Effective Heuristics for the Minimum Monitoring Set 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: Network Measurement, Set Cover by Pairs, Heuristics, Combinatorial Optimization.

Abstract. This paper deals with the Minimum Monitoring Set Problem (MMSP), which consists of placing measurement nodes in a network to be able to conduct end–to-end network performance measurements between each of a given set of target nodes. Each target node should be covered by a pair of measurement nodes, under the constraint that the paths between the target node and each of the pair nodes should be node disjoint. Besides, the routes among network nodes may not be arbitrarily defined and should obey standard network routing policies, such as OSPF. This problem can be viewed as a special case of the Set Cover with Pairs Problem, which was introduced as a non-trivial generalization of the Set Cover Problem (SCP), that, in turn, is known to be NP-hard. 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

Gu et al. (Gu et al., 2007) proposed techniques for estimating one-way loss from a measurement host to network routers which exploit commonly implemented features on commercial routers and do not require any new router capabilities.

The Minimum Monitoring Set Problem (MMSP) is based on these techniques, and consists of finding pairs of measurement nodes for a given set of target 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 MMSP 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 distinct 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 represents the set of locations for measurement hosts. In practice, we wish 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 Problem (MMSP), where given $B, M \subset V$, find $S \subseteq M$ of minimum size that is a cover of B.

In Section 2, as shown in (Breslau et al., 2007), we note that this problem can be viewed as a special case of the Set Cover with Pairs (SCPP) Problem (Hassin and Segev, 2005), that 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.

In (Breslau et al., 2007), some algorithms were proposed for the SCPP, namely: a greedy algorithm, which is a simplification of the algorithm proposed by (Hassin and Segev, 2005); a Genetic Algorithm based on the random key method (Bean, 1994); and some exact approaches for the problem. In addition, the Double Hitting Set algorithm was developed specifically for the MMSP that exploits typical structures of OSPF routing tables.

In this work, we propose three new constructive algorithms, which are shown in Section 3 and one new local search algorithm described in Section 4. Section 5 contains the results obtained using these new heuristics and a comparison with the results found in the literature. Finally, Section 6 presents the concluding remarks of this work.

2 The Minimum Monitoring Set Problem

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 $U = \{e_1, ..., e_n\}$ and $A_1, ..., A_m \subseteq U$, can be interpreted as Set Cover with Pairs instance by setting $C(i,j) = A_i \cup A_j$ for every $i = j$. Therefore, the Set Cover with Pairs Problem is a generalization of the Set Cover Problem. The hardness results regarding SCP extend to SCPP which, in turn, is known to be NP-hard (Garey and Johnson, 1979).

The first algorithm proposed for the SCPP (Hassin and Segev, 2005) was a greedy algorithm obtained from the algorithm developed in (Chvátal, 1979) for the Set Cover Problem.

For the MMSP, 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. $B \subseteq V$ corresponds to the set of branch points and $S \subseteq V$ represents the set of locations for measurement hosts. The objective of the Minimum Monitoring Set Problem (MMSP) is to find $S \subseteq M$ of minimum size that is a cover of B.

In order to understand the relationship between MMSP and SCPP, 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 $R(b, v)$.

Breslau at al. (Breslau et al., 2007) modeled the Set Cover by Pairs Problem as a simple mixed integer linear program (MIP) as shown below.

The following notation is used:

Constants

V = set of vertices
E = set of edges
B = set of branch nodes
M = set of measurement nodes

Variables

$$x_v = \begin{cases} 1, & \text{if node measurement } v \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

Using this notation, the SCPP may be formulated as follows:

$$\min \sum_{v=1}^{m} x_v \tag{1}$$

Subject to:

$$y_{u,v} \leq x_u; \forall u, v \in M, u < v, \tag{2}$$

$$y_{u,v} \leq x_v; \forall u, v \in M, u < v, \tag{3}$$

$$\sum y_{u,v} \geq 1; \forall (u, v) \in M, \text{that covers } b \in B - M, u < v \tag{4}$$

$$x_b + \sum y_{u,v} \geq 1; \forall (u, v) \in M, \text{that covers } b \in B \cap M, u < v \tag{5}$$

$$x_v \in \{0, 1\}, \forall v \in M \tag{6}$$

$$y_{u,v} \geq 0, \forall u \in M, \forall v \in M \tag{7}$$

The objective function (1) minimizes the number of selected measurement nodes. Constraints (2) and (3) guarantee that $y_{u,v}$ is positive only if x_u and x_v are in the chosen set. Constraints (4) state that for each branch node b, which is not a potential measurement node ($b \in B - M$), there should be at least one pair (u, v) that covers b. Constraints (5) state that for each branch node b, which is a potential measurement node ($b \in B \cap M$), the node b should be selected as its monitor node or there should be at least one pair (u, v) that covers b.

In (Breslau et al., 2007) MMSP instances were treated with this formulation and solved with CPLEX linear and mixed integer linear programming solver. Optimal solutions were found for all instances with $|M| \leq 100$.

3 Constructive heuristics

In this section we describe the three constructive heuristics developed for MMSP.

The first one is based on selecting a pair of measurement nodes at each construction step and the other two are based on selecting one measurement node at each construction step.

3.1 The Best Pair Heuristic (BPH)

This heuristic is based on the cheapest insertion strategy. At each iteration, the algorithm chooses the best pair of measurement nodes for a given evaluation function.

The algorithm starts with an empty set of measurement nodes S. Next, the branch nodes are sorted in ascending order according to the number of possible pairs of measurement nodes that may cover them. Thereafter, while S does not cover all branch nodes, a not yet covered branch node b is selected according to the established order. A Restricted Candidate List (RCL) for b is created by ordering the list of the measurement nodes pairs (m_i, m_j), which covers b, in descending order according to the function $f()$ shown in the Equation (8), where $coverage(m_i, m_j)$ represents the number of new branch nodes that will be covered by (m_i, m_j)

and $increase\_solution(m_i, m_j) \in [0, cost(m_1) + cost(m_2)]$ is the increase in the cost solution by inserting the pair (m_i, m_j). If one of the nodes of the pair or both already belongs to S, the cost of the solution will not increase by inserting it. The maximum increase in the solution cost will occur when both nodes do not belong to the solution.

The size of this list is restricted by selecting the first α% elements of the list. Then, a pair of measurement nodes m_1 and m_2, which covers b, is randomly selected from the RCL.

$$f(m_1, m_2) = coverage(m_1, m_2)/increase\_solution(m_1, m_2) \tag{8}$$

Algorithm 1 Best Pair Heuristic(M,B,α)

```
S ← { };
BL ← Sort_Branch_Nodes(B);
while (S not covers all branch nodes in B) do
    b ← Select_Next_Uncovered_Branch (BL)
    RCL ← Create_List_Monitor_Pairs( M,b, f(), $\alpha$ )
    $\{m_1, m_2\}$ ← Choose_Monitor_Pair_Randomly ( RCL )
    S ← S ∪$\{m_1, m_2\}$
    Update_Branch_List (BL)
end while
return S
```

3.2 The Best Monitor Heuristic (BMH)

This algorithm is a simplification of the greedy algorithm introduced in (Hassin and Segev, 2005). In this approach, in each iteration, every measurement node $m \in M - S$ is evaluated according to the number of new pairs that will be completed by inserting it, as well as 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 with it already 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 yet been inserted in the solution.

In Equation 9, m is evaluated according to the number of elements of the sets B' and B'' and the cost of node m. The weights p_1 and p_2 are associated with each set.

$$f(m) = (p_1 \times |B'| + p_2 \times |B''|)/cost(m) \tag{9}$$

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 $m \in M - S$ 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 9. Then, the first $\alpha\%$ candidates are chosen to form a Restricted Candidate List and a monitor m is randomly selected from this list. This algorithm is described in Alg. 2.

3.3 The Percentage Heuristic (PH)

The structure of this heuristic is similar to that of the Best Monitor Heuristic. The difference is the function which evaluates the inclusion of measurement nodes.

Algorithm 2 Best Monitor Heuristic(M,B,α)

```
S ← { }
while (S not covers all branch nodes in B) do
  RCL ← Create_List_Monitor(M,f(), $\alpha$)
  $m$ ← Choose_Monitor_Randomly ( RCL )
  S ← S ∪{$m$}
end while
return S
```

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, we consider how m can contribute to cover the elements of B_m. Equation 10 is used to evaluate each node m, where, for each $b_m \in B_m$, the percentage of cover pairs in which m participates is computed as:

$$g(m) = \left(\sum_{b \in B_m} \textit{percentage of participation}(m, b) \right) \Big/ \, cost(m). \tag{10}$$

For example, consider the subset $B_m = \{b_5, b_7\}$. If a node m is one of the elements in 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 nodes are ordered in descending order according to this function, and one node is randomly selected from the α% of these candidates and is included in the solution.

The algorithm ends when all branch nodes are covered.

4 Local Search

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

The algorithm is described in Alg. 3, where S corresponds to the initial solution, $percent$ indicates the number of elements that will be removed from the solution and $withoutImproves$ is the number of iterations without improvement that is allowed by the algorithm.

Algorithm 3 Local Search(M,B,S, $percent$, $withoutImproves$)

```
bestSolution ← S
countImproves ← 0
while (countImproves < $withoutImproves$) do
  $S'$ ← Remove_Elements_of_Solution($percent$,bestSolution)
  $S$ ← Rebuild_Solution ($S'$)
  if ( cost(S) < cost(bestSolution) ) then
    bestSolution ← S
    countImproves ← 0
  else
    countImproves ++
  end if
end while
return bestSolution
```

The first stage of the algorithm randomly removes $(|S| \times percent/100)$ nodes of the initial solution, creating an 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.

5 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.

To evaluate the heuristics, 560 instances introduced in (Breslau et al., 2007) were used. These heuristics were generated based on network structures and routing policies with the intent of mimicking real networks and applications.

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 size of the branch nodes and measurement nodes sets. 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 to 5 and the parameter $percent$ was set to 30. These parameters were adjusted empirically after preliminary tests. In the same way, in the Best Monitor Heuristic the parameter p_1 was set to 2 and p_2 to 1.

The first conducted experiment was executed using greedy constructive procedures. For all three constructive heuristics, the first element of the Restricted Candidate List is selected to be in the solution.

Table 1 shows a comparison between the solutions obtained by each greedy heuristic (GBPH-Greedy Best Pair Heuristic, GPH-Greedy Percentage Heuristic and GBMH-Greedy Best Monitor Heuristic) using or not using the Local Search-LS. These results are compared to the GA results obtained by (Resende, 2008). We chose 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 labeled by G indicates the group of instances, IP is the value found for this metric when the optimal solution is obtained using the MIP formulation (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 performances of the algorithms were similar. Therefore, we only report 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 heuristics (without local search) heuristics GBPH, GPH and GBMH 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 GPH, but was able to significantly increase the number of best solutions achieved by heuristics GBPH and GBMH.

GBPH+LS, GPH+LS and GBMH+LS did not obtain the best solution in 4, 5 and 10 instances respectively.

Regarding the computational effort, the average time of the heuristics GBPH, GPH and GBMH 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 by 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	GBPH	GBPH+LS	GPH	GPH+LS	GBMH	GBMH+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: Computational results for greedy algorithms

The other experiment was executed using $\alpha = 30$ for the constructive heuristics. Each proposed algorithm was executed ten times with ten different random number seeds.

In Table 2, the best results obtained are presented by group of instances. There are 560 instances, but we show the results only for 176 instances for which we obtained different results among the proposed algorithms and GA. For 384 instances, all algorithms found the same result.

In this table, the column labeled by G indicates the group of instances, $\#Inst.$ is the number of instances in this group for which the algorithms present different results and the other columns show, for each algorithm, the number of instances where the algorithm found the best value among all algorithms.

We can see that for best values all three proposed heuristics outperform the GA Algorithm. The pure heuristics BPH, PH and BMH did not obtain the best solution in 13, 7 and 5 instances respectively, while the GA heuristic did not find the best solution in 18 instances. The use of the

G	#Inst.	AG	BPH	BPH + LS	PH	PH + LS	BMH	BMH + LS
M1-B1	56	50	46	54	50	54	52	56
M1-B2	34	30	34	34	34	34	34	34
M1-B4	28	24	28	28	28	28	28	28
M1-B8	11	9	11	11	11	11	10	11
M2-B2	32	32	29	31	31	31	32	32
M4-B4	11	10	11	11	11	11	11	11
M8-B8	4	3	4	4	4	4	4	4
Sum	176	158	163	173	169	173	171	176

Table 2: Best results for random algorithms

local search algorithm was effective when applied to all constructive heuristics. Both BPH+LS and PH+LS did not obtain the best solution in only 3 instances and BMH+LS obtained all best results.

In Table 3, we present another comparison of the proposed algorithms by showing the average results obtained in ten runs for the same 176 instances.

In this table, the column labeled by G indicates the group of instances, $\#Inst.$ is the number of instances in this group for which the algorithms present different results and the other columns show, for each algorithm, the number of instances where the algorithm found the best average value among all algorithms.

G	#Inst.	BPH	BPH + LS	PH	PH + LS	BMH	BMH + LS
M1-B1	56	46	45	52	53	2	28
M1-B2	34	34	34	34	34	2	29
M1-B4	28	28	28	28	28	2	19
M1-B8	11	11	11	11	11	2	8
M2-B2	32	29	29	31	31	1	25
M4-B4	11	11	11	11	11	1	10
M8-B8	4	4	4	4	4	1	3
Sum	176	163	162	171	172	11	122

Table 3: Average results for random algorithms

We can see that for average values the PH heuristic obtained the best results. The pure heuristics BPH, PH and BMH did not obtain the best average solution in 13, 5 and 165 instances respectively. The use of the local search algorithm showed to be effective only when applied to BMH heuristic. BPH+LS, PH+LS and BMH+LS did not obtain the best average solution in 14, 4 and 54 instances respectively.

We can see that the heuristics based on BPH and PH show a similar behavior regarding best and average values, while heuristics based on BMH present quite different results. The pure BMH heuristic obtained 171 best results and only 11 best average results, and BMH+LS obtained 176 best results and 122 best average results. This discrepancy can be explained by the Standard Deviation (SD) values presented by the heuristics. BPH and PH presented null values for SD for all executions, BPH+LS presented values different from null in 11 instances and PH+LS in only 2 instances, while BMH presented values different from null in 166 instances and BMH+LS in 57 instances.

In order to verify whether or not the differences of mean values obtained by the proposed heuristics are statistically significant, we employed the unpaired Students t-test technique.

Table 4 presents, for each pair of heuristics and for each group of instances, the number of solutions that presents a p-value less than 0.01, which means that the probability of the difference of performance being due to random chance alone is less than 0.01. Between parenthesis, there is a pair showing, for these solutions that are statistically significant, the number of better solutions found by the heuristic on the left side of the pair and by the heuristic on the right side.

G	#Inst.	BPH+LS / PH+LS	BPH+LS / BMH+LS	PH+LS / BMH+LS
M1-B1	56	5(2/3)	2(2/0)	4(4/0)
M1-B2	34	0	0	0
M1-B4	28	0	0	0
M1-B8	11	0	0	0
M2-B2	32	2(0/2)	1(0/1)	2(1/1)
M4-B4	11	0	0	0
M8-B8	4	0	0	0
Total	176	7(2/5)	3(2/1)	6(5/1)

Table 4: t-test results

We can see that we do not have many statistically significant results, but when there are some differences, the PH+LS heuristic shows better results.

The average execution time of the constructive heuristics BPH, PH and BMH for the instances at the Table 2 are respectively 74.48, 78.23 and 54.30 seconds. The use of local search increases the running time by only 30 seconds, as the heuristics generated good initial solutions.

The presented execution times are different for the greedy and the random algorithms because we calculated the average execution time using the results obtained only for the instances in which the algorithms presented different results. As the greedy algorithms presented different results in instances that demand more computational time, their average execution time is higher than the time obtained by the random algorithms.

6 Conclusions

In this work, three constructive heuristics and a local search algorithm for the Minimum Monitoring Set Problem (MMSP) were presented. To analyze the performance of the proposed algorithms, one set of instances was used composed of 560 unit cost instances proposed by (Breslau et al., 2007).

Computational results showed that the three proposed constructive greedy algorithms were able to improve the results obtained by the GA heuristic (Breslau et al., 2007) and the proposed local search was effective to improve the results obtained by two of the three greedy constructive heuristics. The constructive heuristic GPH and the GBPH+LS obtained the best results.

The experiments conducted using the random constructive algorithms showed that these algorithms were also able to find better results than GA and that the local search was able to improve the results for all three random constructive heuristics.

Results obtained for average values showed that the heuristics based on Best Pair Heuristic (BPH) and on Percentage Heuristic (PH) presented more robust results than the ones based on Best Monitor Heuristic (BMH).

Statistics tests also showed that PH+LS heuristic present more statistically significant results than the other heuristics.

These good results obtained by the proposed heuristics encourage us, as future work, to develop metaheuristics approaches, like GRASP or ILS, for MMSP using the presented heuristics.

References

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

Breslau, L., Diakonikolas, I., Duffield, N. G., Gu, Y., Hajiaghayi, M., Johnson, D. S., Karloff, H., Resende, M. G. C., Sen, S. and Towsley, D.: 2007, Optimal node placement for path disjoint 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.: 1979, *Computers and Intractability; A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., New York, NY, USA.

Gu, Y., Breslau, L., Duffield, N. G. and Sen, S.: 2007, GRE encapsulated multicast probing: a scalable technique for measuring one-way loss, *SIGMETRICS ’07: Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, ACM, New York, NY, USA, pp. 355–356.

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.

Resende, M. G. C.: 2008, Personal Communication.