Heuristics for Joint Optimization of Monitor Location and Network Anomaly Detection

Emna Salhi, Samer Lahoud, Bernard Cousin
IRISA / University of Rennes 1, France
{emna.salhi, samer.lahoud, bernard.cousin}@irisa.fr

***Abstract*—To reduce monitoring cost, the number of monitors to be deployed have to be minimized and the overhead of monitoring flows on the underlying network have to be reduced. In a recent work, we demonstrated, using ILP formulations, that there is a trade-off between theses two minimization objectives. However, we have shown that the trade-off could be efficiently balanced by jointly optimizing monitor location and anomaly detection costs.**

The problem is NP-complete, hence ILPs could not deliver solutions for large networks. In this paper, we address the scalability issues. We propose two greedy algorithms that optimize monitor location cost and anomaly detection cost jointly. The first algorithm is based on an exhaustive heuristic that explores all paths that are candidate to be monitored, in order to select a subset of paths that reduces the total monitoring cost. On the opposite, the second algorithm is based on a selective heuristic that avoids exploring all the candidate paths to further improve scalability. The main challenge of this heuristic is to not degrade the solution quality. The two algorithms have been evaluated through extensive simulations on networks of hundred of billions of paths. The comparison of the solutions delivered by the two algorithms to each other and to the solutions delivered by the ILP demonstrates that the selective algorithm provides near-optimal solutions, while achieving a desirable scalability with respect to the network size and significant reduction of the computation time.

I. Introduction

One of the highly desired features in network monitoring is to minimize the cost of deploying and maintaining monitoring devices, and reduce communications between the monitoring devices and the *Network Operations Center* (NOC); while preventing monitoring flows from interfering with real traffic flows. These features could be achieved by minimizing the number of monitors that are to be deployed (*e.g.* [6], [7]), minimizing the number of monitored paths (*e.g.* [4], [5], and avoiding redundant measurements (*e.g.* [2], [3]); respectively. However, we argue that there is an interplay between these objectives. Indeed, reducing the number of monitoring devices and the number of monitoring flows results in monitoring long paths that are quite likely to overlap, which increases redundant measurements. Our previous work [1] illustrated this conflictual aspect through ILP formulations. We proposed a monitoring cost model that includes a monitor location cost and an anomaly detection cost. Simulation results demonstrated that a joint optimization of the two costs reduces efficiently their trade-off. However, the problem is NP-complete, and hence, solutions can not be found for large networks using ILPs.

Typically, to overcome scalability issues, the set of candidate paths that are to be monitored is restrained to a small subset. Unfortunately, none of the related works investigated the impact of this restriction on the quality of the monitoring solution neither did they specify how to choose the set of candidate paths.

In this paper, we further investigate these issues. Acknowledging the efficiency of the joint optimization model to balance the trade-off between the multiple minimization objectives [1], we keep on considering this technique to devise novel greedy algorithms. The aim is to come up with large-scale heuristics that apply for large networks and achieve good quality solutions.

The first algorithm that we call exhaustive greedy algorithm starts by selecting a pair of monitor locations that maximizes the number of covered links; and then it explores all paths between the selected monitors, in order to choose a set of paths that maximizes the number of links it can cover and minimizes redundant measurements. Additional monitors and paths are selected iteratively in order to reduce the number of overlaps among paths that are to be monitored, thereby reducing the cost of the solution.

The second algorithm that we call selective greedy algorithm is based on a heuristic that minimizes drastically the number of explored paths. The underlying idea is to choose a set of non-overlapping paths that maximizes link coverage, and then select additional paths and potentially additional monitors to cover the remaining links. The paths of this set should be long enough to increase the number of covered links. However, we do not require them to be the longest paths of the network. This relaxation avoids exploring all paths between selected monitors. We perform an in depth-first order exploration of the network to find long enough paths between the considered monitors quickly. This would reduce the computation time and improve the scalability of the algorithm. However, this heuristic should not worsen the quality of the solution. To investigate this issue, we compare this algorithm to the exhaustive algorithm and to the ILP through extensive simulations. Results demonstrate that the selective algorithm provides solutions quite close to the exact solutions. Furthermore, it outperforms the exhaustive algorithm and the ILP in terms of computation time and scalability.

This work has been supported by Alcatel-Lucent Labs France, under the grant n. 09CT310-01

II. PROBLEM FORMULATION

The main notations used in the paper are reported in TABLE I. The network topology is represented as a undirected graph $G = (N, E)$ comprising a set of nodes N connected by links in E. Let P be the set of network paths. Unless mentioned, the set of monitor locations, i.e. candidate locations where to deploy monitors, is the set of network nodes N; and the set of paths that are candidate to be monitored is the set of network paths P. Let Cl_l be the cost of monitoring link l and Cm_n the cost of monitoring node n. The link monitoring cost expresses the cost of injecting monitoring flows throughout the link. It should be proportional to the load of the link, in order to balance fairly the monitoring load over all the network links. The monitor location cost expresses the effective cost of deploying hardware and software monitoring devices on the selected location. This cost should include also the distance of the location to the NOC, in order to reduce the communications between the monitors and the NOC. The cost of monitoring a path equals the sum of its individual link monitoring costs. The total monitoring cost is expressed as follows:

$$\textit{Total Monitoring Cost} = \sum_{l \in E, p \in P} Cl_l \delta_{lp} Z_p + \sum_{n \in N} Cm_n Y_n \quad (1)$$

The above cost is the cost of locating monitoring devices and detecting link-level network anomalies. We note that, in this work, we are not interested in the localization of anomalies. We adopt a two-phase monitoring approach that decouples the localization task from the detection task. Acknowledging the claim that network anomalies are rare events [4], the efficiency of this two-phase approach lies in the minimization of network usage when the network performs normally, *i.e.* during the detection phase.

In this work, we are interested in selecting a set of monitor locations and and a set of paths to be monitored that can cover all the network links. The challenge is to reduce jointly the number of monitors to be deployed and the load of monitoring flows on the underlying network. We come up with two greedy algorithms that address this problem. The aim is to find large-scale heuristics that achieve low-cost solutions and reduce the computation time. The two algorithms will be compared to each other, and to the exact solution of the problem formulated in [1], in order the assess their efficiency.

III. HEURISTICS

In this section, we describe two greedy algorithms inspired from the monitoring cost model introduced in [1]. The aim of the algorithms is to optimize jointly the cost of monitor location and the cost of network anomaly detection. The first algorithm is based on an exhaustive heuristic that explores all the network paths; whereas the second algorithm is based on selective heuristics that address scalability issues by reducing the number of explored paths. The challenge is to improve scalability without negatively impacting the solution quality.

TABLE I
NOTATIONS USED THROUGHOUT THE PAPER

Symbol	Definition
CP	The set of candidate paths
SP	The set of selected paths
BP	The set of backup paths
TP	The set of temporary selected paths
SM	The set of selected monitors
$length(p)$	The length of path p in number of links
$nbOverlaps(p)$	The number of overlaps of p with paths in SP and TP
$nbCL$	The number of links covered by paths in SP
$nbTCL$	The number of links temporary covered by paths in TP
$nbOverlaps$	The number of overlaps among paths in SP and TP
Cm_n	The cost of deploying a monitor on node n
Cl_l	The cost of monitoring link l
Z_p	A binary variable that indicates wether path p is selected to be monitored
Y_n	A binary variable that indicates wether node n is selected as a monitor location
δ_{lp}	A binary parameter that indicates wether link l belongs to path p

A. Exhaustive Greedy Algorithm

Algorithm 1 describes the exhaustive algorithm. The aim of the algorithm is to select a set of monitor locations and a set of paths to be monitored that cover all the network links in a way that minimizes the number of monitors and reduces overlaps among monitored paths. This algorithm is a two-step nested algorithm.

- First step: It selects a monitor location. This is done by adding the end-node of a path that has its other end-node selected as monitor location, and maximizes the term *#_newly_covered_links - #_of_links_already_covered*. We can also take into consideration the distance of monitors to the NOC in order to privilege the nearest monitors, thereby reducing the communication cost.
- Second step: it greedily selects paths to be monitored between the monitor added at the first step and monitors previously selected. At each iteration, a path that maximizes the term *#_newly_covered_links - #_of_links_already_covered* is selected. The second step ends when all the network links are covered or when there are no more paths that can cover links that are not yet covered.

These two nested steps are re-iterated greedily until adding a new monitor does not improve the best solution obtained over the previous iterations. We expect that this algorithm performs well for small and sparse networks, because it explores all monitor locations and all paths between the new monitor and the previously selected monitors in order to select the best items. This leads to select few paths and monitors that achieve a full link coverage, and reduces redundant measurements. However, for the same reasons, the algorithm is not expected to scale for large and highly meshed networks where the number of paths between a pair of nodes increases drastically. An obvious heuristical improvement of this algorithm would be to relax the first step by randomly choosing a monitor location. This would avoid exploring all paths having one of their end-nodes selected as monitor, however, we still have

to explore all paths between the newly added monitor and the previously added monitors in order to choose a path that achieves a good balance between the number of links it covers and the number of redundant measurements it yields.

Algorithm 1 Exhaustive Greedy Algorithm

1: $p \leftarrow$ the longest path of the network
2: Add the end nodes of p to SM
3: $CP \leftarrow$ {all paths between nodes in SM }\{p}
4: Add p to SP
5: $nbCL \leftarrow length(p)$
6: Select a path p in $CP \cup BP$ that maximizes $length(p) - nbOverlaps(p)$
7: **if** (nbOverlaps(p) == 0) **then**
8: Add p to SP
9: $nbCL \mathrel{+}= length(p)$
10: **else**
11: Add p to TP
12: $nbTCL \mathrel{+}= length(p) - nbOverlaps(p)$
13: $nbOverlaps \mathrel{+}= nbOverlaps(p)$
14: **end if**
15: $CP \leftarrow CP \setminus \{p\}$
16: **if** (not all links are covered OR the solution can be improved) **then**
17: $BP \leftarrow TP$
18: Resert TP, $nbTCL$ and $nbOverlaps$
19: $CP \leftarrow$ {all paths having one and only one of their end nodes in SM }
20: Select a path p in CP that maximizes $length(p) - nbOverlaps(p)$
21: update SP, TP, nbOverlaps, nbCL and nbTCL (Steps 7-14)
22: Let n be the end node of p that is not in SM
23: $CP \leftarrow$ {all paths having one of their end nodes n and the other one in SM}\{p}
24: Add n to SM
25: Go to step 6
26: **else**
27: END OF THE ALGORITHM
28: **end if**

B. Selective Greedy Algorithm

As mentioned above, the main challenge we are facing is scalability concerns with respect to the number of candidate paths. Our purpose is to come up with new heuristics that reduce the number of explored paths without worsening the quality of the solution. We propose a two-stage greedy algorithm. The first stage selects two monitor locations and then selects a set of non-overlapping paths that maximize link coverage. The second stage selects additional paths and monitors in order to cover the remaining links. For the first stage, we provide a heuristic that avoids exploring all paths between the selected monitors. For the second stage, we provide a heuristic that minimizes jointly the number of monitors and redundant measurements, thereby reducing the total monitoring cost.

Algorithm 2 Selective Greedy Algorithm

1: **for all** $(n1, n2) \in N^2$ **do**
2: Reset SM, SP, $nbOverlaps$, L
3: Add $n1$ and $n2$ to SM
4: $CP \leftarrow$ {all paths between $n1$ and $n2$}\{paths between $n1$ and $n2$ crossing links in L}
5: **while** (CP ! = $\emptyset$) **do**
6: Select a "good path" p in CP
7: Add p to SP
8: $L \leftarrow L$\{links of p}
9: update CP (step ?)
10: **end while**
11: **while** (L ! = $\emptyset$) **do**
12: let $p = \{i..k..j\}$ be the longest path composed of links in L
13: **if** ($i \notin SM$) **then**
14: compute a path p_1 between i and a node in SM that minimizes $length(p_1) - \#links$ of p_1 in L
15: **if** (($\#links$ of p_1 in L) $* C_l > C_m$) **then**
16: $p_1 \leftarrow NULL$
17: Add i to SM
18: **else**
19: $nbOverlaps \mathrel{+}= \#links$ of p_1 in L
20: $L \leftarrow L$\{links of p_1}
21: **end if**
22: **end if**
23: **if** ($j \notin SM$) **then**
24: compute a path p_2 between j and a node in SM that minimizes $length(p_2) - \#links$ of p_2 in L
25: **if** ($\#links$ of p_2 in $L * C_l > C_m$) **then**
26: $p_2 \leftarrow NULL$
27: Add j to SM
28: **else**
29: $nbOverlaps \mathrel{+}= \#links$ of p_2 in L
30: $L \leftarrow L$\{links of p_2}
31: **end if**
32: **end if**
33: Add $concatenate(p_1, p, p_2)$ to SP
34: **end while**
35: **end for**

Stage 1: it selects a set of disjoint paths between a pair of monitors. The aim is to cover as much links as possibile without generating overlaps among selected paths. This problem can be viewed as a maximum set packing problem [8]. The maximum set packing problem is the problem of selecting the maximum number of pairwise disjoint sets among the input sets. This problem was shown to be NP-complete [8]. In our case, the set of paths between the considered monitors maps to the input sets and the elements are the network links. This formal complexity proof of the problem justifies our proposal of an approximative heuristic. The key idea is to select iteratively a path having its end

nodes selected as monitors, remove links that are covered by the selected path from the network graph, and then re-iterate. The selected paths must cover the maximum number of links. A good heuristic would be to select, at each iteration, the longest path. However, such a heuristic requires exploring all paths between the considered monitors, and subsequently yields serious scalability issues. Alternatively, our heuristic selects at each iteration a "good path". By "good path" we mean a path that is not necessarily the longest path, but it is a long path which computation do not require exploring a large number of network paths. An efficient solution is to explore the network graph in an in depth-first order between the considered monitors. The first stage ends when no more paths can be constructed between the considered monitors.

Stage 2: by the end of the first stage we get two deployed monitors, a set of disjoint paths to be monitored and a set of uncovered links. This stage aims at covering these uncovered links by selecting more paths and potentially deploying more monitors in a way that reduces jointly the number of monitors and redundant measurements. The algorithm proceeds by computing the longest path composed of only uncovered links, say p_{ul}. For the two end nodes of the computed path, it proceeds as follows. If the end-node is not already selected as monitor then it computes a path segment, say p, from this node to one of the selected monitors, that minimizes the term: *length(p) - #_links_of_p_that_are_uncovered*. The idea is to choose a path segment that maximizes the number of uncovered links and minimizes the number of redundant measurements. Now if the cost of redundant measurements generated by p dominates the cost of deploying a monitor at the end-node, then a monitor will be deployed at the end-node and p will be dropped, otherwise p will be concatenated to p_{ul}. Links covered by p_{ul} are removed from the set of uncovered links, and the process is re-iterated until the set of uncovered links is empty.

IV. Performance Evaluation

In this section, we describe first the methodology of our evaluation, then we show and analyze our results.

A. Evaluation Methodology

Extensive simulations on random topologies generated using the topology generator BRITE (AS level, Waxman model) [9] was conducted on a PC equipped with a 2,992.47 MHz Intel(R) Core(TM)2 Duo processor and 3.9 GB of RAM. A summary of the characteristics of the topologies considered in our evaluation is depicted in TABLE II. All results are the mean over 20 simulations. The aim of the evaluation is to investigate the efficiency of our heuristics. Namely, we 1) implemented the two proposed algorithms and compared the solutions they delivered in order to verify their scalability with respect to the network size, and the impact of the selective heuristics on the quality of the solutions; 2) compared the solutions delivered by the algorithms to the exact solutions delivered by the path-based ILP [1], in order to investigate the gap of the greedy solutions to the optimal; and also 3) relaxed the path-based ILP formulation to a linear program, solved the LP, performed a random rounding of the LP results and then took the results as an input for the exhaustive greedy algorithm. The LP results constitutes a good starting point for the greedy algorithm and reduces the number of candidate paths and candidate monitors, that is why we consider the use of the LP results in combination with the greedy algorithm.

TABLE II
SUMMARY OF THE TOPOLOGIES CONSIDERED IN THE EVALUATION

Topology	# of nodes	# of links	Average # of paths
TOP(6, 10)	6	10	162
TOP(8, 18)	8	18	3.176
TOP(10, 31)	10	31	209.235
TOP(12, 41)	12	41	3.679.756
TOP(15, 59)	15	58	362.919.718
TOP(20, 80)	20	80	135.604.169.577
TOP(30, 120)	30	120	295.438.105.637
TOP(50, 250)	50	250	536.337.473.112

In our evaluation, we considered a centralized active monitoring infrastructure, where the NOC have a global view of the network topology. The cost of deploying monitors was set equal to the cost of monitoring links, *i.e.* $Cl_l = Cm_n = 1$ $\forall l, n$.

We refer to the exhaustive greedy algorithm as GA-1.1, to the exhaustive greedy algorithm with random selection of monitors as GA-1.2, to the selective greedy algorithm as GA-2, and to the path-based ILP formulation as ILP. We denote by LP-assisted greedy algorithm the exhaustive greedy algorithm applied on the results of the LP.

B. Simulation Results

TABLE III depicts the CPU computation time versus the network topology for the five approaches. Results show that the ILP formulation does not deliver a solution for topologies with 10 nodes and 31 links, and larger due to memory concerns; whereas GA-1.1 and the LP-assisted greedy algorithm suffer memory concerns for networks with 12 nodes and 41 links, and larger. This is because these approaches handle a large set of candidate paths, and hence do not scale for meshed networks. Note that the computation time of the LP-assisted greedy algorithm increases exponentially. TABLE III shows that the resolution of GA-1.1 takes less than 4 seconds of CPU time for these topologies. This means that the resolution of the LP takes quite a long time for average networks and shows serious scalability concerns for large networks. However, in a future work, we will focus on this issue, and propose an efficient solution to resolve the linear relaxations of the ILP formulations proposed in [1].

Further, we note that the random selection of monitors improves slightly the scalability of the exhaustive algorithm. Indeed, we got a solution for topologies (12, 41) in about 10 seconds. However, as discussed previously, this improvement does not hold for highly meshed networks.

TABLE III
CPU RUNNING TIME (OOM MEANS OUT OF MEMORY)

Topology	ILP	LP-Assisted GA	GA-1.1	GA-1.2	GA-2
TOP(6, 10)	0,03 s	0,0035 s	$<$ 1 tic	$<$ 1 tic	$<$ 1 tic
TOP(8, 18)	98,3 s	3,75 s	0,02 s	$<$ 1 tic	$<$ 1 tic
TOP(10, 31)	OOM	55242,46 s	3,96 s	0,26 s	0,02 s
TOP(12, 41)	OOM	OOM	OOM	9,65 s	0,02 s
TOP(15, 59)	OOM	OOM	OOM	OOM	1,03 s
TOP(20, 80)	OOM	OOM	OOM	OOM	4,48 s
TOP(30, 120)	OOM	OOM	OOM	OOM	33,11 s
TOP(50, 250)	OOM	OOM	OOM	OOM	177, 59 s

As expected, the selective greedy algorithm succeeds to overcome the memory insufficiency problem, and delivers solutions for all the considered topologies in quite a short time, *e.g.* 177 seconds for the largest networks.

Now, we investigate the quality of the solutions. For this purpose, we compute the total monitoring cost as given in (1). Simulation results for the five approaches and the 8 considered topologies are depicted in TABLE IV. This metric illustrates the cost gap between the different approaches and shed light on the impact of the selective heuristics on the quality of the solution. Surprisingly, the selective algorithm performs better than the exhaustive algorithm and the LP-assisted greedy algorithm, although it does not explore all the network paths. This is because, the selective algorithm starts by covering the maximum number of paths using only 2 monitors and without generating redundant measurements, and then, it balances the load between monitors and links while covering the remaining links. Furthermore, for small networks, the gap between the exact solutions of the ILP and the solutions of the selective algorithm is negligible.

TABLE IV
THE TOTAL MONITORING COST (OOM MEANS OUT OF MEMORY)

Topology	ILP	LP-Assisted GA	GA-1.1	GA-1.2	GA-2
TOP(6, 10)	12,7	13,75	13,65	13,9	12,8
TOP(8, 18)	21,8	22,35	22,45	23,9	22,55
TOP(10, 31)	OOM	37,11	36,55	37,05	35,9
TOP(12, 41)	OOM	OOM	OOM	48,4	45,9
TOP(15, 59)	OOM	OOM	OOM	OOM	53,55
TOP(20, 80)	OOM	OOM	OOM	OOM	66,95
TOP(30, 120)	OOM	OOM	OOM	OOM	132,05
TOP(50, 250)	OOM	OOM	OOM	OOM	220,79

Lacking of results for large networks for the four first approaches to be compared to the solutions of the selective algorithm, we plotted in Fig.1 the percentage of network usage for this algorithm in order to validate its efficiency for large networks. The percentage of network usage is computed as follows:

$$\%\ Network\ Usage = \left(\frac{\sum_{l\in E, p\in P}\delta_{lp}Z_p + \sum_{n\in N}Y_n}{|N|+|E|}\right) * 100$$

Fig.1 shows that the percentage of network usage decreases when the network size increases. This could be explained by the fact that when the number of paths increases, we get more candidate paths, which increase the possibility to reduce the number of overlaps among monitored paths using few monitors. We note that the percentage of network usage increases slightly for TOP(30, 120) and then it remains constant for TOP(50, 250), although the number of links and the number of paths have doubled. This demonstrates that the selective algorithm scales well with respect to the network size, and specifically the number of network paths.

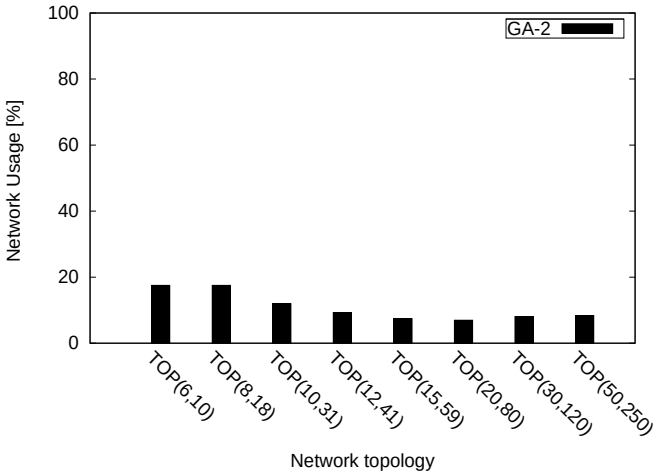

Fig. 1. Resource Usage for GA-2

V. CONCLUSION

In this paper, we proposed novel greedy algorithms for joint optimization of monitor location and network anomaly detection. Our main goal was to come up with large-scale heuristics that reduce the overall monitoring cost. We evaluated our algorithms on networks with hundred of billions of paths. Results show that the selective algorithm outperforms the exact approach, *i.e.* the ILP, and the exhaustive algorithm in terms of scalability and computation time. Furthermore, it provides near-optimal solutions for small networks, and tends to decrease the percentage of network usage when the network size gets larger. Our ongoing work is on extending our monitoring cost model and our heuristics to multi-domain networks.

REFERENCES

[1] E. Salhi, S. Lahoud and B. Cousin, *Joint Optimization of Monitor Location and Network Anomaly Detection*, IEEE LCN, 2010.

[2] P. Barford , N. Duffield, A. Ron and J. Sommers, *Network Performance Anomaly Detection and Localization*, IEEE INFOCOM, 2009.

[3] Y. Zhao, Z. Zhu, Y. Chen, D. Pei and J. Wang, *Towards Efficient Large-Scale VPN Monitoring and Diagnosis under Operational Constraints*, IEEE INFOCOM, 2009.

[4] K.V.M Naidu, D. Panigrahi and R. Rastogi, *Detecting Anomalies Using End-to-End Path Measurements*, IEEE INFOCOM, 2008.

[5] S. Argawal, K.V.M. Naidu, and R. Rastogi, *Diagnosing link-level anomalies using passive probes*, IEEE INFOCOM, 2007.

[6] R. Kumar and J. Kaur, *Efficient Beacon Placement for Network Tomography*, ACM/Usenix IMC, 2004.

[7] Y. Chen and D. Bindel and R.H. Katz, *Tomography-based Overlay Network Monitoring*, ACM/Usenix IMC, 2003.

[8] U. Feige, S. Goldwasser, L. Lovasz, S. Safra, and M. Szegedy, *Interactive proofs and the hardness of approximating cliques*, J. ACM, 43(2):268292, 1996.

[9] BRITE, [Online]. Available: http://www.cs.bu.edu/brite/