Effective Network Monitoring

Yuri Breitbart, Feodor Dragan, Hassan Gobjuka
Department of Computer Science
Kent State University
Kent, OH 44242
{yuri,dragan,hgobjuka}@cs.kent.edu

Abstract— **Various network monitoring and performance evaluation schemes generate considerable amount of traffic, which affects network performance. In this paper we describe a method for minimizing network monitoring overhead based on Shortest Path Tree (SPT) protocol. We describe two different variations of the problem: the A-Problem and the E-Problem, and show that there is a significant difference between them. We prove that finding optimal solutions is NP-hard for both variations, and propose a theoretically best possible heuristic for the A-Problem and three different heuristics for the E-Problem, one of them being also theoretically best possible. We show that one can compute in polynomial time an $O(ln|V|)$-approximate solution for each of these problems. Then, we analyze the performance of our heuristics on large graphs generated using Waxman and Power-Law models as well as on real ISP topology maps. Experiment results show more than 80% improvement when using our heuristics on real topologies over the naive approaches.**

Classification: Algorithms, BFS-tree, Network Monitoring

I. Introduction

Knowledge of the up-to-date network bandwidth utilization as well as network topology is crucial for numerous important network management tasks, including traffic engineering and verifying QoS guarantees for end-user applications. Deploying network measurement and topology tools at key network locations emerged as a main strategy in gathering such information.

However, this approach causes the generation of a significant amount of traffic which shares the same network infrastructure with user applications. From the point of view of these other services this traffic is an overhead since it is of no immediate interest at the user level. Furthermore, placing a monitoring tool at a certain node would only guarantee measurements along the edges of the Shortest Path Tree (SPT) rooted at that node. For the purposes of this paper we assume that each network edge has a weight one and thus, every SPT becomes Breadth-First-Search tree (BFS). Thus, to monitor all active paths in the network, the monitoring tools should be placed at a set of network nodes such that the SPTs rooted at these nodes cover all edges of the network. To reduce the network overhead caused by network monitoring and/or topology traffic, one needs to find minimum number of network nodes such that their BFS-trees cover all network edges. The problem has several interesting variations. One approach is to find minimum set of nodes such that regardless which BFS-trees are selected rooted at these nodes every network edge will be covered. Such an approach makes a good sense, when we do not want any coordination between the selection of BFS-trees at these nodes. Furthermore, in a practical network, the network BFS-tree periodically changes due to the changes in the traffic patterns and network link failures. Consequently, the selection of nodes whose union of arbitrary BFS-trees rooted at these nodes cover every network edge reduces the amount of network management but may increase the network management traffic. Alternatively, we consider minimum number of nodes such that there is a set of BFS-trees selected at these nodes that cover all network edges. In the latter case, network manager should be able to coordinate the selection of BFS-trees at each of the selected nodes, which in turn may cause an additional network traffic. On the other hand, one would expect that the number of selected nodes should be smaller than in the former case.

In this paper we investigate these two approaches and the tradeoff between the amount of network traffic and the minimum number of nodes to place the network management and network topology tools. We prove that both variations of the problem are NP-hard in the number of network nodes. We generate several heuristic algorithms for each of the problems and prove that these heuristics are the best approximation for a selection of minimum number of BFS-trees that cover every edge of the network regardless what variation of the cover problem is being considered. We also conduct extensive simulation study and demonstrate that the number of nodes required for the first variation of network edge cover problem is about 40% more than the number of nodes required for the second variation of this problem. We also run our algorithms on actual ISP providers networks (ATT, Level 3, and Sprint). We demonstrate that using our algorithms, the network manager may significantly reduce the number of BFS-trees comparatively with the current methods used by network managers to select the nodes for placing the network management tools.

The rest of the paper is organized as follows. Section II describes the prior work done in this area. Section III formulates the basic model and the problem statement as well as complexity of the stated problem and their variations. In Section IV, we propose heuristic algorithms and analyze their performance. Section V describes experimental results for networks generated using Waxman and Power-Law models and for real ISP networks. Section VI concludes the paper.

II. Related work

There is a significant body of literature in the area of deploying measurement points and studying their characteristics. However, only few projects focused on minimizing the overhead of such deployment.

IDMaps [11] studies distance monitoring and estimation by finding distance between Tracers, which are monitoring boxes, placed at various network nodes. The distance maps form the virtual topology of the Internet. However, IDMaps does not assume that each tracer monitors a shortest path tree as it is assumed here.

[8] and [17] study link monitoring and delays in IP networks based on a single point-of-control. PingTV [16] and Atlas [18] uses ICMP to generate a logical map of the network from a single probing host. PingTV monitors the traffic condition and network outages of networks with hierarchical structure by pinging hosts in hierarchical order. Atlas captures the topology of IPv6 networks by probing from an initial set of seeds that grows whenever new routers discovered. Either of these methods is proactive. That is, it sends network probes whereas in our methods we do not require any network probes.

In [12] and [13], the authors develop techniques to infer the performance of all of the links (or specified subset of links) that are contained by the trees.

[15] was the first study to show that the concept of "more measurement points is better" is not accurate by showing that the topology can obtained using few measurement points.

[14] studies deploying minimum number of beacons on a network of known topology and BGP-like routing policy so that every link is monitored by messages originating from at least one beacon.

In [1], the authors propose a model to minimize the overhead of monitoring all links of a given network. However, this approach is different from ours as it considers weighted networks and shortest path trees rooted at these nodes are fixed.

III. Model

A network is a graph $G = (V, E)$, where V is the set of nodes and E is the set of direct communication lines between nodes, called *edges*. The number of nodes and edges are respectively denoted by $|V|$ and $|E|$. A shortest path between nodes s and t is denoted by $P_{s,t}$ and the length of this path is denoted by $d(s,t)$. Clearly, between any two nodes there exist possibly more than one shortest path. However, as the name suggest all these paths have the same length $d(s,t)$. For the purposes of our discussion we assume that graph G representing the network is connected. That is, there is a path between any two nodes in V. Consider a node v. For every node $v_i \in V$ we select a shortest path between v and v_i. The union of all these paths is a shortest path tree T_v rooted at v. We call such a tree T_v a *breadth-first search tree (BFS-tree)* if graph G is not weighted. Clearly, for each node v there are many BFS-trees T_v. We denote by S_v the set of all BFS-trees rooted at node v.

Let G be a graph and $v \in V$. We call an edge $e = (a,b) \in E$ *horizontal* with respect to v if $d(v,a) = d(v,b)$ in G. The following observation states that a BFS-tree rooted at a node v cannot cover any edge that is horizontal with respect to v.

Observation 1: Let $G = (V, E)$ be a graph and v be a node of G. Any edge of G which is horizontal with respect to v cannot belong to any BFS-tree T_v rooted at v.

We call an edge $e = (a,b) \in E$ *vertical* with respect to v if $d(v,a) = d(v,b) + 1$ or $d(v,b) = d(v,a) + 1$. Let $e = (a,b)$ be a vertical edge with respect to v and assume that $d(v,a) = d(v,b) + 1$ holds. Then, e is called an *edge unavoidable by* v if any shortest path between v and a includes node b. The following observation holds.

Observation 2: Let $G = (V, E)$ be a graph and v be a node of G. Edge e of G is unavoidable by v if and only if any tree $T_v \in S_v$ contains e.

We are interested in the following two problems.

- E-Problem ("Exist"-Problem): Given a graph $G = (V, E)$, select a minimum set of nodes $R \subseteq V$ and for each node $v \in R$ a tree $T_v \in S_v$ such that the union of selected trees covers all edges of G.
- A-Problem ("Any"-Problem): Given a graph $G = (V, E)$, select a minimum set of nodes $R \subseteq V$ such that, regardless which tree $T_v \in S_v$ is selected for a node $v \in R$, the union of those trees cover all edges of G.

First we demonstrate that optimal solutions for each of these problems are considerably different. Consider, for example a rectilinear grid of size $\sqrt{n} \times \sqrt{n}$ depicted on Figure 1. We number nodes of the grid from 1 to n row-wise (ith row nodes are $(i-1)\sqrt{n}+1, (i-1)\sqrt{n}+2, \ldots, i\sqrt{n}$).

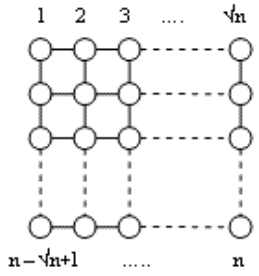

Fig. 1. A-Problem vs E-Problem. On the rectilinear grid of size $\sqrt{n} \times \sqrt{n}$, optimal solution to the E-Problem consists of two trees while optimal solution to the A-Problem consists of $\sqrt{n}$ trees.

One can select two BFS-trees, one rooted at node 1 and another rooted at node n, such that their union covers all the edges of G. Indeed, one can consider a tree rooted at node 1 which is formed by edges $((i-1)\sqrt{n}+1, i\sqrt{n}+1)$, $(j, j+1)$ and $(i\sqrt{n}+j, i\sqrt{n}+j+1)$ (so called "first column and all rows"-tree) and a tree rooted at node n which is formed by edges $(n-\sqrt{n}+j, n-\sqrt{n}+j+1)$, $(i\sqrt{n}, (i+1)\sqrt{n})$ and $((i-1)\sqrt{n}+j, i\sqrt{n}+j)$ (so called "last row and all columns"-tree), where $1 \le i \le \sqrt{n}-1$, $1 \le j \le \sqrt{n}-1$. It is easy to see that both trees are BFS-trees. Thus, a solution to the E-problem consists of only two BFS-trees. In view of Observation 2, it is rather simple to show also that an optimal solution to the A-problem consists of $\sqrt{n}$ roots (BFS-trees). It is easy to see that selecting the nodes on a diagonal of the grid generates a set of BFS-trees that completely cover all edges of the grid.

Using the technique developed in [1], one can easily show that the E-Problem is NP-hard even for unweighted graphs.

It appears that the A-Problem is NP-hard too. It is proven by reducing the set cover problem to it. As a byproduct we get also the NP-hardness of the problem considered in [1] even on unweighted graphs.

As we mentioned above, the NP-hardness of the E-problem on unweighted graphs immediately follows from the construction given in [1]. This result can be directly derived also from the NP-hardness of the A-problem. Thus, the following theorem holds.

Theorem III.1: Both the A-Problem and the E-Problem are NP-hard even for unweighted graphs.

IV. Heuristic algorithms

First we provide a heuristic for the A-problem and point out that our heuristic is the best possible polynomial time approximation algorithm for the problem.

For each node v of graph $G = (V, E)$ we construct a set U_v of unavoidable by v edges in G. It is easy to see that for a given v, the set U_v can be obtained in time $O(|E|)$. Consider now an instance of the set cover problem $(E, \{U_v : v \in V\})$, where E is the universe of elements and $\{U_v : v \in V\}$ is the collection of subsets. We have the following lemma.

Lemma IV.1: A set $R \subseteq V$ is an optimal solution to the A-Problem on a graph $G = (V, E)$ if and only if $\{U_v : v \in R\}$ is an optimal solution to the corresponding set cover problem.

The well-known greedy heuristic for the set cover problem translates into a greedy heuristic, depicted in Figure 2, for the A-Problem. According to [2], the greedy algorithm is a $(ln(\Delta) + 1)$-approximation algorithm for the set cover problem, where Δ is the size of the biggest subset. Since in our case, for any $v \in V$, U_v cannot contain more edges than a BFS-tree rooted at v has, we have the following result.

Theorem IV.1: The Greedy algorithm computes a $(ln(|V|)+1)$-approximation for the A-Problem.

Note that the worst-case time complexity of the Greedy algorithm can be shown to be $O(|V||E|)$.

Input: A graph $G = (V, E)$
Output: A set $R \subseteq V$ of roots

set $R := \emptyset$
for each $v \in V$ compute set U_v of edges unavoidable by v
while $E \neq \emptyset$ do
  choose a node $v \in V \setminus R$ such that $|U_v \bigcap E|$ is maximum (break ties randomly)
  set $R := R \cup \{v\}$, $E := E \setminus U_v$
return R

Fig. 2. A formal description of the Greedy algorithm for the A-Problem.

The reduction from the set cover problem to the A-problem, can be extended to derive a lower bound for the best approximation ratio achievable by any polynomial time algorithm. Using the idea from [1], one can prove the following result.

Theorem IV.2: The lower bound of any polynomial time approximation algorithm for the A-Problem as well as for the E-Problem is $ln(|V|)$.

In the rest of section we provide several heuristics to find a solution to the E-problem. A natural greedy heuristic for the E-Problem would be a procedure where at each step a BFS-tree (and hence a root) is chosen which covers the maximum number of not yet covered edges of G. We call this tree a *current_best BFS-tree*. To find a current_best BFS-tree, one should not iterate over all possible BFS-trees (the number of which could be exponential). Instead, one can do the following. Iterate over all not considered yet nodes of G, say nodes of $S \subseteq V$, building for each node v of S a best possible BFS-tree rooted at v, i.e., a BFS-tree T_v which contains the maximum number of uncovered yet edges of G (a so called *current_best BFS-tree rooted at* v). And then, among those trees $\{T_x : x \in S\}$, choose a tree T_v which covers the maximum number of uncovered edges. To find a current_best BFS-tree rooted at a node v one can use a function given in Figure 3. Clearly, this function works in linear time.

Input: A graph $G = (V, E)$, a node v of G, and a subset $E' \subset E$ of uncovered yet edges
Output: A current_best BFS-tree T_v rooted at v

set $U := \emptyset$ and $q := \max\{d(u, v) : u \in V\}$
compute the layers $L_i(v) := \{u \in V : d(u, v) = i\}$, $i = 1, \ldots, q$, of G with respect to v
for each $u \in V \setminus \{v\}$ do
  let u belong to the layer $L_i(v)$
  if there exists an edge (u, x) in E' such that $x \in L_{i-1}(v)$
  then add such an edge (u, x) to U
  else add to U an arbitrary edge (u, x) with $x \in L_{i-1}(v)$
return tree $T_v := (V, U)$

Fig. 3. A function current_best_BFS-tree(G, v, E') which, given a graph $G = (V, E)$, a node v and a set of uncovered yet edges $E' \subset E$, returns a BFS-tree T_v rooted at v which contains the maximum number of edges from E'.

Now we can give a formal description of the greedy strategy described above for the E-Problem (see Figure 4). We call it Max_New_Edges heuristic. It is easy to see that the runtime of this heuristic is $O(|R||V||E|)$.

Input: A graph $G = (V, E)$
Output: A set $R \subseteq V$ of roots and a family $\mathcal{T} = \{T_v : v \in R\}$ of $|R|$ BFS-trees

set $R := \emptyset$, $\mathcal{T} := \emptyset$ and $E' := E$
while $E' \neq \emptyset$ do
  for each $v \in V \setminus R$ compute
  $T_v :=$current_best_BFS-tree(G, v, E')
  among the trees $\{T_v : v \in V \setminus R\}$ computed, choose a tree T_x which contains the maximum number of edges from E' (break ties randomly)
  set $R := R \cup \{x\}$, $E' := E' \setminus \{$the edge set of $T_x\}$ and $\mathcal{T} := \mathcal{T} \cup \{T_x\}$
return R and $\mathcal{T}$

Fig. 4. A formal description of the Max_New_Edges heuristic for the E-Problem.

A rather standard technique can be used to show that the Max_New_Edges heuristic is an $O(ln|V|)$-approximating algorithm for the E-Problem.

Theorem IV.3: The Max_New_Edges heuristic computes a $O(ln|V|)$-approximation for the E-Problem.

Our next heuristic makes use of the notion of unavoidable edges. In this method, the number of unavoidable edges is calculated with respect to each node in the graph. Then, a node v with the maximum number of uncovered unavoidable edges is selected. If there are more than one such nodes, we break ties by selecting a node arbitrarily. Finally, using a function given in Figure 3, a current_best BFS-tree rooted at node v is calculated, and the edges of this tree are declared covered. The process is repeated until all edges of the graph are covered. We call this heuristic Max_Unavoidables. Its formal description is given in Figure 5. Clearly, it runs also in time $O(|R||V||E|)$.

Input: A graph $G = (V, E)$
Output: A set $R \subseteq V$ of roots and a family $\mathcal{T} = \{T_v : v \in R\}$ of $|R|$ BFS-trees

set $R := \emptyset$ and $\mathcal{T} := \emptyset$
for each $v \in V$ compute the set U_v of edges unavoidable by v;
while $E \neq \emptyset$ do
    choose a node $v \in V \setminus R$ such that $|U_v \bigcap E|$ is maximum (break ties randomly);
    set T_v :=current_best_BFS-tree(G, v, E)
    set $R := R \cup \{v\}$, $E := E \setminus \{$the edge set of $T_v\}$ and $\mathcal{T} := \mathcal{T} \cup \{T_v\}$
return R and $\mathcal{T}$

Fig. 5. A formal description of the Max_Unavoidables heuristic for the E-Problem.

Each of our two next heuristics selects a new root v using different strategy but both of them construct a current_best BFS-tree rooted at node v using function current_best_BFS-tree. Heuristic Max_Degree chooses v to be a node from $V \setminus R$ with the maximum number of uncovered incident edges, while heuristic Random_Root chooses v randomly from $V \setminus R$. In the next section we will experimentally compare the described above four heuristics for the E-Problem against each other and against a very naive heuristic Random_Trees. In this heuristic, at each iteration a root v is selected randomly and a random BFS-tree rooted at v is constructed, and this is repeated until the constructed trees cover all edges of G.

In the remaining part of this section we demonstrate that the difference between the optimal solution and the one returned by Max_Degree heuristic can be significantly different for a given graph G. As an example, we construct a (5n+4)-node graph as follows. We consider two sets of n nodes labeled a_1, a_2, ..., a_n, e_1, e_2, ..., e_n, two sets of n+1 nodes labeled b_1, b_2, ..., b_n, b_{n+1}, d_1, d_2, ..., d_{n+1}, and a set of n+2 nodes labeled c_0, c_1, c_2, ..., c_{n+1}. We connect these nodes as follows. Each node a_i ($i < n$) is connected to nodes c_i and a_{i+1}. Similarly, each node e_i ($i < n$) is connected to nodes c_i and e_{i+1}. Each node b_i ($i \leq n$) is connected to nodes b_{i+1}, c_{i-1}, and c_i. Similarly, each node d_i ($i \leq n$) is connected to nodes d_{i+1}, c_{i-1}, and c_i. An example of such a graph with 5*5+4 nodes is depicted in Figure 6. Heuristic Max_Degree will return $O(|V|)$ BFS-trees rooted at black nodes c_1, c_2, ..., c_n, while it is easy to see that there are three BFS-trees rooted at grey nodes b_1, d_1 and e_1 which cover all edges of the graph.

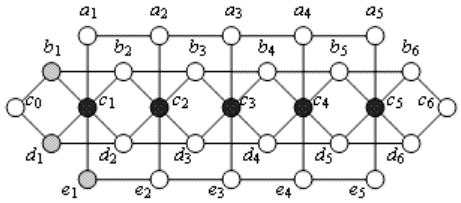

Fig. 6. A graph G for which there are three BFS-trees covering all edges of G, but heuristic Max_Degree will return $O(|V|)$ BFS-trees.

V. Experimental results

In this section, we describe our experiment environment and results we obtained from various algorithms that we have developed. Our experiments are based on simulations as well as realistic scenarios. Simulations are based on network topologies generated using Waxman [4] and Power-Law [3] models. The realistic scenarios are based on ISP topology maps which are obtained from [6]. The results show the performance advantage of our heuristics when comparing them with the Random_Root and Random_Trees heuristics.

A. Waxman model

We generate 300-node network topologies using the Waxman model, which is a popular topology model for networking research. Different network topologies can be generated by varying three parameters: (1) n, the number of nodes in the network graph; (2) α, a parameter that controls the density of short edges in the network; and (3) β , a parameter that controls the average node degree.

For our experimental purposes, we start with the values α=0.15 and β=0.2 (i.e. average degree = 6), we fix the value of α and increase the value of β gradually to simulate dense network with average low degree. Then, we repeat the same process but by fixing the value of β and increasing the value of α to simulate networks with short links and average high degree (i.e. average degree = 59). The reason for increasing values of α and β instead of keeping them fixed is that real ISP topology maps can be completely different from each other as shown in Figure 9. Figure 7 shows the results we got running the heuristic of the A-problem as well as different heuristics for the E-problem on networks generated using Waxman model.

B. Power-Law model

Power-Law model can be used to generate router-level topologies [3]. For our experimental purposes, we generate 300-node flat (i.e. non hierarchical) power-law topologies using BRITE [5], which is the best known power-law-based topology generator. BRITE generates different topologies by changing the values of the following parameters: (1) HS, Size

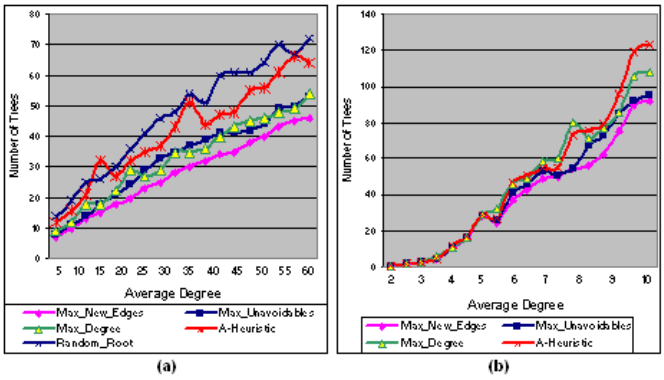

Fig. 7. Number of BFS-trees constructed by different heuristics for a 300-node graph generated by (a) Waxman model and (b) Power-Law model.

AS	Name	Routers	Links
1221	Telstra (Australia)	3,726	9,014
1239	Sprint (USA)	10,332	25,841
1755	Ebone (Europe)	291	1,097
2914	Verio (USA)	6,523	19,289
3257	Tiscali (Europe)	506	1500
3356	Level3 (USA)	1,786	13,838
3967	Exodus (USA)	447	1,842
4755	VSNL (India)	292	669
6461	Abovenet (USA)	654	2,675
7018	AT&T (USA)	11,800	22,309

TABLE I

THE NUMBER OF ROUTERS AND LINKS FOR ALL TEN ISPS.

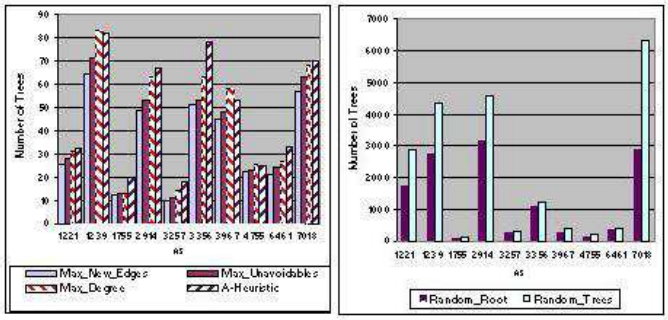

Fig. 8. Number of BFS-trees constructed by all heuristics for the ten selected ISP topologies.

of one side of the plane; (2) LS, Size of one side of a high-level square; (3) NP, Node Placement; (4) m, Number of links added per new node; and (5) IG, Incremental Growth.

We start with average degree m=2 (i.e. tree topology) and increase the degree gradually until m = 10 (i.e. average degree of 10). Figure 7 shows the results we got running the heuristics of the A-problem as well as E-problem on networks generated using Power-Law model.

C. ISP map topologies

We run our heuristics on ten diverse ISP topologies (See Figure 8). We obtain the topology maps from [6] and convert them from Rocketfuel format to adjacency list graph format after removing all inter-AS links. Figure 9 shows three sample backbones overlaid on a map of the United States. The sample backbone maps were obtained from [7]. We see that the size of these networks range from about three hundreds to more than twelve thousands as shown in Table I. Also, we see that topologies could be completely different. For example, we see that the AT&T's backbone network topology includes hubs in major cities and spokes that fan out to smaller per-city satellite POPs. In contrast, Sprint's network has few more POPs than 40 in the USA, all in major cities, and well connected to each other, implying that their smaller city customers are back hauled into these major hubs. Level3 represents yet another paradigm in backbone design. It has a highly connected backbone which is most likely the result of using a circuit technology, such as ATM or frame relay private virtual circuits, to tunnel between POPs.

D. Discussion

Our experiments show changing results as the topology changes. In the results obtained from Power-Law model, all the proposed non-trivial E-Problem heuristics return similar results when the average degree is low (i.e. less than five). However, when increasing the average degree, Max_New_Edges covers all edges with less number of trees than other heuristics because Max_New_Edges takes a global view of the graph before making the next root selection while Max_Unavoidables takes into account only the number of unavoidable edges. Max_Degree and Max_Unavoidables cover all edges with almost the same number of trees whenever the number of unavoidable edges and the degree are similar. The improvement with respect to the naive approaches is very huge (about 80% when the average degree is 6) as shown in Figure 8.

Results obtained from Waxman model show noticeable difference among the E-Problem heuristics as shown in Figure 7. Max_New_Edges heuristic covers all edges with less number of trees than other heuristics. Also, we notice the unexpected situation when Max_Degree heuristic covers all edges with less number of trees than Max_Unavoidables heuristic in some graphs. The reason behind that is the number of unavoidable edges and the maximum degree are equal and there are more than one node to select from in the next iteration. Since heuristics break the tie of selecting the next node randomly in such case, Max_Unavoidables may select a node which is different from what Max_Degree selects. Then, Max_Unavoidables may need more roots to cover all edges as a consequence of that selection. In general, the improvement of our heuristics over the naive heuristics is more than 50%. The improvement on graphs generated using Waxman and Power-Law model differs because Power-Law model tends to generate tree-like topologies which is not the case with Waxman model.

In the case of tree-like topologies, Max_New_Edges, Max_Unavoidables, and Max_Degree tend to select the same node at each iteration. However, selecting roots randomly in such topologies returns very high number of roots as the heuristic may select a leaf node at each iteration and hence, the tree can cover only very few new edges. Also, both Random_Root and Random_Trees tend to return similar results as number of unavoidable edges are large comparing to avoidables. However, in the topologies generated using Waxman model, the difference is big between results obtained from Random_Root and Random_Trees as the options varies at each iteration.

In real topologies, we see that number of edges and connectivity are proportional to the number of roots. Again, Max_New_Edges heuristic outperforms Max_Unavoidables, which in turn outperforms Max_Degree. Also, when number of links decreases, the performance of Random_Root and Random_Trees becomes closer. In networks that have tree-like topologies like AT&T, the improvement with respect to naive approaches goes to almost 100%. In highly connected networks such as Level3, we see more than 90% improvement. In relatively small networks such as VSNL, the improvement is about 80%.

VI. Conclusion

In this paper, we considered the problem of minimizing the overhead of network monitoring. We defined the optimization problem and showed hardness of finding an optimal solution for that problem. We proposed the best possible heuristic for one variation and several heuristics for the other variation, and presented theoretical analysis of these heuristics. We ran extensive experiments using simulations as well as real network topology maps to study the performance in different scenarios. We compared the results that we obtained with two naive approaches and showed the drastic improvement using our heuristics.

References

[1] Y. Bejerano and R. Rastogi, Robust Monitoring of link delays and faults in IP networks, *In Proceedings of IEEE INFOCOM,* (2003).

[2] C. Chvatal, A greedy heuristic for the set-covering problem, *Math. of Operation Research,* 4 (1979), 233–235.

[3] M. Faloutsos, P. Faloutsos, and C. Faloutsos, On Power-Law Relationships of the Internet, *In Proceedings of ACM SIGCOMM,* (1999).

[4] B. M. Waxman, Routing of multipoint connections, *IEEE Journal on Selected Areas in Communications,* 6(9):1671–1622 (1988).

[5] A. Medina, A. Lakhina, I. Matta, J. Byers, http://www.cs.bu.edu/brite/, *Boston University,* (2002).

[6] N. Spring, R. Mahajan, and D. Wetherall, Measuring ISP Network Topologies with Rocketfuel, *In Proceedings of ACM SIGCOMM,* (2002).

[7] R. Sterner, Color landform atlas of the United States, *http://fermi.jhuapl.edu/states/.*

[8] Y. Breitbart, C. Y. Chan, M. Garofalakis, R. Rastogi, and A. Silberschatz, Efficiently Monitoring Bandwidth and Latency in IP Networks, *In Proceedings of IEEE INFOCOM,* (2000).

[9] D. Breitgand, D. Raz, Y. Shavitt, The Travelling Miser Problem, *In Proceedings of IEEE INFOCOM,* (2002).

[10] Cooperative Association for Internet Data Analysis (CAIDA), *http://www.caida.org/.*

[11] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, On the Placement of Internet Instrumentation, *In Proceedings of IEEE INFOCOM,* (2000).

[12] M. Adler, T. Bu, R. Sitaraman, and D. Towsley, Tree Layout for Internal Network Characterizations in Multicast Networks, *In Proceedings of NGC'01,* (2001).

[13] T. Bu, N. Duffield, F. Lo Presti, Network Tomography on General Topologies, *In Proceedings of ACM SIGMETRICS,* (2002).

[14] J. Horton, A. Lopez-Ortiz, On the Number of Distributed Measurement Points for Network Tomography, *In Proceedings of the Conference on Internet Measurement Conference,* (2003).

[15] P. Bar-ford, A. Bestavros, J. Byers, M. Crovella, On the Marginal Utility of Network Topology Measurements, *In Proceedings of the First ACM SIGCOMM Workshop on Internet Measurement Workshop,* (2001).

[16] A. Gubin, W. Yurcik, L. Brumbaugh, PingTV: A Case Study in Visual Network Monitoring, *In Proceedings of the Conference on Visualization,* (2001).

[17] J. Walz, B. Levine, A Hierarchical Multicast Mmonitoring Scheme, *In Proceedings of NGC on Networked Group Communication,* (2000).

[18] D. Waddington, F. Chang, R. Viswanathan, and B. Yao, Topology discovery for public IPv6 networks, *ACM SIGCOMM Computer Communication Review,* (2003).

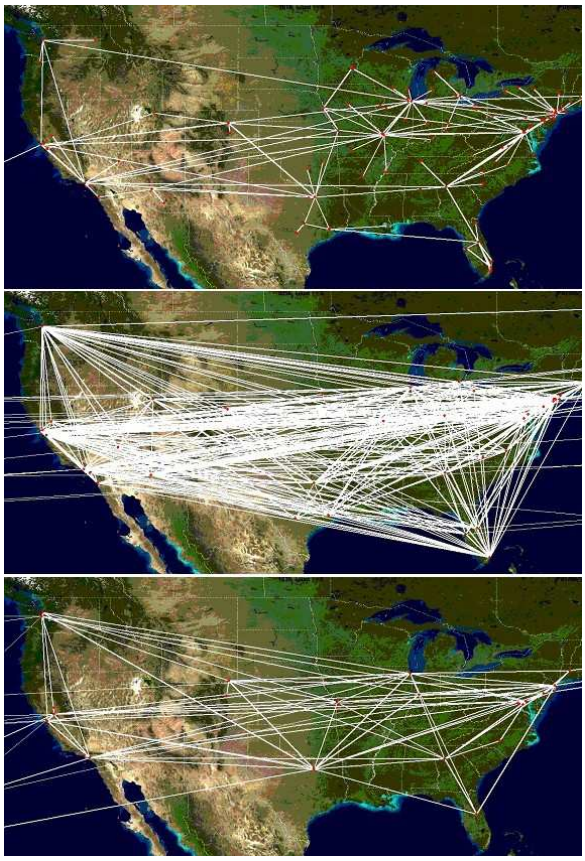

Fig. 9. Backbone (POP level) topologies for AT&T (top), Level3 (middle), and Sprint (bottom). Shaded relief background image Ray Sterner, Johns Hopkins University Applied Physics Laboratory, used with permission.