPs. Five test problems were generated for each of twelve problem categories. These categories were defined by all combinations of two cost-correlation structures, two levels of problem size, and three levels of problem density. Both cost-correlation structures exhibit a high degree of cost correlation. In the first structure, the perfect cost-correlation set-covering problem (PSCP), the cost of column j is set equal to the number of rows that it covers. In the second structure, the near-perfect cost-correlation set-covering problem (NSCP), the cost of column j is set equal to the number of rows it covers plus a uniformly distributed random integer from {–1, 0, +1}. In

terms of problem size, the smaller problems had 400 rows and 4000 columns, while the larger problems had 800 rows and 8000 columns. Problem densities were 2, 6, and 10 percent. The coverage-correlation structure for the WSCPs in part 3 was obtained in the same manner as for the MCSCPs in part 2.

Insert Table 3 Here

Insert Table 4 Here

We used *mtime* = 1 minute (4 minutes) for the 400 × 4000 (800 × 8000) test problems for both SAHNM and SAHWM. A summary of the computational results is presented in table 4. The solution costs associated with SAHWM were, on average, 2.04% lower than those obtained by SAHNM. More importantly, SAHWM yielded a lower solution cost than SAHNM for 56 of the 60 test problems, and a higher solution cost on only four problems. The mean percent solution cost savings associated with SAHWM was virtually the same for NSCPs (2.04%) and PSCPs (2.03%). Thus, hypothesis **H3** is well supported by the experimental results.

4. Conclusions and extensions

The morph implementation strategy is designed to supplement SA by improving an incumbent solution with a very modest incremental expenditure of computational effort. The morph lists are central to the success of this strategy. First, SAHWM generates morph lists only for columns that appear in solution, thereby eliminating the need to generate such lists for all columns. Second, SAHWM passes morph lists from one replicate to the next. Third, as SAHWM constructs partial solutions, the morphing procedure examines the morph lists of each column in solution, seeking a replacement that improves the solution. Finally, since only the columns that are likely to be effective replacements are stored in the morph lists, columns that are unlikely to be effective replacements will not be evaluated unnecessarily.

As expected (**H1a** and **H1b**), the morphing process neither improved nor worsened the cost of solutions for test problems with low degrees of coverage correlation (though it did take slightly longer to find the same solutions). Additionally, we found both SAHNM and SAHWM to be highly competitive with the best procedure (CFT [14]) known to us for these problems.

For the morphing process to yield improved solutions, the likelihood of an exchange of columns resulting in an improved solution must be fairly high. Such is the case for MCSCPs exhibiting high coverage correlation, and WSCPs exhibiting high cost and coverage correlation. For these test problems, the morphing process offered a substantial benefit. Specifically, SAHWM yielded a solution cost that was less than (less than or equal to) SAHNMs solution cost for 66.7% (100%) of the MCSCPs and 93.3% (93.3%) of the WSCPs (supporting **H2** and **H3**, respectively).

Morph-based improvement strategies should be particularly useful in heuristic procedures that alternate between feasibility and infeasibility. This conclusion is based on the observation

that morphing a partial (infeasible) solution – with the goal of improving the objective while reducing the extent of problem infeasibility – is superior to morphing only feasible solutions – with the goal of improving the objective. Intuitively, this result makes sense, since there are more degrees of freedom with a partial solution than with a solution that must retain feasibility. Since heuristic methods for other computationally difficult problems – such as bin-packing or assembly line balancing – often alternate between feasibility and infeasibility, we believe the morphing concept has broad applicability. The morphing strategy is designed to take advantage of similarities in solution elements and the concept is likely to be most useful in situations where such similarities exist.

References

E. Balas, Cutting planes from conditional bounds: A new approach to set covering, Mathematical Programming Study 12(1980)19–36.

E. Balas and M.C. Carrera, A dynamic sub-gradient-based branch and bound procedure for set covering, Operations Research 44(1996)875-890.

E. Balas and A. Ho, Set covering algorithms using cutting planes, heuristics, and sub-gradient optimization: A computational study, Mathematical Programming Study 12(1980)37–60.

E. Balas and S.M. Ng, On the set covering polytope: I. All the facets with coefficients in {0, 1, 2}, Mathematical Programming 43(1989)57–69.

E. Balas and S.M. Ng, On the set covering polytope: II. Lifting all the facets with coefficients in {0, 1, 2}, Mathematical Programming 45(1989)1–20.

J.E. Beasley, An algorithm for set-covering problems, European Journal of Operational Research 31(1987)85-93.

J.E. Beasley, A Lagrangian heuristic for set covering problems, Naval Research Logistics 37(1990) 151–164.

J.E. Beasley and P.C. Chu, A genetic algorithm for the set covering problem, European Journal of Operational Research 94(1996)392-404.

J.E. Beasley and K. Jörnsten, Enhancing an algorithm for set covering problems, European Journal of Operational Research 58(1992)293–300.

J. Bouliane and G. Laporte, Locating postal relay boxes using a set covering algorithm, American Journal of Mathematical and Management Sciences 12(1992)65–74.

G.B. Brown, G.W. Graves and D. Ronen, Scheduling ocean transportation of crude oil, Management Science 33(1987)335–346.

M.J. Brusco and L.W. Jacobs, A simulated annealing approach to the cyclic staff-scheduling problem, Naval Research Logistics 40(1993)69–84.

M.J. Brusco, L.W. Jacobs, R.J. Bongiorno, D.V. Lyons and B. Tang, Improving personnel scheduling at airline stations, Operations Research 43(1995)741–751.

A. Caprara, M. Fischetti and P. Toth, A heuristic algorithm for the set covering problem, *Lecture Notes in Computer Science*, vol. 1084, 1996, pp. 72–84.

S. Ceria, P. Nobili and A. Sassano, A Lagrangian-based heuristic for large-scale set covering problems, Working Paper, University of Roma La Sapienza, Italy, 1995.

V. Cerny, Thermodynamical approach to the traveling salesman problem, Journal of Optimization Theory and Applications 45(1985)41-51.

V. Chvatal, A greedy heuristic for the set covering problem, Mathematics of Operations Research 4(1979) 233–235.

J. Current and M. O'Kelly, Locating emergency warning sirens, Decision Sciences 23(1992) 221- 234.

R.H. Day, On optimal extracting from a multiple file data storage system: An application of integer programming, Operations Research 13(1965)482–494.

J. Dongarra, *Linpack Benchmark*, URL-http://ap01.physik.unigreifswald.de/~ftp/bench/linpack.html, October 2, 1995.

J. Etcheberry, The set covering problem: A new implicit enumeration algorithm, Operations Research 25(1977)760–772.

M.L. Fisher and P. Kedia, Optimal solution of set covering partitioning problems, Management Science 36(1990)674–688.

M.L. Fisher and M.B. Rosenwein, An interactive optimization system for bulk cargo ship scheduling, Naval Research Logistics 36(1989)27–42.

J. Gleason, A set covering approach to bus stop location, Omega 3(1975)605-608.

G.W. Graves, R.D. McBride, I. Gershkoff, D. Anderson and D. Mahidhara, Flight crew scheduling, Management Science 39(1993)736–745.

F. Harche and G.L. Thompson, The column subtraction algorithm: An exact method for solving weighted set covering, packing, and partitioning problems, Computers and Operations Research 21(1994)698–706.

A.M. Hey and N. Christofides, Algorithms for the set covering problem using graph theory, Working Paper, Imperial College, London, England, 1993.

K.L. Hoffman and M. Padberg, Solving airline crew scheduling problems by branch-and-cut, Management Science 39(1993)657–682.

J. Holmes, F.B. Williams and L.A. Brown, Facility location under a maximum travel restriction: An example using day care facilities, Geographical Analysis 4(1972)258–266.

L.W. Jacobs and M.J. Brusco, Note: A local-search heuristic for large set-covering problems, Naval Research Logistics 42(1995)1129–1140.

D.S. Johnson, C.R. Aragon, L.A. McGeoch and C. Schevon, Optimization by simulated annealing: An experimental evaluation; Part I, Graph partitioning, Operations Research 37(1989)865–892.

D.S. Johnson, C.R. Aragon, L.A. McGeoch and C. Schevon, Optimization by simulated annealing: An experimental evaluation; Part II, Graph coloring by number partitioning, Operations Research 39(1991)378–406.

D.J. Jones, M.A. Beltramo and M.S. Daskin, A metaheuristic for large set-covering problems, Working Paper, General Motors R&D Center, Warren, MI, 1994.

R.M. Karp, in: *Complexity of Computer Computations*, eds. R.E. Miller and J.W. Thatcher, Plenum Press, New York, 1972.

S. Kirkpatrick, C.D. Gelatt, Jr. and M.P. Vecchi, Optimization by simulated annealing, Science 220(1983)671–683.

C.E. Lemke, H.M. Salkin and K. Spielberg, Set covering by single branch enumeration with linear programming sub-problems, Operations Research 19(1971)998–1022.

L.A.N. Lorena and F.B. Lopes, A surrogate heuristic for set covering problems, European Journal of Operational Research 79(1994)138–150.

N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A. Teller and E. Teller, Equation of state calculations by fast computing machines, Journal of Chemical Physics 21(1953)1087–1092.

R.A. Rushmeier and G.L. Nemhauser, Experiments with parallel branch-and bound algorithms for the set covering problem, Operations Research Letters 13(1993)277–286.

H.M. Salkin and R.D. Koncal, Set covering by an all integer algorithm: Computational experience, Journal of the Association for Computing Machinery 31(1973)336–345.

B.M. Smith, IMPACS – A bus crew scheduling system using integer programming, Mathematical Programming 42(1988)181–187.

G. Thompson, A simulated-annealing heuristic for shift scheduling using non-continuously available employees, Computers and Operations Research 23(1996)275-288.

C. Torregas, R. Swain, C. ReVelle and L. Bergman, The location of emergency service facilities, Operations Research 19(1971)1363–1373.

F.J. Vasko and G.R. Wilson, Hybrid heuristics for minimum-cardinality set covering problems, Naval Research Logistics Quarterly 33(1986)241–249.

F.J. Vasko and G.R. Wilson, An efficient heuristic for large set covering problems, Naval Research Logistics Quarterly 31(1984)163–171.

F.J. Vasko and F.E. Wolf, Solving large set-covering problems on a personal computer, Computers and Operations Research 15(1988)115–121.

F.J. Vasko, F.E. Wolf and K.L. Stott, Optimal selection of ingot sizes via set covering, Operations Research 35(1987)346–353.

W. Walker, Application of the set covering problem to the assignment of ladder trucks to fire houses, Operations Research 22(1974)275–277.

D. Wedelin, An algorithm for large scale 0–1 integer programming with application to airline crew scheduling, Annals of Operations Research 57(1995)283–295.

Table 1

SAHNM results for the low cost-correlated, low coverage-correlated WSCPs.

Problem	Replicate 1	2	3	4	5	6	7	8	9	10	SAHNM solution	Average time[a]	Adjusted time[b]	CFT solution	CFT time[c]
E1	29	29	29	29	29	29	29	29	29	29	29	0.36	0.36	29	11.30
E2	30	30	30	30	30	30	30	30	30	30	30	10.29	10.29	30	177.39
E3	27	27	27	27	27	27	27	27	27	27	27	13.51	13.51	27	40.96
E4	28	28	28	28	28	28	28	28	28	28	28	1.75	1.75	28	11.43
E5	28	28	28	28	28	28	28	28	28	28	28	0.62	0.62	28	15.91
F1	14	14	14	14	14	14	14	14	14	14	14	1.05	1.05	14	14.43
F2	15	15	15	15	15	15	15	15	15	15	15	0.82	0.82	15	13.57
F3	14	14	14	14	14	14	14	14	14	14	14	3.13	3.13	14	108.04
F4	14	14	14	14	14	14	14	14	14	14	14	1.75	1.75	14	13.48
F5	14	13	13	14	13	14	13	13	13	13	13	18.55	18.55	13	87.43
G1	176	178	178	176	178	176	176	176	176	176	176	58.81	58.81	176	63.91
G2	155	155	155	155	156	155	156	155	156	155	155	49.00	49.00	154	340.61
G3	168	168	167	166	167	167	170	167	167	169	166	43.09	763.09	166	425.22
G4	172	171	171	168	172	168	172	168	171	174	168	59.24	779.24	168	164.57
G5	168	168	170	169	168	169	168	168	168	168	168	69.86	69.86	168	103.13
H1	64	64	64	64	64	64	64	64	64	64	64	19.37	19.37	63	630.91
H2	64	63	64	63	63	64	64	64	64	64	63	39.87	99.87	63	385.65
H3	59	59	59	61	59	59	59	59	60	60	59	111.05	111.05	59	678.39
H4	58	58	58	58	58	59	58	58	58	58	58	60.52	60.52	58	103.30
H5	55	55	55	55	55	55	55	55	55	55	55	10.18	10.18	55	67.57
									Overall average:		67.20	28.64	106.64	67.10	172.86

[a] Average CPU time per replicate.
[b] Transformed SAHNM time.
[c] CFT time (adjusted to 100 MHz Pentium CPU time).

Table 2

SAHWM results for the low cost-correlated, low coverage-correlated WSCPs.

Problem	Replicate										SAHNM solution	Average time[a]	Adjusted time[b]	CFT solution	CFT time[c]
	1	2	3	4	5	6	7	8	9	10					
E1	29	29	29	29	29	29	29	29	29	29	29	0.57	0.57	29	11.30
E2	30	30	30	30	30	30	30	30	30	30	30	11.92	11.92	30	177.39
E3	27	27	27	27	27	27	27	27	27	27	27	17.09	17.09	27	40.96
E4	28	28	28	28	28	28	28	28	28	28	28	1.93	1.93	28	11.43
E5	28	28	28	28	28	28	28	28	28	28	28	0.80	0.80	28	15.91
F1	14	14	14	14	14	14	14	14	14	14	14	1.79	1.79	14	14.43
F2	15	15	15	15	15	15	15	15	15	15	15	1.53	1.53	15	13.57
F3	14	14	14	14	14	14	14	14	14	14	14	4.05	4.05	14	108.04
F4	14	14	14	14	14	14	14	14	14	14	14	2.95	2.95	14	13.48
F5	14	14	14	14	13	13	13	14	13	13	13	16.57	256.57	13	87.43
G1	176	178	176	176	178	176	176	176	176	177	176	42.26	42.26	176	63.91
G2	155	155	155	155	156	155	156	155	156	155	155	58.64	58.64	154	340.61
G3	168	167	168	166	168	166	170	168	167	168	166	65.36	785.36	166	425.22
G4	171	173	171	168	172	168	172	172	171	168	168	88.39	808.39	168	164.57
G5	168	168	169	169	169	168	168	168	169	168	168	73.70	73.70	168	103.13
H1	64	64	64	64	64	64	64	64	64	64	64	24.35	24.35	63	630.91
H2	63	64	64	64	64	64	64	64	64	64	63	32.79	32.79	63	385.65
H3	59	59	60	61	59	60	59	59	59	60	59	96.16	96.16	59	678.39
H4	58	58	58	58	58	58	58	58	58	58	58	83.38	83.38	58	103.30
H5	55	55	55	55	55	55	55	55	55	55	55	11.69	11.69	55	67.57
										Overall average:	67.20	31.80	115.78	67.10	172.86

[a] Average CPU time per replicate.
[b] Transformed SAHWM time.
[c] CFT time (adjusted to 100 MHz Pentium CPU time).

Table 3

Results for the high coverage-correlated MCSCPs.

Row × Column	Density	Number of "best" solution costs[a]		Mean percent improvement in solution cost
		SAHNM	SAHWM	(SAHWM over SAHNM)
400 × 4000	2%	0	5	2.94
	6%	3	5	1.63
	10%	4	5	1.18
800 × 8000	2%	0	5	3.64
	6%	0	5	4.05
	10%	3	5	1.90
	Total/Average	10	30	2.56

[a] The number of test problems (out of the 5 test problems associated with each combination of matrix size and density) for which the heuristic identified the best solution cost across both heuristics.

Table 4

Results for the high cost- and coverage-correlated WSCPs.

Row × Column	Correlation and density	Number of "best" solution costs[a]		Mean percent improvement in solution cost
		SAHNM	SAHWM	(SAHWM over SAHNM)
400 × 4000	NSCPs-2%	1	4	0.92
	NSCPs-6%	0	5	1.76
	NSCPs-10%	0	5	1.58
	PSCPs-2%	0	5	1.27
	PSCPs-6%	0	5	3.19
	PSCPs-10%	1	4	1.18
800 × 8000	NSCPs-2%	0	5	2.65
	NSCPs-6%	0	5	2.62
	NSCPs-10%	0	5	2.70
	PSCPs-2%	0	5	3.06
	PSCPs-6%	0	5	2.43
	PSCPs-10%	2	3	1.05
Total/Average		4	56	2.04

[a] The number of test problems (out of the 5 test problems associated with each combination of matrix size and density) for which the heuristic identified the best solution cost across both heuristics.