EXPERIMENTS WITH LAGRASP HEURISTIC FOR SET K-COVERING

LUCIANA S. PESSOA, MAURICIO G. C. RESENDE, AND CELSO C. RIBEIRO

Abstract. The set k-covering problem is a variant of the classical set covering problem, in which each object is required to be covered at least k times. We describe a hybrid Lagrangean heuristic, named LAGRASP, which combines subgradient optimization and GRASP with path-relinking to solve the set k-covering problem. Computational experiments carried out on 135 test instances show experimentally that by properly tuning the parameters of LAGRASP, it is possible to obtain a good trade-off between solution quality and running times. Furthermore, LAGRASP makes better use of the dual information provided by subgradient optimization and is able to discover better solutions and to escape from locally optimal solutions even after the stabilization of the lower bounds, whereas other strategies fail to find new improving solutions.

1. Introduction

The set k-covering problem (SC$_k$P) is a generalization of the set covering problem, in which each element should be covered, at least, k times [1]. Let $A = \{a_{ij}\}_{i=1,\ldots,m}^{j=1,\ldots,n}$ be a binary matrix and $c_j > 0$ the cost of column $j = 1, \ldots, n$. A column j covers a row i if $a_{ij} = 1$. The solution of problem SC$_k$P consists in finding a minimum-cost subset S of columns of matrix A, such that each row of the latter is covered by at least k columns in S.

An integer programming formulation for the set k-covering problem is

$$z(x) = \min \sum_{j=1}^{n} c_j x_j \tag{1}$$

s.t.

$$\sum_{j=1}^{n} a_{ij} x_j \geq k, \quad i = 1, \ldots, m, \tag{2}$$

$$x_j \in \{0, 1\}, \quad j = 1, \ldots, n. \tag{3}$$

where $x_j = 1$ if column j belongs to the minimum-cost subset S, $x_j = 0$ otherwise.

Applications of set multicovering arise in a variety of fields, such as marketing, logistics, security, telecommunications, and computational biology [2, 3, 4]. Though many of these applications can be modeled as set covering problems, for reliability

Date: March 1, 2011.

Key words and phrases. GRASP, hybrid heuristics, metaheuristics, path-relinking, Lagrangean relaxation, Lagrangean heuristics, local search, set covering, set multicovering, set k-covering.

AT&T Labs Research Technical Report.

purposes they are treated as multicovering. Other variants of the set covering problem and related applications can also be found in [5, 6, 7].

Lucena [8] noticed that few branch-and-cut algorithms exist for set covering problems and its variants, because the separation problem associated with some families of strong inequalities ([9, 10, 11]) are very hard to be solved even for heuristics. Therefore, the use of heuristics for large problems with large duality gaps is a need.

Pessoa and Ribeiro [12] proposed a GRASP heuristic for SC_kP in the computational biology context. In this paper we present extensive computational experiments with variants of a LAGRASP for set k-covering. LAGRASP [13] is a Lagrangean heuristics, based on the hybridization of subgradient algorithms to solve its Lagrangean relaxation with greedy and GRASP heuristics.

This paper is organized as follows. In Section 2 we briefly describe a GRASP with path-relinking heuristic, as well a Lagrangean heuristic for the set k-covering problem. Futhermore, we give a description of how both are combined to find approximate solutions to the SC_kP. Experimental results are presented in Section 3. Finally, in Section 4 we draw some concluding remarks.

2. Computational Methods

2.1. **GRASP with Path-Relinking.** GRASP is a multi-start metaheuristic, in which each iteration consists of two phases: construction and local search [14].

During the first phase solutions are constructed by a randomized greedy algorithm. A greedy algorithm builds a solution, adding one column at a time to a partial solution, until all rows are k-covered, i.e. are covered by at least k columns. Initially, the partial solution does not contain any column, so all rows are uncovered. Let the *cardinality* of a column be the number of rows that are not k-covered and that can be covered by the column. At each step of the construction, a partial solution is on hand. If a column is not in the partial solution, it is a candidate to be added to the partial solution if its cardinality is positive. Columns which attend these criteria compose a candidate list L. Each column $j \in L$ is evaluated according to a greedy function defined to be the ratio ρ_j of the cost of the column to its cardinality. The greedy algorithm adds to the partial solution a column with minimum greedy function value. A randomized version of the above greedy algorithm defines a restricted candidate list (RCL) made up of candidate elements with a greedy function value below a specified cut-off value. A standard cut-off is defined to be $\rho^- + \alpha(\rho^+ - \rho^-)$, where $\rho^- = \min\{\rho_j \mid j \in \text{L}\}$, $\rho^+ = \max\{\rho_j \mid j \in \text{L}\}$, and $\alpha \in \Re$ is such that $0 \leq \alpha \leq 1$. The column to be added to the partial solution is selected at random from the RCL.

Solutions built with the randomized greedy algorithm are not guaranteed to be locally optimal, even with respect to simple neighborhood structures. Therefore, the application of local search to such a solution usually results in an improved locally optimal solution. Feo and Resende [14] proposed a local search algorithm based on q, p-exchanges. Every q columns in the current solution S are considered to be exchange by every possible set of p columns not in S so as the cost function value is decreased. The local search proposed in this paper makes use of a $(1, 0)$-exchange algorithm, in which we attempt to remove superfluous columns from the multicover, and a $(1, 1)$-exchange algorithm, in which we attempt to replace a more expensive column in the multicover by a less expensive unused one.

The basic implementation of GRASP is memoryless, since any iteration does not make use of information collected in previous iterations. Path-relinking is an intensification strategy that can be applied to introduce memory structures in GRASP [15]. It explores paths in the solution space connecting good-quality solutions, one of them being an initial solution x^s and the other a target solution x^t. The procedure maintains a pool P of diverse elite solutions during the search. Path-relinking is carried out after local search, between the local minimum x and a randomly selected pool solution x_p. The attribution of x and x^p to the initial solution x^s or to the target solution x^t depends on the path-relinking strategy. Different approaches have been considered in the implementation of this procedure [15]. We used the backward strategy, in which the initial solution is the best between x and x_p.

2.2. **Lagrangean Heuristics.**

In the following, we base our description on [13]. Lagrangean relaxation [16, 17] is a mathematical programming technique that can be used to provide lower bounds for combinatorial optimization problems. However, the primal solutions produced by the algorithms used to solve the Lagrangean dual problem are not necessarily feasible. Held and Karp [18, 19] were among the first to explore the use of the dual multipliers produced by Lagrangean relaxation to derive lower bounds, applying this idea in the context of the traveling salesman problem.

Lagrangean heuristics exploit the dual multipliers to generate primal feasible solutions. Beasley [20, 21] described a Lagrangean heuristic for set covering which can be extended to the set k-covering problem.

A Lagrangean relaxation of the set k-covering problem can be defined by associating dual multipliers $\lambda_i \in \mathbb{R}_+$, for $i = 1, \ldots, m$, to each inequality (2). This results in the following *Lagrangean relaxation problem* LRP(λ):

$$\begin{aligned} \min \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} \lambda_i (k - \sum_{j=1}^{n} a_{ij} x_j) \\ \text{s.t.} \\ x_j \in \{0,1\}, \quad j = 1, \ldots, n. \end{aligned}$$

By letting $c'_j = c_j - \sum_{i=1}^{m} \lambda_i a_{ij}$, formulation LRP$(\lambda)$ simplifies to

$$\begin{aligned} z'(\lambda) = \min \ \sum_{j=1}^{n} c'_j x_j + \sum_{i=1}^{m} \lambda_i k \\ \text{s.t.} \\ x_j \in \{0,1\}, \quad j = 1, \ldots, n, \end{aligned}$$

whose optimal solution $x'(\lambda)$ is given by

$$x'_j(\lambda) = \begin{cases} 1, & \text{if } c'_j \le 0 \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

for $j = 1, \ldots, n$, where the objective function value given by

$$z'(\lambda) = \sum_{j=1}^{n} c'_j x'_j(\lambda) + k \sum_{i=1}^{m} \lambda_i$$

is a lower bound to the optimal value of the original problem (1)–(3). The best lower bound $z'(\lambda^*)$ is the solution of the *Lagrangean dual problem* LDP:

$$z_D = \max_{\lambda \in \mathbb{R}^m_+} z'(\lambda). \tag{5}$$

Subgradient optimization may be used to solve (5). Subgradient algorithms may start from any feasible set of dual multipliers, such as $\lambda_i = 0$, for $i = 1, \ldots, m$, and iteratively generate further multipliers. We use the same strategy described in Held et al. [22] for updating the dual multipliers from one iteration to the next.

At any iteration q, let λ^q be the current vector of multipliers and let $x'(\lambda^q)$ be an optimal solution to problem $LRP(\lambda^q)$, whose optimal value is $z'(\lambda^q)$. Furthermore, let $\bar{z}$ be a known upper bound to the optimal value of problem (1)–(3). Additionally, let $g^q \in \mathbb{R}^m$ be a subgradient of $z'(\lambda)$ for $\lambda = \lambda^q$, with

$$g_i^q = k - \sum_{j=1}^{n} a_{ij} x'_j(\lambda^q), \quad i = 1, 2, \ldots, m. \tag{6}$$

To update the Lagrangean multipliers, the algorithm makes use of a step size

$$d^q = \frac{\eta\,(\bar{z} - z'(\lambda^q))}{\sum_{i=1}^{m} (g_i^q)^2}, \tag{7}$$

where $\eta \in (0, 2]$. Multipliers are then updated according to

$$\lambda_i^{q+1} = \max\{0; \lambda_i^q + d^q g_i^q\}, \quad i = 1, \ldots, m, \tag{8}$$

and the subgradient algorithm proceeds to iteration $q + 1$.

Beasley [21] reports as computationally useful to adjust the components of the subgradients to zero whenever they do not effectively contribute to the update of the multipliers, i.e. arbitrarily set $g_i^q = 0$ whenever $g_i^q > 0$ and $\lambda_i^q = 0$, for $i = 1, \ldots, m$.

The Lagrangean heuristic proposed in this section makes use of the dual multipliers λ^q and of the optimal solution $x'(\lambda^q)$ to each problem $LRP(\lambda^q)$ to build feasible solutions to the original problem (1)–(3). To do this, let $\mathcal{H}$ be a heuristic that builds a feasible solution x from an initial solution x^0.

Following Beasley [21], we set $x^0 = x(\lambda^q)$, i.e, $\mathcal{H}$ repairs the initial solution $x(\lambda^q)$ to make it feasible.

Heuristic $\mathcal{H}$ is initially applied from scratch using the original cost vector c. In any subsequent iteration q of the subgradient algorithm, $\mathcal{H}$ uses Lagrangean reduced costs $c'_j = c_j - \sum_{i=1}^{m} \lambda_i^q a_{ij}$ Let $x^{\mathcal{H},\gamma}$ be the solution obtained by $\mathcal{H}$, using a generic cost vector γ corresponding to the reduced cost or the original cost vector. Its cost is given by $\sum_{j=1}^{n} c_j x_j^{\mathcal{H},\gamma}$ and may be used to update the upper bound $\bar{z}$ to the optimal value of the original problem (1)–(3). This upper bound may be further improved by local search and is used to adjust the step size in (7).

Algorithm 1 describes the pseudo-code of the Lagrangean heuristic. Lines 2 to 4 initialize the upper and lower bounds, the iteration counter, and the dual multipliers. The iterations of the subgradient algorithm are performed along the loop in lines 5 to 20. The reduced costs are computed in line 6 and the Lagrangean relaxation problem is solved by inspection in line 7. In the first iteration of the Lagrangean heuristic, the original cost vector is assigned to γ in line 8, while in subsequent iterations the reduced cost vector is assigned in line 9. Lines 10 to 14 determine that a basic heuristic is used to produce a primal feasible solution to

```
1  **LagrangeanHeuristic**
2  Initialize bounds: $\bar{z} \leftarrow \sum_{j=1}^{n} c_j$ and $z_D \leftarrow 0$;
3  Initialize iteration counter: $q \leftarrow 0$;
4  Initialize dual multipliers: $\lambda_i^q \leftarrow 0,\ i = 1, \ldots, m$;
5  repeat
6      Compute reduced costs $c'_j \leftarrow c_j - \sum_{i=1}^{m} \lambda_i^q a_{ij},\ j = 1, \ldots, n$;
7      Solve $LRP(\lambda^q)$ by inspection to obtain $x'(\lambda^q)$;
8      if $q = 0$ then set $\gamma \leftarrow c$;
9      else set $\gamma \leftarrow c'_j$;
10     if $q$ is a multiple of $H$ then
11         Apply a basic heuristic $\mathcal{H}$ with cost vector $\gamma$ to obtain $x^{\mathcal{H},\gamma}$;
12         if $\sum_{j=1}^{n} c_j x_j^{\mathcal{H},\gamma} < \bar{z}$ then
13             $x^* \leftarrow x^{\mathcal{H},\gamma}$;
14             $\bar{z} \leftarrow \sum_{j=1}^{n} c_j x_j^{\mathcal{H},\gamma}$;
15     if $z'(\lambda^q) > z_D$ then $z_D \leftarrow z'(\lambda^q)$;
16     Compute subgradient: $g_i^q = k - \sum_{j=1}^{n} a_{ij} x'_j(\lambda^q),\ \ i = 1, 2, \ldots, m$;
17     Compute step size: $d^q \leftarrow \eta\ (\bar{z} - z'(\lambda^q)) / \sum_{i=1}^{m} (g_i^q)^2$;
18     Update dual multipliers: $\lambda_i^{q+1} \leftarrow \max\{0, \lambda_i^q - d^q g_i^q\}, i = 1, \ldots, m$;
19     Increment iteration counters: $q \leftarrow q + 1$;
20 until stopping criterion satisfied ;
```

Algorithm 1: Pseudo-code of the template for a Lagrangean heuristic.

problem (1)–(3) whenever the iteration counter q is a multiple of an input parameter H. A heuristic $\mathcal{H}$ is applied in line 11 to produce the feasible solution $x^{\mathcal{H},\gamma}$. If the cost of this solution is lower than the current upper bound, the best solution so far and its cost are updated in lines 13 and 14, respectively. If the lower bound $z'(\lambda^q)$ computed in iteration q is greater than the best lower bound z_D, then in line 15 the lower bound z_D is updated. Line 16 computes the subgradient and line 17 computes the step size. The dual multipliers are updated in line 18 and the iteration counter is incremented in line 19.

The Lagrangean heuristic proposed in this work makes use of two variants of $\mathcal{H}$.

The Greedy Basic Heuristic is composed by a non-randomized version of the GRASP construction procedure, described in Section 2.1. This heuristic builds a feasible solution to the original problem by repairing the solution of the *Lagrangean Dual Problem(LDP)*. In this phase, Lagrangean reduced costs are applied. Next, the local search described in Section 2.1 is applied to the resulting solution, using the original costs. We shall refer to the Lagrangean heuristic that uses the greedy heuristic as the *greedy Lagrangean heuristic* or simply GLH.

The GRASP Basic Heuristic is a slightly modified version of the GRASP with path-relinking. In the greedy randomized construction phase, instead of building a solution from scratch, the construction procedure starts with the solution of the Lagrangean relaxed problem. In the construction phase, Lagrangean reduced costs are used, while original costs are used in the local search and path-relinking phases.

Although the GRASP heuristic produces better solutions than the greedy heuristic, the latter is much faster. To appropriately address this trade-off, we choose to use the GRASP heuristic with probability β and the greedy heuristic with probability $1 - \beta$, where β is a parameter of the algorithm. We shall refer to the Lagrangean heuristic that uses the greedy and GRASP heuristic as the *GRASP Lagrangean heuristic* or simply LAGRASP.

We note that this strategy involves three main parameters: the number H of iterations after which the basic heuristic is always applied, the number Q of iterations performed by the GRASP with path-relinking heuristic when it is chosen as the basic heuristic, and the probability β of choosing the GRASP heuristic as $\mathcal{H}$. We shall refer to the Lagrangean heuristic that uses this hybrid strategy as LAGRASP(β, H, Q).

To implement path-relinking in the GRASP basic heuristic, LAGRASP maintains a global pool P which is empty at the start of the subgradient method. Each solution obtained in the GRASP local search phase is a candidate to be inserted in the elite set.

3. Computational Results

To carry out the experiments, 135 test problems were derived from the test instances classes 4, 5, 6, A, B, C e D from the OR-Library [23] by defining three different coverage factors k for each original instance.

- k_{min}: $k = 2$;
- k_{max}: $k = \min\limits_{i=1,\ldots,m} \sum\limits_{j=1}^{n} a_{ij}$;
- k_{med}: $k = \lceil (k_{min} + k_{max})/2 \rceil$

The computational experiments were performed on a 2.33 GHz Intel Xeon E5410 Quadcore computer running Linux Ubuntu 8.04. Each run was limited to a single processor. All codes were implemented in C and compiled with `gcc` 4.1.2.

Some metrics were applied to compare the methods. For each instance, *BestValue* is the overall best solution value obtained by all executions of the methods considered. Then, for each run of a method, *Dev* is the relative deviation in percentage between *BestValue* and the solution value obtained in that run. The average value of *Dev* over all instances and runs of a method in a particular experiment is reported as *AvgDev*. The metric *#Best*, for each method, is the number of runs whose solution value matched *BestValue*. For each method and instance, *NScore* gives the number of methods that found better solutions than this specific method for this instance. In case of ties, all methods receive the same score, equal to the number of methods strictly better than all of them. From *NScore* we derive, for each method, the metric *Score* as the sum of the *NScore* values across all the instances. Thus, lower values of *Score* correspond to better methods. We finally report, for each instance i, $\bar{t}_i$ is the average time across all runs in a particular experiment and *TTime* is the sum of $\bar{t}_i$ for all instances.

In this section, we report the computational experiments involving the hybridization of Lagrangean heuristic and GRASP with path-relinking. Computational results of these greedy Lagrangean heuristics and GRASP with path-relinking for SC$_k$P are described in detail in [13]. We now extend those experiments. Based on those experiments, the GRASP basic heuristic receives the solution x^0 provided by the Lagrangean relaxation. Lagrangean reduced costs are used to evaluate the candidate elements. Parameter α used in the GRASP construction phase was set

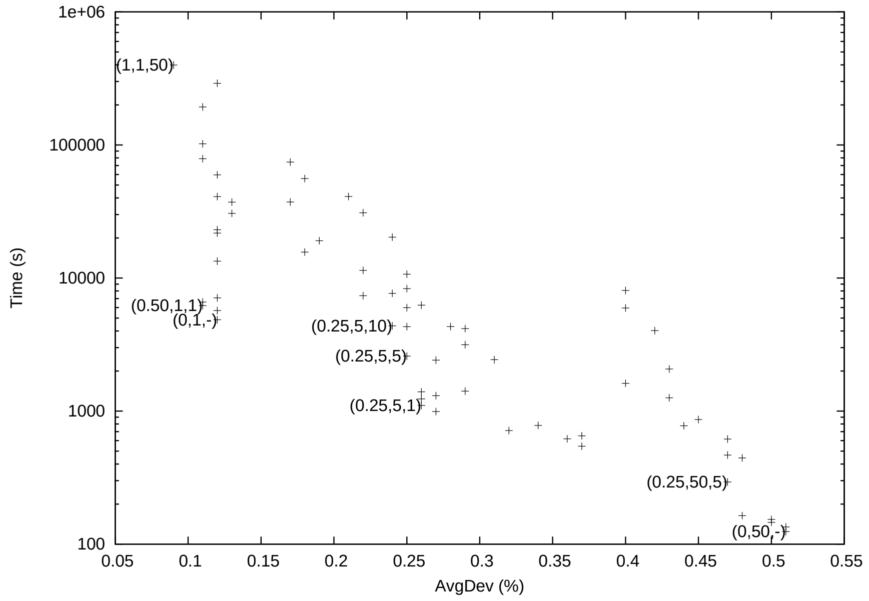

Figure 1. Average deviation from the best value and total running time for 68 different variants of LAGRASP: each point represents a unique combination of parameters β, H, and Q.

to 0.3. Following Beasley [21], LAGRASP stops whenever the lower bound matches the upper bound, or the step size parameter of the subgradient method becomes too small.

The aim of the first experiment is to evaluate the relationship between running times and solution quality for different parameter settings. Parameter H, the number of iterations between successive calls to the heuristic $\mathcal{H}$, was set to 1, 5, 10, and 50. Parameter β, the probability of GRASP being applied as the heuristic $\mathcal{H}$, was set to 0, 0.25, 0.50, 0.75, and 1. Parameter Q, the number of iterations carried out by the GRASP heuristic was set to 1, 5, 10, and 50. By combining these parameter values, 68 variants of LAGRASP were created. Each variant was applied eight times to each instance, with different initial seeds given to the random number generator. A subset of 21 instances was considered in this experiment.

The plot in Figure 1 summarizes the results for all evaluated variants, displaying points whose coordinates are the values of the AvgDev and TTime metrics for each combination of parameter values. Eight variants of special interest are identified and labeled with the corresponding parameters β, H, and Q, in this order. These variants correspond to some selected Pareto points in the plot in Figure 1, i.e, for a given AvgDev value there is no other variant which reaches the same value in less CPU time. Additionally, for a given CPU time, there is no other variant which shows a better result for the AvgDev metric in at most this CPU time. Setting $\beta = 0$ and $H = 1$ corresponds to the greedy Lagrangean heuristic (GLH) or, equivalently, to LAGRASP(0,1,-), whose average deviation from the best value amounted to 0.12% in 4,859.16 seconds of total running time. Table 1 shows the values of AvgDev and TTime for each selected variant.

Table 1. Summary of the numerical results obtained with the selected variants of the GRASP Lagrangean heuristic for 21 instances. Total time (TTime) is given in seconds.

Heuristic	AvgDev	TTime
LAGRASP(1,1,50)	0.09 %	399,101.14
LAGRASP(0.50,1,1)	0.11 %	6,198.46
LAGRASP(0,1,-)	0.12 %	4,859.16
LAGRASP(0.25,5,10)	0.24 %	4,373.56
LAGRASP(0.25,5,5)	0.25 %	2,589.79
LAGRASP(0.25,5,1)	0.26 %	1,101.64
LAGRASP(0.25,50,5)	0.47 %	292.95
LAGRASP(0,50,-)	0.51 %	124.26

Table 2. Summary of the numerical results obtained with the best variants of the GRASP Lagrangean heuristic for 135 instances. Total time (TTime) is given in seconds.

Heuristic	AvgDev	#Best	Score	TTime
LAGRASP(1,1,50)	0.079 %	365	74	1,803,283.64
LAGRASP(0.50,1,1)	0.134 %	242	168	30,489.17
LAGRASP(0,1,-)	0.135 %	238	169	24,274.72
LAGRASP(0.25,5,10)	0.235 %	168	320	22,475.54
LAGRASP(0.25,5,5)	0.247 %	163	350	11,263.80
LAGRASP(0.25,5,1)	0.249 %	164	405	5,347.78
LAGRASP(0.25,50,5)	0.442 %	100	625	1,553.35
LAGRASP(0,50,-)	0.439 %	97	666	569.30

In the following experiment, all the 135 test instances were considered in the comparison of the eight variants of LAGRASP selected above. Table 2 summarizes the results obtained by the eight variants. It shows that LAGRASP(1,1,50) found the best solutions, with their average deviation from the best values being as small as 0.079%. It also found the best known solutions in 365 executions, again with the best performance when the eight variants are evaluated side by side, although at the cost of the longest running times. On the other hand, the smallest running times were observed for LAGRASP(0,50,-), which was over 3000 times faster than LAGRASP(1,1,50) but found the worst-quality solutions.

In another experiment, we illustrate the merit of the proposed approach, showing how Lagrangean heuristic take advantage of the randomization aspect of its basic heuristic. We compare the eight selected versions of LAGRASP using graphics that show the evolution of lower and upper bounds over the SO iterations. We considered 135 instances for SC$_k$P. Since it is impractical to fit 135 plots in this paper, we show the entire collection of plots in `http://www.research.att.com/~mgcr/exp/sckp`. In this paper, we show only a representative set of plots.

Figures 2 to 4 display the typical behavior of LAGRASP for instances scpc4-k$_{min}$, scpd4-k$_{med}$, and scp43-k$_{max}$, respectively. We first observe that all variants

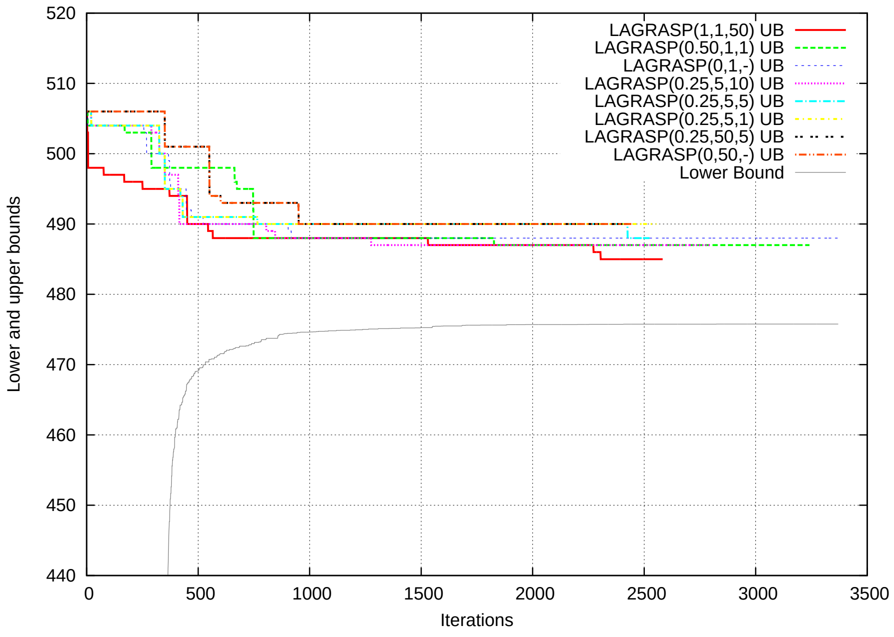

FIGURE 2. Evolution of lower and upper bounds over iterations for different variants of LAGRASP (scpc4-k$_{min}$ instance).

reach the same lower bounds, which is expected since they depend exclusively on the common subgradient algorithm. However, as the dual information (i.e., the lower bound) seems to stabilize, the upper bound obtained by LAGRASP(0,1,-) (or GLH) also seems to freeze. On the other hand, the Lagrangean heuristics based on GRASP continue to make improvements in discovering better upper bounds, since the randomized GRASP construction makes it possible to escape from locally optimal solutions and to find new, improved upper bounds. Only 19 out of 135 instances did not present the same behavior. In these cases, as shown in Figure 5 for instance scpa2-k$_{max}$, LAGRASP(0,1,-) (or GLH) overcame the results of the other versions of LAGRASP. These results, however, are in Table 1 and 2 and can be explained by the stochastic nature of LAGRASP.

4. Concluding Remarks

In this paper, we describe a hybrid Langrangean heuristic with GRASP and path-relinking, and extend the results presented in [13] for set k-covering. The comparison of different variants of LAGRASP showed that, by properly tuning its parameters, it is possible to obtain a good trade-off between solution quality and running time. Extensive experiments on 135 instances showed, graphically, that LAGRASP can take advantage of randomization aspect to make better use of dual information provided by subgradient optimization. As a consequence, LAGRASP is able to discover better solutions and to escape from locally optimal solutions after the stabilization of the lower bounds, whereas the greedy Lagrangean heuristic fails to find new improving solutions. The plots for the extended experiment, can be downloaded from `http://www.research.att.com/~mgcr/exp/sckp`.

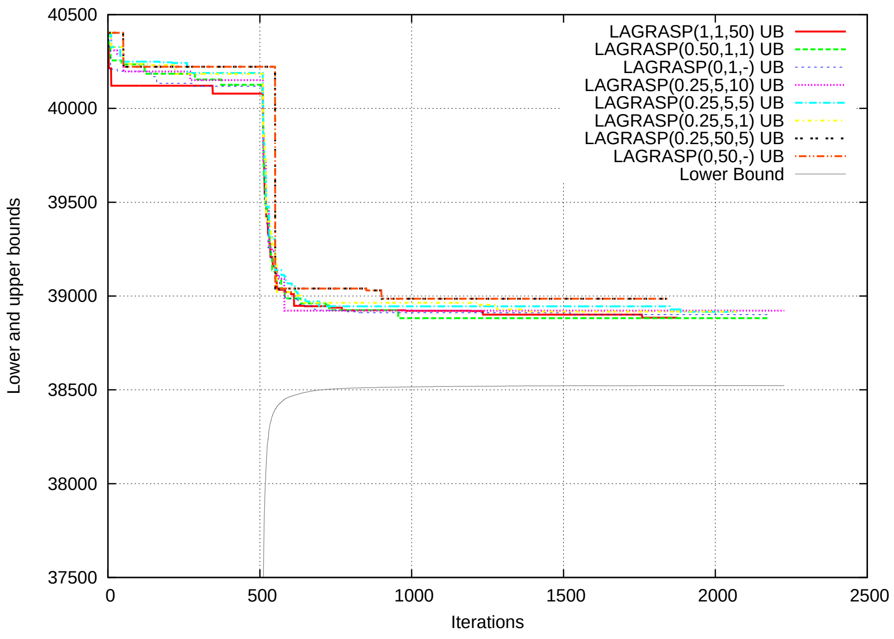

Figure 3. Evolution of lower and upper bounds over iterations for different variants of LAGRASP (scpd4-k$_{med}$ instance).

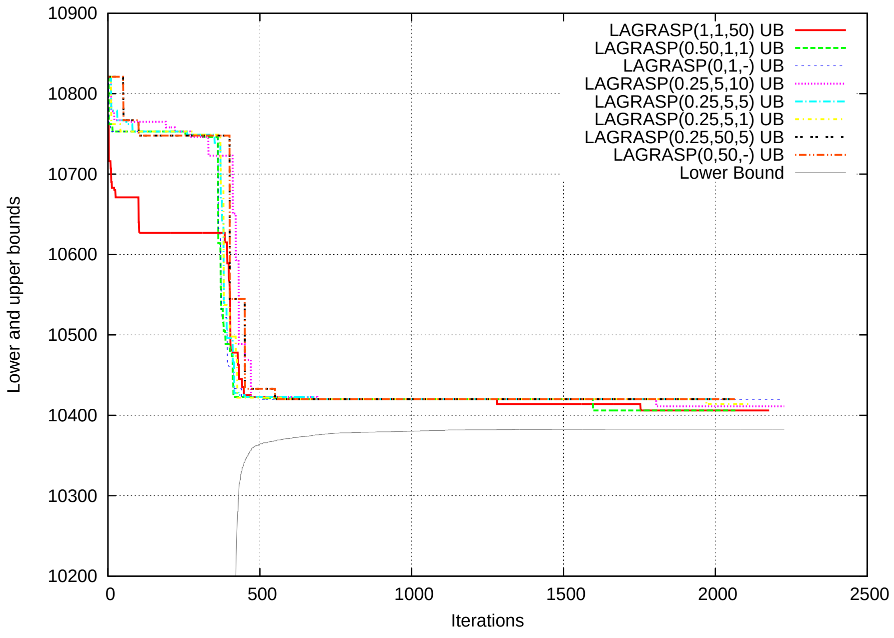

Figure 4. Evolution of lower and upper bounds over iterations for different variants of LAGRASP (scp43-k$_{max}$ instance).

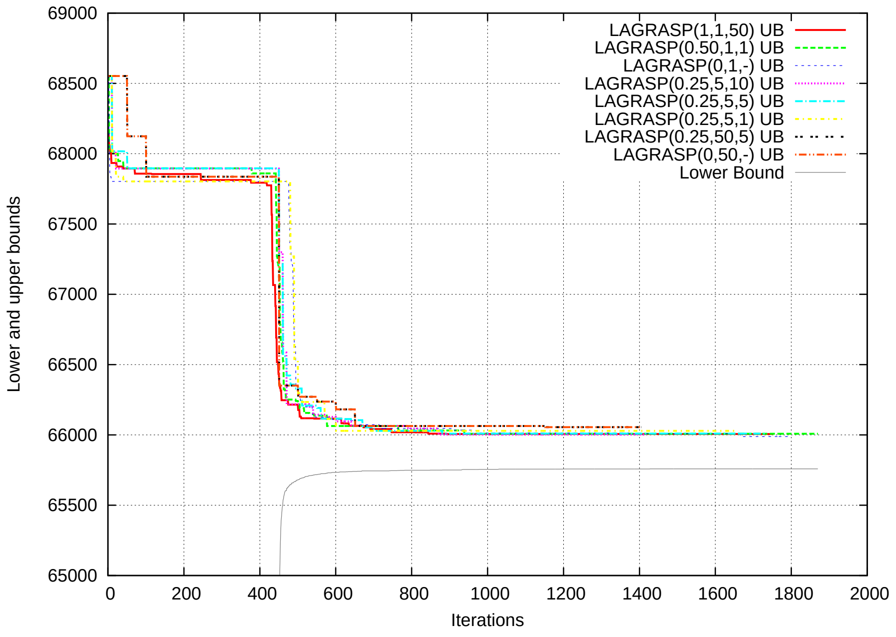

FIGURE 5. Evolution of lower and upper bounds over iterations for different variants of LAGRASP (scpa2-k$_{max}$ instance).

REFERENCES

[1] Vazirani, V.: Approximation Algorithms. Springer-Verlag, Berlin (2004)
[2] Bafna, V., Halldorsson, B. V., Schwartz, R., Clark, A.G., and Istrail, S.: Haplotypes and informative snp selection algorithms: don't block out information. In: RECOMB '03: Proceedings of the seventh annual international conference on Research in Computational Molecular Biology, pp. 19–27. ACM (2003)
[3] Resende, M.G.C.: An optimizer in the telecommunications industry. SIAM SIAG/Optimization Views-and-News 18(2), 8–19 (2007)
[4] Hall, N.G., Hochbaum, D.S.: The multicovering problem. European Journal of Operational Research 62(3), 323–339 (1992)
[5] Resende, M.G.C., Toso, R.T., Gonçalves, J.F., Silva, R.M.A.: A biased random-key genetic algorithm for the Steiner triple covering problem. Optimization Letters (2011). doi:10.1007/s11590-011-0285-3
[6] Breslau, L., Diakonikolas, I., Duffield, N., Gu, Y., Taghi, H.M., Johnson, D., Karloff, H., Resende, M.G.C., Sen, S.: Disjoint-Path Facility Location: Theory and Practice. In: Proceedings of the Thirteenth Workshop on Algorithm Engineering and Experiments (ALENEX), 60–74 (2011)
[7] Gonçalves, L., Martins, S.L., Ochi, L., Subramanian, A.: Exact and heuristic approaches for the set cover with pairs problem. Optimization Letters (2011). doi: 10.1007/s11590-011-0289-z
[8] Lucena, A. Private communication (2009)
[9] Avella, P., Boccia, M., Vasilyev, I.: Computational experience with general cutting planes for the Set Covering problem. Operations Research Letters 37, 16–20 (2009)
[10] Balas, E., Ng, S.: On the set covering polytope: I. All the facets with coefficients in 0,1,2. Mathematical Programming, Series A. 43, 57–69 (1989)
[11] Balas, E., Ng, S.: On the set covering polytope: II. Lifting the facets with coefficients in 0,1,2. Mathematical Programming, Series A. 45, 1–20 (1989)

[12] Pessoa, L.S., Ribeiro, C.C.: A GRASP for the minimum informative subset problem. In: CIRRELT (ed.) Proceedings of the Seventh Metaheuristics International Conference (MIC2007), 44–44 (2007)

[13] Pessoa, L.S., Resende, M.G.C., Ribeiro, C.C.: A hybrid Lagrangean heuristic with GRASP and path-relinking for set k-covering. Technical report, AT&T Labs Research (2010)

[14] Feo, T., Resende, M.G.C.: Greedy randomized adaptative search procedures. Journal of Global Optimization 6(2), 109–133 (1995)

[15] Resende, M.G.C., Ribeiro, C.C.: Greedy randomized adaptive search procedures: Advances and applications. In: Gendreau, M., Potvin, J. (eds) Handbook of Metaheuristics. 2nd edn, pp. 293–319. Springer Science+Business Media (2010)

[16] Beasley, J.E.: Lagrangean relaxation. In: Reeves, C.R. (ed.) Modern heuristic techniques for combinatorial problems. pp. 243–303. Blackwell Scientific Publications (1993)

[17] Fisher, M.: The Lagrangian relaxation method for solving integer programming problems. Management Science 50, 1861–1871 (2004)

[18] Held, M., Karp, R.M.: The traveling-salesman problem and minimum spanning trees. Operations Research 18, 1138–1162 (1970)

[19] Held, M., Karp, R.M.: The traveling-salesman problem and minimum spanning trees: Part II. Mathematical Programming 1, 6–25 (1971)

[20] Beasley, J.E.: An algorithm for set-covering problem. European Journal of Operational Research 31, 85–93 (1987)

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

[22] Held, M., Wolfe, P., Crowder, H.: Validation of subgradient optimization. Mathematical Programming 6, 62–88 (1974)

[23] Beasley, J.E.: OR-Library: Distributing test problems by electronic mail. Journal of the Operational Research Society 41, 1069–1072 (1990)

(Luciana S. Pessoa) Department of Informatics and Applied Mathematics, Universidade Federal do Rio Grande do Norte, Campus Universitário - Lagoa Nova, Natal, RN 59072-970 Brazil.
E-mail address: luciana@dimap.ufrn.br

(Mauricio G. C. Resende) Algorithms and Optimization Research Department, AT&T Labs Research, 180 Park Avenue, Room C241, Florham Park, NJ 07932 USA.
E-mail address: mgcr@research.att.com

(Celso C. Ribeiro) Department of Computer Science, Universidade Federal Fluminense, Rua Passo da Pátria, 156, Niterói, RJ 24210-240 Brazil.
E-mail address: celso@ic.uff.br