INVESTIGATION OF LAGRANGIAN HEURISTICS FOR SET COVERING PROBLEMS

by

Techapichetvanich, Kavee

Bricker, Dennis

Department of Industrial Engineering

The University of Iowa

Iowa City , IA 52242

USA

May 1993

ABSTRACT

This paper presents new Lagrangian Heuristics for the set covering problem (SCP). These heuristics are designed to be embedded within an algorithm (e.g., subgradient optimization) to search for optimal Lagrangian multipliers. A Lagrangian heuristic may adjust a (perhaps infeasible) solution of a Lagrangian relaxation and/or make use of information available in the form of the Lagrangian multipliers. Such an algorithm was presented by J.E. Beasley who reported that, in computational experiments, it outperformed a number of other existing heuristic algorithms. However, his heuristic algorithm which uses only the Lagrangian relaxation solution and ignores the multipliers, worked well only for random-cost problems which may bear little resemblance to typical real world applications. We present four extensions of his algorithm designed to perform well for classes of problems which appear to be much harder to solve than Beasley's random-cost problems but which more adequately model real world problems, i.e., unicost and correlated-cost problems. (The latter class displays a positive correlation between the cost of a column and its density, i.e., the number of rows covered.) Computational results, based on problems involving 200 rows and 1000 columns, indicate that our Lagrangian heuristics do produce good-quality solutions and outperform Beasley's heuristic significantly for unicost and correlated-cost problems.

KEY WORDS: set covering problem, Lagrangian heuristic algorithms, Lagrangian relaxation

Introduction and Problem Statement

Given a set of points I={1, 2, ... m} and a collection of sets A_j, j J={1, 2, ...n}, we say that set A_j "covers" point i I if i A_j, indicated by $a_{ij} = 1$, while $a_{ij} = 0$ if i A_j. A collection of sets A_j, j S where S is a subset of J, is called a "cover" of I if

$$I \quad \bigcup_{j \in S} A_j$$

The set covering problem (SCP) is the problem of finding the cover of I having minimum cost. Formally, the problem can be defined as follows:

Given a vector c and and an *m*-row by *n*-column zero-one matrix A, where

c_j = cost of column j ($c_j > 0$);

a_{ij} = 1 if row i is covered by column j,

= 0 otherwise,

SCP is the problem of selecting vector x, where

x_j = 1 if set P_j is in the cover,

= 0 otherwise,

in order to

$$\text{Minimize} \quad \sum_{j=1}^{n} c_j x_j \qquad (1)$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \geq 1 \quad , \quad i = 1,\ldots,m \qquad (2)$$

$$x_j \quad (0,1) \quad , \quad j = 1,\ldots,n \qquad (3)$$

We will use the words *row* and *column* interchangeably with *point* and *set*, respectively. Constraint (2) ensures that each row is covered by *at least* one column; if the inequalities in equation (2) are replaced by equalities, the resulting problem is called the set partitioning problem (SPP). In this paper we will restrict our attention to the set covering problem.

Real-world applications of the SCP abound, and usually fall into two categories:

1. Correlated-cost problems, displaying a positive correlation between the costs of the columns and the number of rows each covers.
2. Unicost problems, i.e., $c_j = 1, j \quad J$.

Examples of SCP applications include airline crew scheduling [1, 2, 14, 15, 20], location of emergency facilities [17, 18, 22, 24], routing problems [13], bus driver scheduling [15], truck deliveries [4], assembly line balancing [14] and informationretrieval [9].

A number of optimizing algorithms for the SCP have been described in the literature [3, 5, 8, 10, 16], typically based on tree-search procedures, but such algorithms are not computationally manageable for large-scale problems. It is clear that, given the lack of polynomial-time optimizing algorithms, there is a need for a computationally effective heuristic algorithm capable of producing good-quality (near-optimal) solutions with considerably less effort. The so-called Lagrangian heuristic algorithm demonstrated by J.E. Beasley [6] looked promising and was reported to outperform the heuristic algorithm of Balas and Ho [3] and two heuristic algorithms of Vasko and Wilson [23]. The average deviation from optimal for the Lagrangian heuristic algorithm was reported to be only 0.638%, compared to 7.416% for the Balas and Ho algorithm and 6.428% and 3.311% for those of Vasko and Wilson. However, the test problems had randomly-generated costs which may bear little resemblance to many real world applications. We will modify his algorithm so as to improve its performance in solving unicost problems and correlated-cost problems, problems which in practice prove to be much more difficult to solve than his random-cost problems. Details of Beasley's and our heuristics are presented in the next section.

Outline of the Algorithms

In this section, we present Lagrangian heuristic algorithms based upon Lagrangian relaxation of the covering constraints (equation (2)) with subgradient optimization being

used to adjust the Lagrange multipliers (optimal dual variables). Each Lagrangian relaxation provides a lower bound for the SCP, and subgradient optimization is used in an attempt to maximize this lower bound.

Readers unfamiliar with Lagrangian relaxation and subgradient optimization should refer to Fisher [11, 12].

By relaxing constraint (2) of the SCP and defining Lagrange multipliers t_i (≥ 0) for each row i (i = 1,..., m) associated with constraint (2), we have a Lagrangian relaxation problem (LRP) as follows:

$$\text{Minimize} \quad \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} t_i \left(1 - \sum_{j=1}^{n} a_{ij} x_j \right)$$

or

$$\text{Minimize} \quad \sum_{j=1}^{n} \left(c_j - \sum_{i=1}^{m} t_i a_{ij} \right) x_j + \sum_{i=1}^{m} t_i \qquad (4)$$

$$\text{subject to} \quad x_j \quad (0,1) , \; j = 1,\ldots,n \qquad (5)$$

Let $\overline{C}_j$ represent the reduced cost of column j (j = 1,..., n), i.e.,

$$\overline{C}_j = \left(c_j - \sum_{i=1}^{m} t_i a_{ij} \right),$$

and let ZLB represent the optimal objective value of the LRP above. Computing ZLB is trivial, that is, we select column j if its reduced cost $\overline{C}_j \leq 0$.

The solution x of the problem LRP may not be feasible for SCP, but adjusting this solution by adding or removing columns may provide good solutions. A heuristic algorithm which makes use of solution x of LRP and/or the Lagrangian multipliers t_i will be referred to as a Lagrangian heuristic algorithm.

To obtain good (perhaps optimal) solutions of the primal, i.e., equations (1) - (3), we will use a heuristic algorithm at each iteration of the dual search. In this paper we will describe five Lagrangian heuristic algorithms, all of which begin with the solution to the

Lagrangian relaxation problem, add sets to the cover to gain feasibility if necessary, and remove superfluous sets in the solution. Beasley's algorithm [6] will be referred to as heuristic #1 in this paper for ease of reference. Our other 4 heuristic algorithms are essentially extensions of Beasley's, and, like his algorithm, use not only the solution (perhaps infeasible) of the Lagrangian relaxation. Unlike his algorithm, however, our algorithms also use the information available from the Lagrangian multipliers.

These five heuristic algorithms differ in the criteria for choosing a set to be added (in order to cover a point not currently covered) and for removing a superfluous set:

- Heuristic #1 (of Beasley), after identifying an uncovered point, adds the set which covers the point at lowest cost. This is repeated until a feasible cover is obtained, after which the algorithm removes the superfluous sets (if any) having highest cost.
- Heuristic #2, after identifying an uncovered point, considers the q least-cost sets covering the point as candidates and, of these, adds to the cover the set which has the lowest "reduced cost". Like heuristic #1, when a feasible cover is obtained, it removes the superfluous sets having highest cost. We use a large value of q in order to guarantee that all potential candidate columns that could be added to the cover are considered, although a small value of q ("partial pricing") could be used to save some computational effort, possibly at the expense of a higher cost solution.
- Heuristic #3, like heuristic #2, uses "partial pricing" by selecting, from q candidates which cover a point, the set having lowest "reduced cost", but for this purpose a "modified reduced cost" is computed with zero replacing the Lagrange multipliers of points already covered. Like heuristic algorithms #1 and #2, it removes the superfluous set with highest cost.
- Heuristic #4, an extension of heuristic #2, differs from heuristic #2 only in that, when selecting a superfluous set to be removed from the cover, the superfluous set having highest *reduced* cost is selected.

• Heuristic #5, an extension of heuristic #3, differs from heuristic #3 only in that selection of a superfluous set to be removed from the cover is again based upon the modified reduced cost.

When choosing a column to be added to the cover, the consideration of reduced cost (as used in heuristic #2 and #4) and the modified reduced cost (as used in heuristic #3 and #5) seem to be more effective than consideration of the original cost (as used in heuristic #1). In a sense, the use of reduced cost makes the heuristic less "myopic". Recall that the reduced cost $\overline{C}_j = (c_j - \sum_{i=1}^{m} t_i a_{ij})$. Hence, the higher the number of rows which a certain column covers, the lower its reduced cost is likely to be. While several points might be left uncovered in the solution of LRP, all of these heuristics are myopic in that they will attempt to cover these points one at a time. When, in order to cover a certain point, we choose to add a column with the lowest reduced cost to the cover, we are likely to be better off, since that column tends to cover more rows than do other columns having higher reduced costs.

The same reasoning suggests the use of the modified reduced cost in Heuristics #3 and #5. But here, we go one step further by not giving credit, when calculating the reduced cost of a candidate column, for covering rows which are covered by other columns already included in the cover. That is, the substitution of zero for the Lagrange multipliers (t_i) of rows already covered will make the modified reduced cost of that column higher and therefore less attractive when choosing the next set to be added to the cover.

Before we go into the details of these algorithms, we would like to point out that rows and columns of the test problems can be pre-arranged in a systematic way for more computationally efficient performance without loss of generality as follows:

• First, we order the columns in ascending cost order, with columns of equal cost being ordered in descending order of the number of rows they cover. In essence this means that for any row i the column min { j | a_{ij} = 1, j = 1,...,n } is the "best" column to use in covering row i.

• Next, we order the rows in ascending order of the number of columns covering them. This idea arises from the intuition that we are better off trying to cover first that row which has the fewest columns covering it. Thus, if in actual computation we always start processing data from the first row to the last row, then the first row should be the row with a minimum number of columns covering it.

Now, let ZMAX represent the maximum lower bound found, ZUB the best feasible solution found and P_k the lower bound when column k (k = 1,...,n) is forced to be in the solution. Denote by $\min_q X$ the set of q smallest values of the set X, and $\text{argmin}_q X$ the indices of these values. The details of the algorithm can be presented in the following steps:

Step 1. Initialize ZMAX = -∞, ZUB = ∞, $P_j = c_j$ (j = 1,..., n) and $t_i = \min [c_j \mid a_{ij} = 1, j \in J]$ for each i ∈ I.

Step 2. Solve LRP with the current set of multipliers (t_i) and denote the solution by ZLB and x_j , j ∈ J. Update ZMAX by ZMAX = max(ZMAX , ZLB).

Step 3. Apply one of the heuristic algorithms to construct a feasible solution S to the original SCP. Denote the cost of this solution, which is an upper bound on the optimal solution, by ZUB.

Step 4. Stop if ⌈ZMAX⌉ = ZUB (ZUB is then the optimal solution), where ⌈a⌉ denotes the ceiling value of a, i.e., the smallest integer greater than or equal to a.

Step 5. Considering the original SCP [equations (1) - (3) above] and the corresponding LRP [equations (4) and (5) above] it is clear that imposing the additional constraint that a particular column k must be in the cover (x_k = 1) results in an SCP whose corresponding

lower bound is ZLB + $\overline{C}_k$ if $x_k = 0$ in the solution of LRP, and ZLB otherwise (i.e., if $x_k = 1$). Hence we can update P_k using

$$P_k = \max (P_k , ZLB + \overline{C}_k), \qquad \text{if } x_k = 0, \quad k = 1,...,n. \qquad (6)$$

$$P_k = \max (P_k , ZLB), \qquad \text{if } x_k = 1, \quad k = 1,...,n. \qquad (7)$$

We can therefore remove columns from further consideration by setting

$$c_k = \infty, \qquad \text{if } P_k > ZUB, \quad k = 1,...,n. \qquad (8)$$

since plainly a column k with P_k (the lower bound corresponding to an optimal solution containing column k) greater than ZUB cannot be in an improved feasible solution.

Step 6. Calculate the subgradient (G) of the dual objective using

$$G_i = 1 - \sum_{j=1}^{n} a_{ij} x_j \qquad , \qquad i = 1,...,m. \qquad (9)$$

We found it computationally useful to adjust the components of the subgradient, if they were not going to effectively contribute to the update of the multipliers, in the following manner:

$$G_i = 0, \quad \text{if } t_i = 0 \text{ and } G_i < 0 , \quad i = 1,...,m. \qquad (10)$$

Step 7. Stop, if $\left(\sum_{i=1}^{m} (G_i)^2 \right) = 0$ since in this case the Lagrangian dual has been optimized.

Step 8. Define a step size T by

$$T = f (1.05 ZUB - ZLB) / \left(\sum_{i=1}^{m} (G_i)^2 \right) \qquad (11)$$

where f = 2 initially. If ZMAX has not increased in the last 30 iterations of the subgradient procedure with the current value of f then f is halved. This approach to deciding the value of f is modeled on that of Fisher [11, 12]. Equation (11) is the standard expression for step size in subgradient optimization except for the factor 1.05 which we have included to be consistent with the results of Beasley [6], who states that he used this value because

computationally he found it useful in ensuring that T does not become too small as the gap between ZUB and ZLB closes.

Step 9. Stop if the algorithm reaches the maximum number of iterations allowed (an arbitrarily decided stopping criterion). Here we terminated after 1000 iterations. Note that we used this maximum number of iterations as a stopping criterion rather than $f \leq 0.005$ as did Beasley [6], because we felt that it permits a more fair comparison of the results from different heuristic algorithms.

Step 10. Update the Lagrange multipliers using

$$t_i = \max (0 , t_i + TG_i), \quad i = 1,...,m \qquad (12)$$

(the standard expression for updating nonnegative Lagrange multipliers) and go to step (2) to solve the new LRP problem.

The Lagrangian heuristic algorithms which might be utilized in step 3 are as follows:

Heuristic #1 [9]

(a) Let $S = \{j \mid x_j = 1, j = 1,...,n \}$ be the solution to LRP.

(b) Select a row i which is uncovered (i.e., $\sum_{j=1}^{n} a_{ij} x_j = 0$). If none, go to (d).

(c) Add to S the column corresponding to $\min \{j \mid a_{ij} = 1, c_j < \infty, j = 1,...,n \}$. Go to (b).

(d) Consider each column $j \in S$ in descending index (j) order; if $S - \{j\}$ is a feasible solution to the SCP, let $S = S - \{j\}$.

(e) Update ZUB by $ZUB = \min (ZUB , \sum_{j \in S} c_j)$.

Heuristic #2

(a) Let $S = \{ j \mid x_j = 1, j = 1,...,n \}$ be the solution to LRP.

(b) Select a row i which is uncovered (i.e., $\sum_{j=1}^{n} a_{ij} x_j = 0$). If none, go to (d).

(c) Add to S the column corresponding to min { $\overline{C}_h$ | h argmin$_q$ [j | $a_{ij} = 1$, $c_j < \infty$, j J]}. Go to (b).

(d) Consider each column j S in descending index (j) order; if S - {j} is a feasible solution to the SCP, let S = S - {j}.

(e) Update ZUB by ZUB = min (ZUB , $\sum_{j \in S} c_j$).

Heuristic #3

(a) Let S = { j | $x_j = 1$, j = 1,...,n } be the solution to LRP.

(b) Select a row i which is uncovered (i.e., $\sum_{j=1}^{n} a_{ij} x_j = 0$). If none, go to (d).

(c) Add to S the column corresponding to min { $\overline{C}_h$ | h argmin$_q$ [j | $a_{ij} = 1$, $c_j < \infty$, j J]} and for every j J such that a_{ij}=1, replace $\overline{C}_j$ by $\overline{C}_j + t_i$. Go to (b).

(d) Consider each column j S in descending index (j) order; if S - {j} is a feasible solution to the SCP, set S = S - {j}

(e) Update ZUB by ZUB = min (ZUB , $\sum_{j \in S} c_j$).

Heuristic #4

(a) Let S = { j | $x_j = 1$, j = 1,...,n } be the solution to LRP.

(b) Select a row i which is uncovered (i.e., $\sum_{j=1}^{n} a_{ij} x_j = 0$). If none, go to (d).

(c) Add to S the column corresponding to min { $\overline{C}_h$ | h argmin$_q$ [j | $a_{ij} = 1$, $c_j < \infty$, j J]}. Go to (b).

(d) Consider each column j S in descending "reduced cost" ($\overline{C}_j$) order; if S - {j} is a feasible solution to the SCP, set S = S - {j}.

(e) Update ZUB by ZUB = min (ZUB , $\sum_{j \in S} c_j$).

Heuristic #5

(a) Let $S = \{ j \mid x_j = 1, j = 1,...,n \}$ be the solution to LRP.

(b) Select a row i which is uncovered (i.e., $\sum_{j=1}^{n} a_{ij} x_j = 0$). If none, go to (d).

(c) Add to S the column corresponding to min $\{ \overline{C}_h \mid h \quad \text{argmin}_q [j \mid a_{ij} = 1, c_j < \infty, j \in J] \}$ and for every $j \in J$ such that $a_{ij}=1$, replace $\overline{C}_j$ by $\overline{C}_j + t_i$. Go to (b).

(d) Consider each column $j \in S$ in descending "modified reduced cost" ($\overline{C}_j$) order; if $S - \{j\}$ is a feasible solution to the SCP, set $S = S - \{j\}$.

(e) Update ZUB by ZUB = min (ZUB , $\sum_{j \in S} c_j$).

Note that, in heuristic #1, the use of the column ordering when adding to the cover is ineffective with the unicost problems, since this ordering would not be sufficiently discriminating between columns to give good-quality results. In this case, heuristics #2-5, which are based upon the reduced costs rather than the original costs, perform better, as will be shown in the computational results in the next section.

Computational Results

The four new Lagrangian heuristics presented in this paper together with Beasley's Lagrangian heuristic were programmed in FORTRAN 77 and implemented on Hewlett Packard/Apollo DN 10000 workstations supported by the Iowa Computer-Aided Engineering Network (ICAEN). In order to compare the performance of these 5 Lagrangian heuristic algorithms, we coded them with the same program structure without utilizing vector or parallel processing capabilities. We have solved each of 50 test problems using each of these 5 Lagrangian heuristic algorithms. "Partial pricing" was not used in these tests, i.e., $q= \infty$ in heuristic algorithms #2 through 5.

Table 1 presents a summary of the test problems. Problem sets 4 and 6 were taken from Beasley [6, 7]. These problems were created by Beasley using the scheme of Balas

and Ho [3], namely, column costs (c_j) are integers randomly generated, every column covers at least one row, and every row must be covered by at least two columns.

Problem sets 4c and 6c were modifications of the original problem sets 4 and 6, having the same A matrices but with the costs randomly generated so as to have high correlations between the cost of a column and the number of rows covered by that column. (Coefficients of correlation are in the range of 0.75 - 0.90.) Generating the costs for these problem sets was done so as to reflect the property of many real world applications that the marginal unit cost to cover more rows is decreasing as the number of rows already covered by that column is increasing. For example, in a facility location application of SCP, there are fixed cost and variable cost elements incurred in setting up a facility to serve the demand points.

Problem sets 4u and 6u were also modifications of the original problem sets 4 and 6, having the same A matrices but with uniform costs of 1 for all the columns.

Table 1. Test problem details

Problem set	Number of rows	Number of columns	Density* (%)	Cost characteristics	# of problems in problem set
4	200	1000	2	Randomly generated between 1- 100	10
6	200	1000	5	" "	5
4c	200	1000	2	Correlated with # of rows covered	10
6c	200	1000	5	" "	10**
4u	200	1000	2	Unicosts of 1	10
6u	200	1000	5	"	5

* Note that the density of a SCP is the percentage of ones in the (a_{ij}) matrix.

** Note that there are 10 test problems in problem set 6c in which 5 of them have cost values generated from one seed number while the other 5 have cost values generated from the other seed number.

The computational results obtained from the Lagrangian heuristic algorithms were compared and summarized in Tables 2 - 6, which include:

- ZMAX : value of the greatest lower bound found
- Lowest ZUB : value of the best feasible solution found

- Avg of ZUB : average value of the feasible solutions at each iteration
- CPU Time : total computation time in Apollo DN 10000 seconds excluding input/output time

For more complete results, see Techapichetvanich [21] where the following results are also presented:

- Iteration # where the best feasible solution was found
- Total number of subgradient iterations run
- Frequency of the best feasible solution found
- Total number of columns removed at the end of Lagrangian heuristic algorithm

Table 2 compares the best upper bounds found by each Lagrangian heuristic algorithm for problem sets 4 and 6. In Table 3 is a similar comparison for problem sets 4c and 6c, and in Table 4 for problem sets 4u and 6u. As in Beasley [6], we refer to the first problem in problem set 4 as problem 4.1, while problem 4.2 refers to the second problem in set 4, etc.. This applies to all other problem sets as well. The optimal solutions for problem sets 4 and 6 are reported by Beasley [5], while they are unknown for problem sets 4c, 6c, 4u and 6u. (Here, we have only intended to compare the performance of our 4 Lagrangian heuristic algorithms to Beasley's Lagrangian heuristic algorithm; thus, we did not require the optimal solutions of these problem sets.) Table 5 shows the mean of the ratio of CPU time taken by our 4 Lagrangian heuristic algorithms to that taken by Beasley's Lagrangian heuristic algorithm (#1). Table 6 presents the mean of the ratio of the average of ZUBs from our 4 Lagrangian heuristic algorithms to the average ZUB of Beasley's algorithm.

Table 2. Summary of computational results for problem sets 4 and 6

Problem		Lowest ZUB found from Lagrangian heuristic algorithm #				
Number	ZMAX	1	2	3	4	5
4.1	428.61	429* (389)	429* (236)	429* (146)	429* (236)	429* (146)

4.2	511.14	512* (224)	512* (174)	512* (196)	512* (350)	512* (196)
4.3	515.83	516* (136)	516* (152)	516* (79)	516* (408)	516* (158)
4.4	493.99	495	494*	495	494*	495
4.5	511.56	512* (261)	512* (225)	512* (219)	512* (173)	512* (180)
4.6	557.23	561	**560**	**560**	561	**560**
4.7	429.39	430* (317)	430* (196)	430* (279)	430* (188)	430* (279)
4.8	488.67	**492** (636)	**492** (208)	**492** (349)	**492** (227)	**492** (512)
4.9	638.39	**641** (570)	**641** (469)	**641** (443)	**641** (338)	**641** (443)
4.10	513.25	514* (141)	514* (140)	514* (130)	514* (10)	514* (130)
6.1	133.11	140	141	139	141	139
6.2	140.39	150	149	148	150	148
6.3	139.91	**145** (198)	**145** (15)	**145** (103)	**145** (21)	**145** (103)
6.4	128.91	**131** (123)	**131** (107)	**131** (291)	**131** (119)	**131** (209)
6.5	153.19	165	165	**161**	168	**161**

Note that * associated with the lowest ZUBs in Table 2 indicates that that ZUB converged with ZMAX which means that it is optimal value. Also, the boldface number, e.g., **560**, indicates that that ZUB is optimal value but did not converge with ZMAX.

Note also that ZMAXs from each Lagrangian heuristic algorithm for the same problem are almost identical. The ceilings of ZMAX (éZMAXù) from each Lagrangian heuristic algorithm are mostly the same. Here in Tables 2 - 4, we show the largest one.

The number in parenthesis following the lowest ZUB represents the iteration# that that ZUB was first found. Here we show only in the problems where all 5 Lagrangian heuristic algorithms yield the same solution.

Table 3. Summary of computational results for problem sets 4c and 6c

Problem		Lowest ZUB found from Lagrangian heuristic #				
Number	ZMAX	1	2	3	4	5
4.1c	2356.76	2958	2902	2774	2874	2774
4.2c	2386.27	2999	2871	2813	2898	2826
4.3c	2372.95	2968	2926	2864	2999	2864
4.4c	2397.40	2927	2847	2814	2992	2814
4.5c	2408.19	3073	2981	2911	2985	2855
4.6c	2361.30	2978	2902	2844	2950	2844
4.7c	2435.18	3113	3035	2912	3027	2885
4.8c	2348.15	2917	2900	2770	2877	2732
4.9c	2346.22	2925	2895	2855	2889	2855
4.10c	2423.49	3085	3023	2920	3069	2920
6.1'c	2170.63	3854	3837	3092	3965	3154
6.2'c	2169.93	3766	3746	3162	3925	3162
6.3'c	2170.52	3870	3900	3139	3957	3139
6.4'c	2172.29	3742	3894	3223	4032	3223
6.5'c	2175.30	3784	3536	3193	3907	3136
6.1"c	2143.45	3504	3409	3240	3523	3225
6.2"c	2157.90	3474	3445	3376	3603	3338
6.3"c	2145.67	3494	3461	3400	3577	3233
6.4"c	2154.96	3529	3451	3367	3645	3367
6.5"c	2162.84	3593	3448	3409	3670	3409

Table 4. Summary of computational results for problem sets 4u and 6u

Problem Number	ZMAX	Lowest ZUB found from Lagrangian heuristic # 1	2	3	4	5
4.1u	32.78	47	44	45	44	45
4.2u	31.68	45	41	41	42	42
4.3u	32.43	46	42	44	43	43
4.4u	33.25	47	45	47	46	46
4.5u	32.77	45	43	43	41	44
4.6u	32.22	47	42	44	43	43
4.7u	33.50	45	43	42	43	43
4.8u	31.74	44	43	44	43	45
4.9u	32.86	48	42	44	44	44
4.10u	33.28	44	44	45	43	45
6.1u	14.76	26	25	25	26	26
6.2u	14.26	26	24	25	24	25
6.3u	14.84	28	26	26	26	25
6.4u	14.66	27	26	26	25	26
6.5u	14.88	26	25	25	25	26

Table 5. Summary of mean and standard deviation of the ratio of the CPU time from Lagrangian heuristics #2 to #5 to the CPU time from Lagrangian heuristic #1

Problem Sets	(Mean of [CPU time i / CPU time 1] , Std. dev. of [CPU time i / CPU time 1]) i = 2	i = 3	i = 4	i = 5
4	(0.93 , 0.21)	(1.31 , 0.17)	(1.23 , 0.32)	(1.41 , 0.19)
6	(1.05 , 0.02)	(1.52 , 0.02)	(1.12 , 0.04)	(1.55 , 0.02)
4c	(1.10 , 0.01)	(1.41 , 0.04)	(1.43 , 0.04)	(1.49 , 0.03)
6c	(1.06 , 0.02)	(1.42 , 0.08)	(1.46 , 0.08)	(1.54 , 0.06)
4u	(1.07 , 0.01)	(1.32 , 0.01)	(1.32 , 0.04)	(1.48 , 0.04)
6u	(1.06 , 0.01)	(1.41 , 0.01)	(1.27 , 0.04)	(1.60 , 0.05)

Table 6. Summary of mean and standard deviation of the ratio of the average ZUB of Lagrangian heuristics #2 to #5 to the average ZUB of Lagrangian heuristic #1

Problem Sets	(Mean of [Avg of ZUB i / Avg of ZUB 1] , Std. dev. of [CPU time i / CPU time 1]) i = 2	i = 3	i = 4	i = 5
4	(1.002 , 0.010)	(0.998 , 0.009)	(1.037 , 0.021)	(1.000 , 0.010)
6	(1.019 , 0.010)	(1.013 , 0.006)	(1.087 , 0.029)	(1.013 , 0.007)
4c	(0.995 , 0.012)	(0.967 , 0.006)	(1.057 , 0.016)	(0.966 , 0.005)
6c	(1.018 , 0.010)	(0.977 , 0.005)	(1.105 , 0.020)	(0.978 , 0.005)
4u	(0.853 , 0.018)	(0.873 , 0.020)	(0.861 , 0.018)	(0.887 , 0.019)
6u	(0.884 , 0.020)	(0.890 , 0.023)	(0.900 , 0.015)	(0.933 , 0.008)

In Table 2, we observe that, for the random-cost problems (sets 4 and 6), our Lagrangian heuristics (#2 - #5) seem to give a better solution, i.e., lower ZUB, compared to Beasley's Lagrangian heuristic (#1) for almost all of the problems. Based on the CPU time taken (see Table 5), Lagrangian heuristic #2 appears attractive because of generally lower CPU time. Another interesting observation is that of the iteration number in which the lowest ZUB was first found (reported by the numbers in parentheses). For those test problems where the same lowest ZUB is found by all 5 Lagrangian heuristics, Lagrangian heuristics #2-5 tended to find the best ZUB earlier than did Beasley's. Note also that Lagrangian heuristic #3 and its extension, Lagrangian heuristic #5, tended to find better solutions than did the other Lagrangian heuristics for problem set 6 where the data matrix is more dense.

A result which was first surprising is that, for some problems, the average ZUBs (over all iterations) from Lagrangian heuristics #2-5 seem to be worse than those of Beasley's, even though the best feasible solutions found from ours are better. (See Table 6.) A likely explanation of this result is that Lagrangian heuristic #1 is insensitive to Lagrange multipliers, i.e., it is not affected by the change in multipliers when choosing a column to cover a row (see step 3(b) for Lagrangian heuristic #1). This results in a rather low number of distinct feasible solutions evaluated, and thus a more stable average of ZUBs, i.e., these values do not vary so much as from iteration to iteration. By contrast, our 4 Lagrangian heuristics are sensitive to the change in multipliers since in step (c) when we choose a column to cover a row, we consider the reduced costs which are very sensitive to the multipliers. This causes the ZUB in each iteration to vary, resulting in higher average values. But by evaluating a greater variety of feasible solutions, the search is diversified and we are more likely to find a better ZUB, as our results show.

From Table 3, we observe that, for the correlated-cost problems, Lagrangian heuristics #2, #3, and #5 outperformed Beasley's in terms of the quality of solution found, with the best solutions generally found by either #3 or #5. On the other hand, Lagrangian

heuristic #4 does not look promising here. Lagrangian heuristics #2-5 took more CPU time than did Beasley's, as we expected, because of a more complicated procedure in step (c) and (d) of these Lagrangian heuristics. Nonetheless, this additional CPU time is warranted by the much better solutions obtained.

The average of the ZUBs of Lagrangian heuristics #3 and #5 are evidently better than that of Lagrangian heuristic #1 due to much better solutions found. However, this is not true for Lagrangian heuristic #2 for which the best ZUBs found are not really much better than those from Lagrangian heuristic #1 and so the vacillating behavior described above still dominated.

From Table 4, we observe that, for the unicost problems, at least one of our 4 Lagrangian heuristics gave better solutions than Beasley's in each of the test problems. Lagrangian heuristic #2 looks most promising. The results here have confirmed Beasley's report of poor performance of his Lagrangian heuristic for unicost problems. Again, the CPU times taken from our 4 Lagrangian heuristics are greater than those of Beasley's; however, the averages of ZUBs from our 4 Lagrangian heuristics are better than those found by Beasley's heuristic.

Note that for all of the correlated-cost and unicost problems, a gap remains between the ZMAXs and the lowest ZUBs; therefore, optimality cannot be verified. The gaps in these problems are fairly large, and are the sums of the gap between ZUB and the (unknown) optimum, and the gap between the optimum and ZMAX. Since the ZMAXs found from these 5 Lagrangian heuristics are almost equal for most of the test problems here, this suggests that we cannot significantly improve the ZMAX. (Even running these 5 Lagrangian heuristics for 4000 more iterations still yielded negligible improvement in ZMAXs and no improvement in ZUBs.) Hence, it appears that the gap can be narrowed only by reducing the ZUB (given that the current lowest ZUB is not yet optimal). Additional approaches may be required to accomplish this task. One of the potential approaches with which we experimented is tabu search. Details regarding a tabu search

procedure for the SCP can be found in Techapichetvanich [34] and will be reported in a later paper.

Conclusions

It is clear that our four Lagrangian heuristic algorithms presented in this paper have an equal, or superior, performance to the algorithm of Beasley [6] for almost all of these reasonably sized test problems. This is especially true for the correlated-cost and unicost problems, which more nearly resemble real world applications and prove to be much harder to solve than random-cost problems because of less dominance among the columns.

Which of our 4 Lagrangian heuristic algorithms should be used to solve a particular set-covering problem is dependent on the nature of that problem. If it is a correlated-cost problem, our Lagrangian heuristics #3 and #5 tend to give the best solutions. If it is a unicost problem, any of our 4 Lagrangian heuristics should provide a better solution on the majority of the test problems with a high likelihood of Lagrangian heuristic #2 being the best one. If it is a random-cost problem, Lagrangian heuristic #2 seems to be a good choice in terms of the quality of solution and CPU time required.

References

1. Arabeyre, J. P., Fearnley, J., Steiger, F. C., and Teather, W., "The Airline Crew Scheduling Problem: A Survey", *Transportation Science*, **3**(2), pp. 140-163 (1969).

2. Baker, E. K., Bodin, L. D., Finnegan, W. F., and Ponder, R. J., "Efficient Heuristic Solutions to an Airline Crew Scheduling Problem", *AIIE Transactions*, **11**, pp. 79-85 (1979).

3. Balas, E., and Ho, A., "Set Covering Algorithms Using Cutting Planes, Heuristics, and Subgradient Optimization: A Computational Study", *Mathematical Programming Study*, **12**, pp. 37-60 (1980).

4. Balinski, M., and Quandt, R., "On an Integer Program for a Delivery Problem", *Operations Research*, **12**(2), pp. 300-304 (1964).

5. Beasley, J. E., "An Algorithm for Set-Covering Problems", *European Journal of Operational Research*, **31**, pp. 85-93 (1987).

6. Beasley, J. E., "A Lagrangian Heuristic for Set-Covering Problems", *Naval Research Logistics*, **37**, pp. 151-164 (1990).

7. Beasley, J. E. (1990). OR-Library: Distributing Test Problems by Electronic Mail. *Journal of the Operational Research Society*, **41**(11), pp. 1069-1072 (1990).

8. Christofides, N., and Korman, S., "A Computational Survey of Methods for the Set Covering Problem", *Management Science*, **21**(5), pp. 591-599 (1975).

9. Day, R. H., "On Optimal Extracting from a Multiple File Data Storage System: An Application of Integer Programming", *Operations Research*, **13**, pp. 482-494 (1965).

10. Etcheberry, J., "The Set-Covering Problem: A New Implicit Enumeration Algorithm", *Operations Research*, **25**(5), pp. 760-772 (1977).

11. Fisher, M. L., "The Lagrangian Relaxation Method for Solving Integer Programming Problems", *Management Science*, **27**(1), pp. 1-18 (1981).

12. Fisher, M. L., "An Applications Oriented Guide to Lagrangian Relaxation", *Interfaces*, **15**(2), pp. 10-21 (1985).

13. Foster, B. A., and Ryan, D. M., "An Integer Programming Approach to the Vehicle Scheduling Problem", *Operations Research Quarterly*, **27**, pp. 367-384 (1976).

14. Marsten, R. E., "An Algorithm for Large Set Partitioning Problems", *Management Science*, **20**, pp. 774-787 (1974).

15. Marsten, R. E., and Shepardson, F., "Exact Solutions of Crew Scheduling Problems Using the Set Partitioning Problem: Recent Successful Applications", *Networks*, **11**, pp. 165-177 (1981).

16. Paixao, J., "Algorithms for Large Scale Set Covering Problems", Ph.D. thesis, School of Management, Imperial College, London SW7 2AZ, England, 1984.

17. Patel, N., "Locating Rural Social Service Centers in India", *Management Science*, **25**(1), pp. 22-30 (1979).

18. Revelle, C., Marks, D., and Liebman, J. C., "An Analysis of Private and Public Sector Location Models", *Management Science*, **16**, pp. 692-707 (1970).

19. Salveson, M. E., "The Assembly Line Balancing Problem", *Journal of Industrial Engineering*, **6**, pp. 18-25 (1955).

20. Spitzer, M., "Solution to the Crew Scheduling Problem", presented at the first AGIFORS symposium, October 1961.

21. Techapichetvanich, K., "Investigation of Lagrangian Heuristic and Tabu Search Algorithms for Set Covering Problems", Ph.D. thesis, Department of Industrial Engineering, The University of Iowa, Iowa City, Iowa, USA, 1993.

22. Toregas, C., Swain, R., ReVelle, C., and Bergman, L., "The Location of Emergency Service Facilities", *Operations Research*, 19(6), pp. 1363-1373 (1971).

23. Vasko, F. J., and Wilson, G. R., "An Efficient Heuristic for Large Set Covering Problems", *Naval Research Logistics Quarterly*, **31**, pp. 163-171 (1984).

24. Walker, W., "Application of the Set Covering Problem to the Assignment of Ladder Trucks to Fire Houses", *Operations Research*, **22**, pp. 275-277 (1974).