Effective Heuristics for the Set Cover by Pairs Problem

Luciana Brugiolo Gonçalves[a], Simone de Lima Martins[a], Luiz Satoru Ochi[a] and Mauricio G. C. Resende[b]

[a]*Universidade Federal Fluminense, Instituto de Computação, Passo da Pátria 156 - Bloco E / 350 - 24210-240. Niterói, Brasil.* {lgoncalves,simone, satoru}@ic.uff.br
[b]*AT&T Labs Research, 180 Park Avenue, Florham Park NJ 07932.* mgcr@research.att.com

Keywords: Set Cover by Pairs, Heuristic, Combinatorial Optimization.

Abstract. This paper deals with the Set Cover by Pairs Problem (SCPP). This problem is a generalization of the Set Cover Problem (SCP), which, in turn, is known to be NP-hard. The difference between both problems is that, in the SCPP, one element of the universe is admitted to be covered if there are at least two objects chosen to cover it. In this context, three constructive heuristics and one local search algorithm are proposed. The algorithms developed were tested in 560 instances available in the literature and they were capable of achieving optimal solutions for several instances.

1 Introduction

As introduced in (Hassin and Segev, 2005), the Set Cover by Pairs Problem (SCPP) is considered a generalization of the set cover problem, in which elements are covered by pairs of objects. The objective is to find a minimum cost subset of objects that induces a collection of pairs covering all elements. Formally, let U be a ground set of elements and let A be a set of objects, where each object i has a non-negative cost w_i. For every $\{i, j\} \subseteq A$, let $C(i, j)$ be the collection of elements in U covered by the pair $\{i, j\}$. The Set Cover by Pairs problem asks to find a subset $S \subseteq A$ such that $\bigcup_{\{i,j\} \subseteq S} C(i, j) = U$ and such that $\sum_{i \in S} w_i$ is minimized.

A set cover instance, with $U = \{e1, ..., en\}$ and $A_1, ..., A_m \subseteq U$, can be interpreted as an SCPP instance by setting $C(i, j) = A_i \cup A_j$ for every $i = j$. Therefore, the SCPP is a generalization of the set cover problem. The hardness results regarding set cover extend to SCPP which, in turn, is known to be NP-hard (Garey and Johnson, 1990).

An application that can be shaped as SCPP is Minimum Monitoring Set (MMS), introduced in (Breslau et al., 2007). A problem in network host placement that has its origin in a monitoring problem in communications networking. In this application, it is desired to conduct end-to-end network performance measurements between each of a given set of target nodes and pairs of measurement nodes, under the constraint that the pair of paths from each target to the corresponding pair of measurement nodes are node disjoint.

In this context, the MMS problem can be formalized as follows. Consider a graph $G = (V, E)$ which represents the physical topology of the network, where V is the set of vertices and E is a set of edges. Consider a set $R(u, v)$ of simple paths with initial node u and final node v, where $u, v \in V$. Let u, v, k be distincts in V, $\{u, v\}$ is a pair cover for k if for each pair of routes $(r_1, r_2) \in R(k, u) \times R(k, v)$, r_1 and r_2 are node-disjoint except for their initial node k, i.e., $r_1 \cap r_2 = \{k\}$. Therefore, with $B, S \subset V$, S is a cover of B, if for each $b \in B$ there exist $m_1, m_2 \in S$ such that $\{m_1, m_2\}$ is a pair cover for b.

Thus, B corresponds to the set of routers (or equivalently, branch points) and S a set of locations for measurement hosts. In practice, we wish to be able to perform tomographic measurements to each router using a minimal deployment of measurement hosts chosen from some

set M of possible locations. This motivates the Minimum Monitoring Set (MMS) problem, where given $B, M \subset V$, find $S \subseteq M$ of minimum size that is a cover of B.

This paper deals with the MMS problem considering that each measurement node $i \in M$ has a non-negative cost w_i, and the goal is to find the subset $S \subseteq M$ wich covers B such that $\sum_{i \in S} w_i$ should be minimized.

As detailed in (Breslau et al., 2007), the MMS can be seen as a special case of SCPP. In order to understand this relationship between the two problems, let $U = B$ and $A = M$. The pair $\{u, v\}$, where $u, v \in M$, covers $b \in B$ if and only if no path in $R(b, u)$ has a vertex other than b in common with any path in $R(b, v)$.

The first proposal for the SCPP was submitted in (Hassin and Segev, 2005), which presented a greedy algorithm obtained from the algorithm of (Chvátal, 1979) for the Set Cover Problem.

In (Breslau et al., 2007) some approaches were made to the SCPP: a greedy algorithm, that is a simplification of the algorithm of (Hassin and Segev, 2005); a Genetic Algorithm that explores the idea of random key (Bean, 1994) and exact approaches to the problem. In addition, the Double Hitting Set algorithm was developed specifically for the MMS problem that exploits typical structures of OSPF routing tables.

The paper is organized as follows. Section 2 deals with the three proposed constructive algorithms. In Section 3, the local search algorithm is described. Section 4 presents the results obtained and a comparison with the results found in the literature. Finally, in Section 5 concluding remarks are presented.

2 Constructive heuristic

The cost of selecting a pair $\{m_1, m_2\} \in M$ to participate of the solution S is calculated by $increase\_solution(m_1, m_2) \in [0, cost(m_1) + cost(m_2)]$. If any of the pair nodes belongs to S, there will have no increase in the cost of the solution by inserting it. The maximum increase in the solution cost will occur when both nodes do not belong to the solution.

2.1 Best Pair Heuristic

This heuristic is based on the cheapest insertion heuristic. The algorithm chooses, in each iteration, a pair of mensurement nodes better evaluated for a given function.

The algorithm starts with an empty set of measurement nodes S. Then, the branch nodes are sorted in increasing order according to the number of possible pairs of measurement nodes that may cover them. Thereafter, as long as S does not cover all branch nodes, a not covered branch node b is selected according to the established order. A pair of measurement nodes m_1 and m_2, which covers b, is selected in order that $S \cup \{m_1, m_2\}$ covers a maximum number of additional branch nodes and whose insertion causes the lowest increase in the solution cost. The function which evaluates each pair can be seen in the Equation (1), where $coverage(m_1, m_2)$ represents the number of new branch nodes that will be covered by (m_1, m_2) and $increase\_solution(m_1, m_2)$ is the increase in the cost solution.

$$f(m_1, m_2) = coverage(m_1, m_2)/increase\_solution(m_1, m_2) \quad (1)$$

2.2 Simplified Heuristic

This algorithm is a simplification of the greedy algorithm introduced by (Hassin and Segev, 2005). In this approach, in each iteration each measurement node $m \in M - S$ is evaluated according to the number of new pairs that will be completed by inserting it, and the number of new branch nodes that can have m as one element of a cover pair, but do not have the other pair

element associated to it inserted in S. Let B' be the set of branch nodes covered if m is selected and B" the set of not covered branch nodes, where m is one element of a possible cover pair, but the other pair element has not been yet inserted in the solution. In Equation 2, m is evaluated according to the number of elements of the sets B' and B". To each set is associated a weight p_1 and p_2, and is also considered the cost of node m.

$$f(m) = (p_1 \times |B'| + p_2 \times |B"|)/cost(m) \tag{2}$$

The algorithm starts with an empty set of measurement nodes S. Then, as long as the set solution S does not cover all elements of B, at each iteration a measurement node m ($m \in M - S$) is inserted in S. In order to select the next element m to be inserted, the candidates are sorted in descending order according to the function f described in the Equation 2. The best measurement node is included in the solution.

2.3 Percentual Heuristic

The structure of this heuristic is the same of the previous algorithm. The diference is the function which evaluates the inclusion of measurement nodes.

Consider a measurement node $m \in M - S$ which belongs to pairs that can cover the subset of not covered branch nodes B_m. To evaluate the node m, it is considered how m can help to cover the elements of B_m. Therefore, the Equation 3 is used to evaluate each node m, where, for each $b_m \in B_m$, the percentage of cover pairs that m participates is calculated.

$$g(m) = \left(\sum_{b \in B_m} \textit{percentage of participation}(m, b) \right) \Big/ cost(m) \tag{3}$$

For example, consider the subset $B_m = \{b_5, b_7\}$. If a node m participates of half of the possible pairs that can cover b_5 and $1/4$ of those that can cover b_7, then the value of the function for this node is $g(m) = (0.50 + 0.25)/cost(m)$. The measurement node with greater value associated with this function is the next node to be included in the solution. The algorithm ends when all branch nodes are covered.

3 Local Search

The local search algorithm starts with a solution obtained by one of the constructive algorithms and then, some of the initial solution nodes are removed and the solution is rebuilt.

The pseudocode is described in Alg. 1, where S corresponds to the initial solution, $percent$ indicates the number of elements that will be removed from the solution and $withoutImproves$ the number of iterations without improvement that is allowed by the algorithm.

The first stage of the algorithm removes randomly $(|S| \times percent/100)$ nodes of the initial solution, creating a infeasible solution S' (where $0 \leq percent \leq 100$). Then, to make the solution feasible, the reconstruction is carried out inserting nodes not belonging to S', using the constructive heuristics presented in the previous section.

If the new solution has a lower cost than the best known solution ($bestSolution$), then the $bestSolution$ is updated. The algorithm ends when $withoutImproves$ iterations are carried out without any gain.

4 Computational Results

The proposed algorithms were implemented in C++, compiled in g++ version $4.1.3$ and executed in a PC Intel(R) Core(TM) 2 Duo T7250 with 2GHz CPU and 2GB RAM.

Algorithm 1 Local Search(S, $percent$, $withoutImproves$)

```
bestSolution ← S
countImproves ← 0
while ( countImproves < withoutImproves) do
   S' ← remove percent elements of bestSolution
   S ← rebuild solution S' using a constructive heuristic
   if ( cost(S) < cost(bestSolution) ) then
      bestSolution ← S
      countImproves ← 0
   else
      countImproves ++
   end if
end while
return bestSolution
```

To evaluate the heuristics, 560 instances introduced in (Breslau et al., 2007) were used. The instances were generated with 26, 50, 100, 190, 220, 250, 300 and 558 nodes, all considering that each element of M has the same cost. These 560 instances were separated in seven groups varying the sets of branch and measurement nodes. In four of these groups, it was assumed that the set of potential measurement nodes is the entire set of nodes and different branch node sets were generated using 12.5%, 25% , 50% and 100% of the nodes. The other three groups have $|B| = |M|$, where B is a set consisting of 12.5%, 25% and 50% nodes randomly chosen. To identify the groups, for positive integers (m, b) the notation (Mm_Bb) is used to refer to classes of instances which measurement nodes form a fraction $1/m$ of the total nodes, and branch nodes form a fraction $1/b$ of the total nodes.

In the local search algorithm, the value of parameter $withoutImproves$ was set at 5 and the $percent$ was set at 30. These parameters were adjusted empirically after preliminary tests. In the same way, in Simplified Heuristic the parameter p_1 was set at 2 and p_2 at 1.

Table 1 shows a comparison between the solutions obtained by each heuristic (Best Pair Heuristic-BPH, Simplified Heuristic-SH and Percentual Heuristic-PH) using or not using the Local Search-LS. These results are compared to the GA heuristics proposed by (Breslau et al., 2007). We choosed the GA among all heuristics proposed by (Breslau et al., 2007) because it provides the best results. For each instance, the metric $|S|/|B|$ is presented, which is the ratio of the number of nodes used to cover the set B to the size of B. In this table, the column labelled by G indicates the group of instances, IP the metric associated with optimal solution obtained by (Breslau et al., 2007) and $Best$ is the best result known for those instances where the optimal solution is not known.

For most of instances, the performance of the algorithms was similar. Therefore, in this work is displayed only a subset of 37 instances where it is possible to see differences between the results of the algorithms. When the algorithm reaches the best solution ($Best$ or IP) an "*" is shown.

From Table 1, it is possible to see that the AG algorithm was unable to reach the best solution in 18 instances, while the pure (not using local search) heuristics BPH, PH and HS did not obtain the best solution in 12, 5 and 17 instances respectively. The use of the local search algorithm was not effective when applied to heuristic PH, but was able to significantly increase the number of best solutions achieved by heuristics BPH and HS.

Regarding to the demanded computational effort, the average time of the heuristics BPH, PH

and HS for the instances at the Table 1 are respectively 302.16, 330.85 and 226.57 seconds. The use of local search increases the running time in only 85 seconds, as the heuristics generated good initial solutions. There is no mention in (Breslau et al., 2007) about the machine used, so it was not possible to compare with the AG processing time.

G	Instance	IP	Best	AG	BPH	BPH+LS	PH	PH+LS	SH	SH+LS
M1-B1	n26-i1-m26-b26	0.192	-	0.231	*	*	*	*	*	*
M1-B1	n100-i9-m100-b100	0.300	-	0.310	*	*	*	*	*	*
M1-B1	n250-i8-m250-b250	?	0.212	*	*	*	0.216	0.216	*	*
M1-B1	n250-i9-m250-b250	?	0.220	*	0.224	*	*	*	0.224	*
M1-B1	n300-i0-m300-b300	?	0.223	*	*	*	*	*	0.227	0.227
M1-B1	n300-i2-m300-b300	?	0.220	*	0.223	*	*	*	0.223	0.223
M1-B1	n300-i6-m300-b300	?	0.243	*	0.250	*	*	*	0.247	*
M1-B1	n300-i8-m300-b300	?	0.207	*	*	*	*	*	0.210	0.210
M1-B1	n300-i9-m300-b300	?	0.223	*	0.227	*	*	*	0.227	*
M1-B1	n558-i0-m558-b558	?	0.206	0.208	*	*	*	*	0.208	*
M1-B1	n558-i2-m558-b558	?	0.197	*	0.201	*	0.201	0.201	0.199	*
M1-B1	n558-i3-m558-b558	?	0.185	0.186	*	*	*	*	*	*
M1-B1	n558-i4-m558-b558	?	0.199	0.201	0.201	0.201	*	*	0.203	*
M1-B1	n558-i6-m558-b558	?	0.194	0.195	0.197	0.195	0.195	0.195	0.197	*
M1-B1	n558-i8-m558-b558	?	0.190	*	0.192	*	*	*	0.192	0.192
M1-B1	n558-i9-m558-b558	?	0.195	*	0.199	0.197	0.197	0.197	*	*
M1-B2	n26-i1-m26-b13	0.308	-	0.385	*	*	*	*	*	*
M1-B2	n50-i7-m50-b25	0.320	-	0.360	*	*	*	*	*	*
M1-B2	n558-i2-m558-b279	?	0.290	0.294	*	*	*	*	*	*
M1-B2	n558-i3-m558-b279	?	0.280	*	*	*	*	*	0.283	0.283
M1-B2	n558-i5-m558-b279	?	0.269	0.272	*	*	*	*	*	*
M1-B4	n50-i7-m50-b13	0.462	-	*	*	*	*	*	0.538	0.538
M1-B4	n100-i6-m100-b25	0.480	-	*	*	*	*	*	0.520	0.520
M1-B4	n190-i8-m190-b48	?	0.563	0.583	*	*	*	*	*	*
M1-B4	n300-i1-m300-b75	?	0.453	0.467	*	*	*	*	*	*
M1-B4	n558-i6-m558-b140	?	0.393	0.400	*	*	*	*	*	*
M1-B4	n558-i8-m558-b140	?	0.421	0.429	*	*	*	*	*	*
M1-B8	n26-i9-m26-b4	0.750	-	*	*	*	*	*	1.000	1.000
M1-B8	n50-i8-m50-b7	0.571	-	*	*	*	*	*	0.714	0.714
M1-B8	n300-i1-m300-b38	?	0.605	0.632	*	*	*	*	*	*
M1-B8	n558-i9-m558-b70	?	0.586	0.600	*	*	*	*	*	*
M2-B2	n100-i3-m50-b50	0.360	-	*	0.380	*	*	*	*	*
M2-B2	n558-i3-m279-b279	?	0.283	*	0.287	0.287	0.287	0.287	*	*
M2-B2	n558-i5-m279-b279	?	0.276	*	*	*	*	*	0.280	0.280
M2-B2	n558-i8-m279-b279	?	0.294	*	0.297	*	*	*	*	*
M4-B4	n26-i4-m7-b7	0.571	-	0.714	*	*	*	*	*	*
M8-B8	n50-i7-m7-b7	0.571	-	0.714	*	*	*	*	*	*

Table 1: $|S|/|B|$

For a better evaluation of heuristics, a second set of instances was used, where the costs of the elements of M are varied. These instances are based on the instances proposed by (Breslau et al., 2007), so that two new instances were created from one original instance. The first was generated by randomly selecting in the interval [1,100] a cost w_m to each element $m \in M$. For the second, the interval [1,1000] was used. The instances were also grouped into seven groups as presented before. For each instance, the gap was calculated between the best solution and the solution obtained by each heuristic. Table 2 presents the average of the results and the average of the CPU processing time (seconds) for each group. For this set of instances, only the results obtained for the proposed heuristics are shown because there are no results for the others heuristics in the literature. These instances are available in *http://labic.ic.uff.br/conteudo/instance/covering-by-pairs/*.

The heuristic BPH reached the lowest average gap, followed by heuristic PH. The heuristic SH was unable to obtain good results, on average it is 21.7% far form the best solution. In this table, it is also possible to verify the impact of the use of local search algorithm. For most of these instances, the algorithm was able to reduce the cost of solutions.

Groups	BPH		BPH+LS		PH		PH+LS		SH		SH+LS	
	GAP	t (sec)	GAP	t (sec)	GAP	t (sec)	GAP	t (sec)	GAP	t (sec)	GAP	t (sec)
M1_B1	0.004	175.42	0.001	266.89	0.006	218.10	0.002	310.71	0.223	323.00	0.005	470.39
M1_B2	0.003	76.23	0.001	118.28	0.006	91.68	0.003	136.13	0.342	142.55	0.002	222.32
M1_B4	0.002	32.86	0.002	47.84	0.006	31.69	0.004	48.85	0.391	57.81	0.003	90.83
M1_B8	0.003	9.78	0.001	14.26	0.003	8.58	0.001	11.83	0.383	15.25	0.001	25.72
M2_B2	0.003	6.27	0.002	9.01	0.005	8.91	0.003	12.67	0.107	11.59	0.003	16.65
M4_B4	0.001	0.25	0.001	0.40	0.002	0.40	0.001	0.41	0.046	0.30	0.001	0.54
M8_B8	0.000	0.05	0.000	0.08	0.003	0.04	0.000	0.07	0.030	0.06	0.000	0.10
Average	0.002	42.98	0.001	65.25	0.005	51.34	0.002	74.38	0.217	78.65	0.002	118.08

Table 2: Results obtained in new instances.

5 Conclusions

In this work, three constructive heuristics and a local search algorithm for the Set Cover by Pairs Problem were presented. To analyze the performance of the proposed algorithms, two sets of instances were used. The first set is composed of 560 unit cost instances proposed by (Breslau et al., 2007), where it was shown that the heuristics were competitive and capable of improving 9 of the best known solutions. The best results were achieved by Best Pair Heuristic and Percentual Heuristic, but Simplified Heuristic was responsible for improving the solution of 3 instances. The second group of instances was proposed in this paper and have 1120 instances with varied nodes cost. For this group of instances, it is possible to verify that the heuristics BPH+LS and PH+LS obtained better results, highlighting the gain attributed to the algorithm of local search.

References

Bean, J. C.: 1994, Genetics and random keys for sequencing and optimization, *ORSA Journal on Computing* **6**, 154–160.

Breslau, L., Diakonikolas, I., Duffield, N., Gu, Y., Hajiaghayi, M., Johnson, D., Karloff, H., Resende, M., Sen, S. and Towsley, D.: 2007, Optimal node placement for path disjoing network monitoring, *Technical report*, AT&T Labs Research.

Chvátal, V.: 1979, A greedy heuristic for the set-covering problem, *Mathematics of Operations Research* **4**, 233–235.

Garey, M. R. and Johnson, D. S.: 1990, *Computers and Intractability; A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., New York, NY, USA.

Hassin, R. and Segev, D.: 2005, The set cover with pairs problem, *FSTTCS 2005: Proceedings of the 25th International Conference on Foundations of Software Technology and Theoretical Computer Science*, Vol. 3821, Springer Berlin / Heidelberg, pp. 164–176.