Effective Monitor Placement in Internet Networks

Yuri Breitbart
Department of Computer Science
Kent State University
Kent, OH 44242
yuri@cs.kent.edu

Feodor F. Dragan
Department of Computer Science
Kent State University
Kent, OH 44242
dragan@cs.kent.edu

Hassan Gobjuka
Verizon Labs
6220 Baltimore Natl Pk
Baltimore, MD 21228
hasan.r.gobjuka@verizon.com

***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 prove that finding optimal solutions for both A- and E-problems is NP-hard. We also show that in general, an A-problem solution requires a significantly higher network overhead than an E-problem solution. We propose optimal approximation algorithms for the A- and E-problems and few different heuristics for the E-problem. Namely, we show that one can compute in polynomial time an $O(ln|V|)$-approximate solution for each of these problems. We analyze the performance of our approximation algorithms and heuristics on large graphs generated using Power-Law model. Performance results show that our heuristic algorithms for both problems achieve from 50% to 90% improvement in the network overhead comparatively with earlier algorithms that appeared in literature.**

***Index Terms*— Algorithms, SP-tree, Network Monitoring**

I. Introduction

Knowledge of the up-to-date network bandwidth utilization is crucial for numerous network management tasks, including traffic engineering and verifying QoS guarantees for end-user applications. Deploying network measurement and network monitoring 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 node v would only guarantee measurements along the edges of the Shortest Path Tree (SP-tree) T_v rooted at v. Clearly, for a given node v there are several $SP-Trees$ that are rooted at v. Thus, to monitor all active edges in the network, one needs to select a subset of network nodes V' and a set of shortest path trees rooted at nodes in V' such that the union of edges in all these trees is equal to all network edges. To reduce the network overhead caused by network monitoring one needs to find a minimum number of network nodes such that their SP-trees cover all network edges. The problem has two interesting variations.

The first variation (termed A-problem) is to find a minimum set of nodes such that regardless of which SP-trees are selected, every network edge will be covered by the trees rooted at these nodes. Such an approach makes a good sense, when we do not want any co-ordination between the selection of SP-trees at these nodes. Furthermore, in a practical network, the network SP-tree periodically changes due to the changes in the traffic patterns and network link failures. Consequently, the selection of nodes whose union of arbitrary SP-trees rooted at these nodes cover every network edge, reduces the amount of network management but may increase the network management traffic.

The second variation (termed E-problem) is to find a minimum number of nodes such that *there is* a set of SP-trees selected at these nodes and these selected trees cover all network edges. In the latter case, network manager should be able to coordinate the selection of SP-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 approximation and heuristic algorithms for each of the problems and prove that these approximations are the best for a selection of a minimum number of SP-trees that cover every edge of the network, regardless what variation of the cover problem is being considered. We also conduct an extensive performance study and demonstrate that our algorithms for both problems achieve from 50% to 90% improvement in the network overhead comparatively with earlier algorithms that appeared in literature [12], [13].

The rest of the paper is organized as follows. Section

An extended abstract of these results was presented at the ICCCN'04 conference [4]

Manuscript received October 21, 2008; revised March 2, 2009; accepted March 18, 2009.

II describes the prior work done in this area. Section III formulates the basic model and the problem statement and Section IV shows the complexity of the stated problem and their variations. In Section VI, we propose the best greedy algorithm for the A-problem and Section VII describes heuristic algorithms for the E-problem and analyzes their performance. Section VIII describes experimental results for networks generated using Power-Law model. Section IX concludes the paper.

II. Related work

In the past several years network monitoring has been extensively studied. Many of these studies concentrated on node monitoring while other few focused on the infrastructure and link monitoring. In this section, we discuss the difference between the approaches proposed in the literature that concentrate on link monitoring. [15] provides an excellent review of network monitoring research. In addition, authors prove that the problem of associating links with tracers in optimal way is NP-hard, and they propose an approximation algorithm for that problem.

IDMaps [11] studies distance monitoring and distance estimation. The authors introduce a notion of a $tracer$, which is a monitoring box. Tracers are placed at various network nodes. The probes from these tracers are used to generate the distance maps. However, IDMaps does not assume that each tracer monitors a shortest path tree as it is assumed here.

PingTV [9] monitors the traffic condition and network outages of networks with hierarchical structure by pinging hosts in hierarchical order. This technique generates additional network traffic to perform network measurements. In [3] and [16] authors study link monitoring and delays in IP networks. The first one proposes a link monitoring scheme based on a single point-of-control in the network. Then, the latency information is gathered using management tools such as SNMP and explicitly-routed IP packets. The latter proposes a hierarchical passive multicast monitoring approach that relies on pre-deployed monitoring daemons to detect and isolate faults in hierarchical networks. However, these approaches depend on source routing that has security-related constrains. Consequently, source-routing is disabled in many of today's networks, rendering approaches proposed in [3] and [16] not viable.

The authors of [14] propose a method to monitor link delays in an Enterprize network using round trip delays. This approach is different from our approach since in [14] it is assumed that the distance (i.e. number of hops) between a tracer and a monitored edge is limited. Decreasing the distance between the tracer and monitored nodes, however, eventually results in increasing the number of tracers. For instance, in a tree-like topology, one tracer is sufficient to monitor all edges if the distance is unlimited while the number of tracers may increase drastically when the diameter of the network is large and the distance is small. Also, since the packets travel at the light speed, there is no noticeable delay in monitoring edges when the distance between the tracer and monitored edges is unlimited.

In [8], the authors propose a method called *validation* to monitor the previously discovered Internet edges. They also prove that the validation problem is NP-hard and present a $\Theta(\log n)$-approximation algorithm for the problem. This model aims at monitoring Internet edges, and each node in the graph represents an AS. Thus, each tree doesn't necessarily span all nodes in the graph. The main difference between this model and our model is that we aim at monitoring a single administrative domain. Consequently, each node in our graph represents a network device (i.e., router) and each tree spans all nodes in the network.

Results presented here are most closely associated with the techniques developed in [10], [12] and [1]. However, there are important differences between their and our methods. The authors in [10], [12] and [13] propose a method for monitoring all edges of a network with the minimum set of beacons (called Locally-Flexible Beacons "LFB") in the presence of dynamic routing. The main idea of the LFB is that if there is a unique path between two nodes in the network, then that path can be monitored by a beacon under all possible routing states. However, these methods lack scalability. Using the approach described in [10], every network element that has degree at least three may be part of the beacon set. Since network elements (e.g. routers and switches) has at least three links, this method lacks practicality since it may result in selecting all elements as monitors. [12] and [13] additionally propose a method for computing the minimum set of beacons, called Beacon Minimization Problem (BMP), that can be used to monitor all network edges. Their method provides a noticeable improvement over the one proposed in [10]. However, in their algorithm, the beacon set selection is based on the vertex-cover which means that each beacon monitors links that are incident to that beacon, besides the links that uniquely connect any two nodes. Due to the nature of the vertex-cover algorithm, this approach will always result in larger number of beacons when the network size increases. From the above discussion we conclude that both approaches lack practicality and may result in a selection of large set of beacons even if the network links can be monitored with few beacons placed in carefully selected key nodes in the network.

The techniques developed in [1] have two major differences with our approach. First, in [1], the authors assume that SP-trees are fixed (i.e. static). That is, when the tree changes due to a failure or changing traffic patterns, the results of [1] cannot be applied. However, in our proposed setting, A-problem addresses all possible trees (rooted at a selected node) such that replacing an existing tree with a new tree does not invalidate our results. Furthermore, in the case of E-problem, the system is able to select the best SP-tree for a set of failures and/or changes in the traffic patterns. Also, our approach does not require a weight assignment. Consequently, the NP-hard result in [1], [10] is a special case of our result presented here.

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 $|E|$. Observe that in an Internet Service Provider (ISP) network consisting of a single Open Shortest Path Protocol ($OSPF$) area, each node v in the network forms a $shortest\ path\ routing\ tree$ to route to all other nodes in the network. A shortest path between nodes s and t is denoted by $P_{s,t}$ and the length (i.e. number of edges) in 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 suggests 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 node v. A shortest path tree T_v rooted at v is a spanning tree of G which, for any node v_i, contains a shortest path between v and v_i. We assume that the graph is unweighted. That is, all links (i.e. edges) in the network have the same weight. This can occur, for instance, when OC-48 connections are used as the ISP backbone, all the links have the same capacity, or all the links have delay below an appropriate threshold.

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 SP-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 SP-tree T_v rooted at v.
Proof: Suppose that $e = (a, b)$ is a horizontal edge with respect to v but nevertheless there is a SP-tree T_v containing e. Since the shortest distances from v to a and b are the same, we obtain that there is a loop in T_v, which is a contradiction. □

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 *unavoidable edge by* v if any shortest path between v and a includes Observe that if edge $e = (a, b)$ is unavoidable with respect to v, then $e = (a, b)$ is also vertical edge with respect to v. However, the opposite is not necessarily correct. To illustrate, in the network depicted on Fig. 1, edge (2, 3) is unavoidable and edge (2, $\sqrt{n}+2$) is vertical (but not unavoidable) with respect to node 1. The following observation holds.

Observation 2: Let $G = (V, E)$ be a graph and v be a node of G. Let S_v be the set of all possible shortest path trees rooted at node v. Edge e of G is unavoidable by v if and only if any tree $T_v \in S_v$ contains e.
Proof: Clearly, if any tree $T_v \in S_v$ contains $e = (a, b)$, then e is vertical with respect to v and any shortest path from a to v includes b if $d(v, a) = d(v, b) + 1$. Suppose now, that there is a tree T_v in S_v that does not contain $e = (a, b)$ and e is an unavoidable edge by v. Assume without loss of generality that $d(v, a) = d(v, b) + 1$. Since T_v is a SP-tree, every node of G must be a node of the tree. Consequently, both a and b are in T_v but the edge (a, b) is not in T_v. On the other hand, since (a, b) is unavoidable by v, any shortest path from a to v contains node b. A contradiction obtained completes the proof. □

We are interested in the following three problems and describe their applications in practice.

- 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.
- BR-Problem ("Bejerano/Rastogi"-Problem): Given a graph $G = (V, E)$ and a SP-tree T_v for each $v \in V$, select a minimum set of nodes $R \subseteq V$ such that the union of T_v, $v \in R$, covers all edges of G.

First we demonstrate that optimal solutions for each of the first two problems are considerably different. (Clearly, the size of the optimal solution to the third problem depends on the choice of the family $\{T_v : v \in V\}$.) Consider, for example a rectilinear grid of size $\sqrt{n} \times \sqrt{n}$ depicted on Fig. 1. We number nodes of the grid from 1 to n row-wise (ith row nodes are $(i-1)\sqrt{n}+1, (i-1)\sqrt{n}+2, \ldots, i\sqrt{n}$).

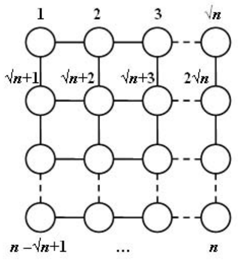


Figure 1. A-Problem vs E-Problem. On the rectilinear grid of size $\sqrt{n} \times \sqrt{n}$, optimal solution to the E-Problem consists of two trees while optimal solution to the A-Problem consists of $\sqrt{n}$ trees.

One can select two SP-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 SP-trees. Thus, a solution to the E-problem consists of only two SP-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 (SP-trees). Indeed, since edges of column j ($j = 1, \dots, \sqrt{n}$) are unavoidable only by nodes of this column, at least one SP-tree rooted at a node of column j must be present in an optimal solution. Any SP-tree rooted at this node will cover all edges of column j, and for any node x not from column j and any edge e from this column, there exists a SP-tree rooted at x which does not contain the edge e. Analogously, at least one SP-tree rooted at a node of row i ($i = 1, \dots, \sqrt{n}$) must be present in an optimal solution. Thus, an optimal solution to the A-problem on a rectilinear grid of size $\sqrt{n} \times \sqrt{n}$ must consist of $\sqrt{n}$ SP-trees with roots one per each column and each row. It is easy to see that selecting the nodes on a diagonal of the grid generates a set of SP-trees that completely cover all edges of the grid.

IV. Hardness results

In this subsection, we prove the hardness of A-Problem, E-Problem, and BR-Problem on unweighted graphs. We first prove that the A-Problem is NP-hard on unweighted graphs by reducing the set cover problem to it. As a byproduct we get also the NP-hardness of the BR-problem even on unweighted graphs. For this one needs to refine substantially the construction of [1].

Theorem 1: The A-Problem is NP-hard even for unweighted graphs.

Proof: We show that the A-problem is NP-hard by reducing a set cover problem (SC) to it. Consider an instance of SC problem $I(Z, Q)$, where Z is a set of m elements and Q is a set of n subsets of these elements. We construct the following unweighted graph $G = (V, E)$. For each element $z_i \in Z$ ($1 \le i \le m$), V contains six nodes u_i, w_i, v_i and a_i, b_i, c_i, and E contains seven edges (u_i, w_i), (u_i, v_i), (v_i, w_i), (a_i, b_i), (a_i, c_i), (b_i, c_i) and (c_i, v_i) (i.e., one has two triangles joined with a bridge edge). For each set $q_j \in Q$, V contains a node labelled by s_j. For each $z_i \in q_j$, E contains edge (s_j, u_i). For each $z_i \notin q_j$, E contains edge (s_j, v_i). In addition, G contains two cliques of size 3 (called *anchor cliques*) $clique_1$ and $clique_2$. $Clique_1$ has nodes r_1, s_1, and t_1, and $clique_2$ has nodes r_2, s_2, and t_2. Also, we have an auxiliary node y which is connected to t_1 and to nodes a_i and b_i ($1 \le i \le m$), and m auxiliary nodes d_i ($1 \le i \le m$) connected to t_1 and v_i. Finally, node t_2 is connected to nodes u_i, v_i, and w_i ($1 \le i \le m$). An example of such a graph is depicted in Fig. 2 for the SC problem with Z = $\{z_1, z_2, z_3, z_4\}$ and four subsets q_1 = $\{z_1, z_2, z_3\}$, q_2 = $\{z_2, z_3, z_4\}$, q_3 = $\{z_1, z_3, z_4\}$, and q_4 = $\{z_1, z_2, z_4\}$.

We claim that there is a solution of size k to the given SC problem if and only if there is a solution of size $k + m + 2$ to the A-problem. Let a solution of the SC problem consist of the subsets $q_{j_1}, \dots, q_{j_k}$. We show that any set S of $k + m + 2$ SP-trees rooted at r_1, r_2, a_i ($1 \le i \le m$), and s_{j_1}, $\dots$, s_{j_k} covers all edges of G. Indeed, any SP-tree T_{r_1} rooted at r_1 evidently covers edges (r_1, s_1) and (r_1, t_1), (t_1, y), (t_1, d_i), (d_i, v_i), (y, a_i), and (y, b_i) ($i = 1, \dots, m$). Since edges (t_2, r_2)

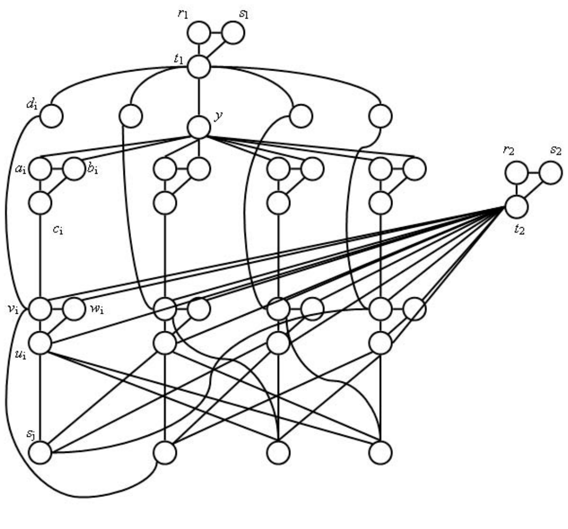


Figure 2. The graph 1 G for an instance of the SC problem with Z = $\{z_1, z_2, z_3, z_4\}$ and four subsets q_1 = $\{z_1, z_2, z_3\}$, q_2 = $\{z_2, z_3, z_4\}$, q_3 = $\{z_1, z_3, z_4\}$, and q_4 = $\{z_1, z_2, z_4\}$.

and (t_2, s_2) are unavoidable by r_1 they are also covered by T_{r_1}. Any SP-tree T_{r_2} rooted at r_2 covers edges (r_2, s_2), (t_2, u_i), (t_2, v_i), (t_2, w_i), (v_i, c_i), (v_i, d_i), (c_i, b_i), and (c_i, a_i) ($i = 1, \dots, m$). Any SP-tree T_{a_i} rooted at a_i ($1 \le i \le m$) covers edges (a_i, b_i), (v_i, u_i), (v_i, w_i) and either (v_i, s_j) ($j \in \{1, \dots, n\}$) if this edge exists or (u_i, s_j) otherwise. Note that both v_i and u_i cannot be connected to the same s_j. Finally, for every $t = i_1, \dots, i_k$, any SP-tree T_{s_t} rooted at s_t covers edges (u_i, w_i) for each $i \in \{1, \dots, m\}$ such that $z_i \in q_t$. Since subsets $q_{j_1}, \dots, q_{j_k}$ cover all elements of Z, each edge (u_i, w_i) ($1 \le i \le m$) will be covered. Thus, all graph edges will be covered by at least one of these $k + m + 2$ SP-trees.

Next, we show that if there is a set of $k + m$ + 2 SP-trees whose union covers all edges of G, then there is a solution for the SC problem of size k. Note that, among those $k + m + 2$ SP-trees, there needs to be a SP-tree rooted at node a_i or b_i to cover the edge (a_i, b_i), and this holds for any $i \in \{1, \dots, m\}$. Also, there must exist, among those trees, a SP-tree rooted at r_1 or s_1 to cover the edge (r_1, s_1) and a SP-tree rooted at r_2 or s_2 to cover the edge (r_2, s_2). Without loss of generality, suppose that the selected roots are r_1, r_2, and $a_1, \dots, a_m$. We show that none of these SP-trees covers edges (u_i, w_i), $1 \le i \le m$. SP-trees rooted at r_1 and r_2 cannot cover edges (u_i, w_i) as these edges are horizontal with respect to both r_1 and r_2 (as well as to s_1, t_1, s_2, t_2). Similarly, for any $i = 1, \dots, m$, edge (u_i, w_i) is horizontal with respect to a_i (as well as to b_i and c_i). Also, any SP-tree T_{a_i} rooted at a_i cannot cover edges (u_ℓ, w_ℓ) for any $\ell \neq i$ because $d(a_i, u_\ell) = d(a_i, w_\ell) = 4$, which means (u_ℓ, w_ℓ) is horizontal with respect to a_i (distances $d(a_i, u_\ell)$ and $d(a_i, w_\ell)$ are realized via node t_2). It is easy to see also that edges (u_ℓ, w_ℓ) ($\ell = 1, \dots, m$) are horizontal with respect to nodes b_i, c_i, d_i, v_i and y, too.

From the previous discussion we see that the remaining k SP-trees must be rooted at nodes u_i, w_i, and s_j for some

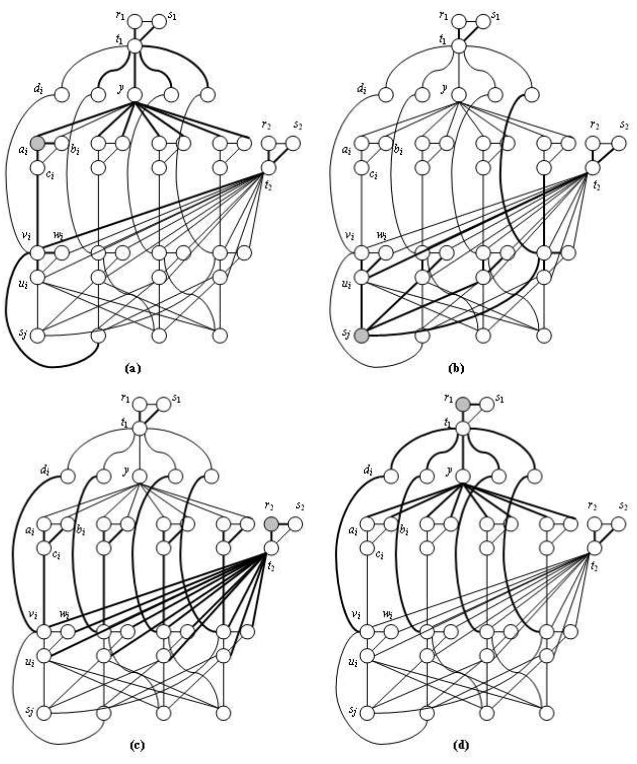


Figure 3. The unavoidable edges of SP-trees T_{a_i}, T_{s_j}, T_{r_2} and T_{r_1}.

subset q_j containing element z_i. Also, any SP-tree rooted at u_i or w_i covers only edge (u_i, w_i) for element z_i and cannot cover any other edge (u_ℓ, w_ℓ) for any $\ell \neq i$ as they are horizontal to both u_i and w_i.

Let $S \subseteq Q$ be a collection of subsets defined as follows. For every SP-tree (out of k remaining SP-trees) rooted at node s_j, we add the set $q_j \in Q$ to S, and for every SP-tree rooted at either node u_i or w_i, we add to S an arbitrary set $q_j \in Q$ such that $z_i \in q_j$. Since the set of k remaining SP-trees covers all element edges (u_i, w_i), $i = 1, \ldots, m$, the collection S covers all the elements of Z, and is a solution to the SC problem of size at most k. Thus, the theorem is proven. □

Using the technique developed in [1] to prove that the BR-problem is NP-hard on weighted graphs, one can easily show that the E-Problem is NP-hard even for unweighted graphs. The freedom to choose an arbitrary tree T_v from S_v for a node v, makes the weight $1 + \epsilon$, used in the reduction of [1], obsolete.

The NP-hardness of the E-problem on unweighted graphs immediately follows from the construction given in [1]. It is easy to see also that this result can be directly derived also from the proof of Theorem 1 as shown below.

Theorem 2: The E-Problem is NP-hard even for unweighted graphs.

Proof: The argument of this proof is very similar to the one used to proof Theorem 1. To prove the hardness of E-Problem on unweighted graphs, we use the same construction $G = (V, E)$ as in the proof of Theorem 1.

We show that if there is a set of $k + m$ + 2 SP-trees such that each edge $e \in G$ is unavoidable with respect to at least one of these $k + m$ + 2 SP-trees, then there is a solution for the SC problem of size k. Note that, among those $k + m + 2$ SP-trees, there needs to be a SP-tree rooted at node a_i or b_i to cover the edge (a_i, b_i), and this holds for any $i \in \{1, \ldots, m\}$. Also, there must exist, among those trees, a SP-tree rooted at r_1 or s_1 to cover the edge (r_1, s_1) and a SP-tree rooted at r_2 or s_2 to cover the edge (r_2, s_2). Without loss of generality, suppose that the selected roots are r_1, r_2, and $a_1, \ldots, a_m$. The set of unavoidable edges for each of these nodes is depicted in Fig. 3. Observe that none of these SP-trees covers edges (u_i, w_i), $1 \leq i \leq m$. Thus, the remaining k SP-trees must be rooted at nodes u_i, w_i, and s_j for some subset q_j containing element z_i. Also, any SP-tree rooted at u_i or w_i covers only edge (u_i, w_i) for element z_i and cannot cover any other edge (u_ℓ, w_ℓ) for any $\ell \neq i$ as they are horizontal to both u_i and w_i.

Let $S \subseteq Q$ be a collection of subsets defined as follows. For every SP-tree (out of k remaining SP-trees) rooted at node s_j, we add the set $q_j \in Q$ to S, and for every SP-tree rooted at either node u_i or w_i, we add to S an arbitrary set $q_j \in Q$ such that $z_i \in q_j$. Since the set of k remaining SP-trees *must* cover all element edges (u_i, w_i), $i = 1, \ldots, m$, the collection S covers all the elements of Z, and is a solution to the SC problem of size at most k. Thus, the theorem is proven. □

Moreover, the same proof proves the NP-hardness of the BR-problem even on unweighted graphs as shown below.

Theorem 3: The BR-Problem is NP-hard even on unweighted graphs.

Proof: Recall that in the BR-Problem, a SP-tree T_v is given for each node $v \in V$. However, each given SP-tree T_v must include the edges that are unavoidable with respect to node v. Since the unavoidable edges of SP-trees rooted at nodes r_1, r_2, a_i, and s_j, $(i = 1, \ldots, m)$, $(j = 1, \ldots, n)$ cover all edges of G, using the same argument as in the proof of Theorem 2, we find that there is a set of $k + m$ + 2 SP-trees such that each edge $e \in G$ is unavoidable with respect to at least one of these $k + m$ + 2 SP-trees, if and only if there is a solution for the SC problem of size k. Thus, the theorem is proven. □

V. Best factor approximation for the A-Problem

Now we provide a heuristic for the A-problem and point out that our heuristic is the best factor approximation algorithm for the problem.

For each node v of graph $G = (V, E)$ we construct a set U_v of unavoidable edges by v 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 5.1: 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.

Proof: Note that, by Observation 2, any SP-tree T_v rooted at a node v must contain all edges of U_v. Therefore, $\bigcup\{U_v : v \in R\} = E$ if and only if R is a solution to the A-Problem. □

The well-known greedy algorithm for the set cover problem translates into a greedy algorithm, depicted in Fig. 5, for the A-Problem. According to [5], 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 SP-tree rooted at v has, we have the following result.

Lemma 5.2: 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 $O(|V||E|)$.

The reduction from the set cover problem, presented in the proof of Theorem 1, can be extended to derive a lower bound for the best approximation ratio achievable by any polynomial time algorithm (see for details [1], [2] where a similar result was proven for the BR-problem).

Lemma 5.3: The lower bound of any polynomial time approximation algorithm for the A-Problem as well as for the E-Problem is $ln(|V|)$.

VI. Best factor approximation for the A-Problem

Now we provide a heuristic for the A-problem and point out that our heuristic is the best factor approximation algorithm for the problem.

For each node v of graph $G = (V, E)$ we construct a set U_v of unavoidable edges by v 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 6.1: 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.

Proof: Note that, by Observation 2, any SP-tree T_v rooted at a node v must contain all edges of U_v. Therefore, $\bigcup\{U_v : v \in R\} = E$ if and only if R is a solution to the A-Problem. □

The well-known greedy algorithm for the set cover problem translates into a greedy algorithm, depicted in Fig. 5, for the A-Problem. According to [5], 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 SP-tree rooted at v has, we have the following result.

Lemma 6.2: 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 $O(|V||E|)$.

The reduction from the set cover problem, presented in the proof of Theorem 1, can be extended to derive a lower bound for the best approximation ratio achievable by any polynomial time algorithm (see for details [1], [2] where a similar result was proven for the BR-problem).

Lemma 6.3: The lower bound of any polynomial time approximation algorithm for the A-Problem as well as for the E-Problem is $ln(|V|)$.

VII. Heuristics for the E-Problem

In this section we provide several algorithms to find a solution to the E-problem. A natural greedy algorithm for the E-Problem would be a procedure where at each step a SP-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 SP-tree*. To find a current_best SP-tree, one should not iterate over all possible SP-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 SP-tree rooted at v, i.e., a SP-tree T_v which contains the maximum number of uncovered yet edges of G (a so called *current_best SP-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 SP-tree rooted at a node v one can use a function given in Fig. 6. Clearly, this function works in linear time.

Now we can give a formal description of the greedy strategy described above for the E-Problem (see Fig. 7). We call it Max_New_Edges algorithm. It is easy to see that the runtime of this algorithm is $O(|R||V||E|)$.

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

Theorem 4: The Max_New_Edges algorithm computes a $O(ln|V|)$-approximation for the E-Problem.

Proof: Let c denote the minimum number of SP-trees needed to cover all edges of a graph $G = (V, E)$, and let g denote the number of SP-trees produced by the Max_New_Edges algorithm. We will show that $(g - 1)/c \leq ln|E| \leq 2\ ln|V|$. Let $m := |E|$. Initially, there are $m_0 = m$ edges left to be covered, and we know that there are c SP-trees that cover all edges of G. Therefore, by the pigeon-hole principle, there must be at least one SP-tree that covers at least m_0/c edges. Since the Max_New_Edges algorithm selects a SP-tree that covers maximum number of uncovered yet edges, it will select a SP-tree that covers at least this many edges. The number of edges that remain to be covered is at most $m_1 = m_0 - m_0/c = m_0(1 - 1/c)$. Applying the argument again, we know that we can cover these edges with c SP-trees (the optimal cover), and hence there

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 $R$;
```

Figure 4. A formal description of the Greedy algorithm for the A-Problem (called A-Heuristic).

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 $R$;
```

Figure 5. A formal description of the Greedy algorithm for the A-Problem (called A-Heuristic).

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 SP-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 $T_v := (V, U)$;
```

Figure 6. A function current_best_SP-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 SP-tree T_v rooted at v which contains the maximum number of edges from 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 $|R|$ SP-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_SP-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 $\mathcal{T}$;
```

Figure 7. A formal description of the Max_New_Edges algorithm for the E-Problem.

exists a SP-tree that covers at least m_1/c uncovered edges, leaving at most $m_2 = m_1 - m_1/c = m_0(1 - 1/c)^2$ edges uncovered. If we apply this argument $g - 1$ times, each time we succeed in covering at least a fraction of

$(1 - 1/c)$ of the remaining edges. Then the number of edges, that remain uncovered after $g - 1$ SP-trees have been chosen by the Max_New_Edges heuristic, is at most $m_{g-1} = m_0(1-1/c)^{g-1}$. We are interested in the largest value of g such that $1 \leq m(1 - 1/c)^{g-1}$. We can rewrite this as

$$1 \leq m((1 - 1/c)^c)^{(g-1)/c},$$

and, using the fact that for all $c > 0$, $(1 - 1/c)^c \leq 1/e$ (where e is the base of the natural logarithm), obtain $1 \leq m(1/e)^{(g-1)/c}$. That is, $e^{(g-1)/c} \leq m$ and therefore, $(g-1)/c \leq ln\ m \leq 2\ ln|V|$. □

Our next algorithm 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 Fig. 6, a current_best SP-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 Fig. 8. Clearly, it runs also in time $O(|R||V||E|)$.

We consider also the following two naive but natural heuristics. Each of these heuristics selects a new root v using different strategy but both of them construct a current_best SP-tree rooted at node v using function current_best_SP-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 remaining part of this section we demonstrate that the difference between the optimal solution and the one returned by Max_Degree heuristic (and therefore, by Random_Root 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 labelled a_1, a_2, ..., a_n, e_1, e_2, ..., e_n, two sets of n+1 nodes labelled b_1, b_2, ..., b_n, b_{n+1}, d_1, d_2, ..., d_{n+1}, and a set of n+2 nodes labelled 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 Fig. 9. Heuristic Max_Degree will return $O(|V|)$ SP-trees rooted at black nodes c_1, c_2, ..., c_n, while it is easy to see that there are three SP-trees rooted at grey nodes b_1, d_1 and e_1 which cover all edges of the graph.

As our experimental results show, even these naive Max_Degree and Random_Root heuristics outperform the one proposed in [12], [13] in minimizing network monitoring overhead in Power-Low graphs.

VIII. COMPARISON

In this section, we compare the performance of our algorithms with other methods in the literature that address similar problem. Namely, we compare the method suggested in [12] and [13], which is called the Beacon Minimization Problem (BMP) method, with the method we proposed for the A-Problem. Then, we compare the Max_New_Edges, Max_Unavoidables with the method proposed for the BMP problem. We first present theoretical worst-case comparison between the method proposed for the BMP and our methods and then we discuss experimental results showing average cases. In the following subsections, we refer to the method proposed in [12] and [13] as BMP method.

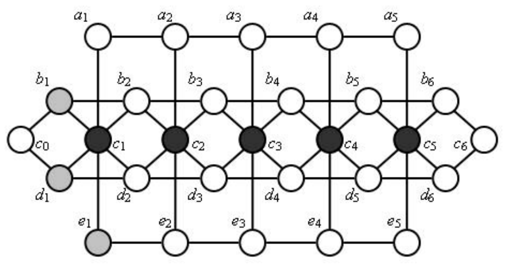


Figure 9. A graph G for which there are three SP-trees covering all edges of G, but heuristic Max_Degree will return $O(|V|)$ SP-trees.

A. Theoretical Comparison

The authors in [12] compared their approach with the one proposed in [10] and showed that their algorithm reduced the number of beacons required by [10] by more than 50%. We show now that comparing to BMP our methods may decrease the number of monitors by factor of $\sqrt{n}$, where n is the number of network nodes. Indeed, consider again the network depicted in Fig. 1. The BMP algorithm finds a uncovered edge in the network and selects a node that is incident to that edge. Then, it adds the set of edges that are incident to the newly selected beacon to the set of covered (i.e. monitored) edges. Note that since there is more than one path between each two nodes, every beacon selected by the BMP algorithm will cover only the links that are incident to the selected beacon. Assume that the BMP algorithm finds a uncovered link (i, i+1) or (i, $i + \sqrt{n}$), i= 1, ..., n-1, and then selects node i as beacon. In this case, n-1 nodes are required by the BMP algorithm to monitor all links. However, as we showed previously, applying the A-Heuristic, only $\sqrt{n}$ beacons are required to monitor all links. Furthermore, all Max_Unavoidables, Max_New_Edges and Max_Degree will monitor all links by only two beacons.

B. Experimental Environment

Now we discuss our experiment environment and results. To compare the performance of the method proposed in [12] with the Max_Unavoidables, we generated 100-, 500- and 1000-node power-law-based networks using $BRITE$ [6]. Each node had degree at least three and

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 $|R|$ SP-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_SP-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 $\mathcal{T}$;
```

Figure 8. A formal description of the Max_Unavoidables heuristic for the E-Problem.

average degree ranging from 5 to 30. The reason for such setup was that all links that are incident to nodes that have degrees one and two can be monitored by exactly one beacon when applying any of the four methods. And thus, it does not affect the results. Also, network elements (e.g. Routers and Switches) are connected to more than two links in practical networks.

We ran each set of experiments for five times, on the same set of inputs using the following procedure. After $BRITE$ generated a network, a root node (i.e beacon) is selected based on the method being tested (e.g. Max_Unavoidables) and the number of links that are monitored by the selected beacon is computed. In the method proposed in [12], a new beacon that is adjacent to at least one uncovered link is selected arbitrarily at each iteration. The process is repeated for all methods until all links are covered by the set of selected beacons. The results were averaged over five runs.

C. Experimental Results

Fig. 10 (a), (b) and (c) shows the relative performance of the BMP, A-Heuristic and the Max_Degree methods for network sizes 100, 500 and 1000, respectively. It can be easily observed that the improvement depends on the network size and average node degree. When the network size was 100 nodes and the average degree increased from 5 to 30, the improvement of A-Heuristic ranged from %76 to %7.3. However, as expected, the number of beacons required by the BMP method increased drastically when the network size increased to 500 and 1000 nodes. The increment in number of beacons was fairly small when the Max_Unavoidables was applied to 500 and 1000-node networks. The performance of Max_Degree method was better than the A-Heuristic by about 50% since it has flexibility in building trees.

Fig.11 (a), (b) and (c) shows the number of beacons required by Max_New_Edges, Max_Unavoidables, Random_Root and the BMP methods for network sizes 100, 500 and 1000, respectively. We can see that the method Max_New_Edges performed better than the Max_Unavoidables and Random_Root methods and the improvement was proportional to the network size and average degree. This is due to the fact that the Max_New_Edges method selects the new beacon based on a global view of the network while the Max_Unavoidables takes into account only the number of unavoidable edges. Note that even the heuristic Random_Root, besides Max_New_Edges and Max_Unavoidables, performed significantly better than the BMP method. This could be predeceased since the BMP method does not build spanning tree as we pointed out previously.

IX. Conclusion

In this paper, we presented two variations of the problem of minimizing the network monitoring overhead when the all network links are covered. We showed that the monitor selection problem to minimize the overhead is NP-hard for both variations. Then, we proposed the best possible polynomial-time approximation algorithm for one variation and several algorithms for the other variation, presented theoretical analysis of these algorithms and showed that one of our algorithms for the second variation is also best possible approximation factor. We compared the results that we obtained with the results described in the literature and found out that our algorithms perform significantly better than previous ones.

References

[1] Y. Bejerano and R. Rastogi, Robust Monitoring of link delays and faults in IP networks, *In Proceedings of IEEE INFOCOM,* (2003), 11 pages.

[2] Y. Bejerano and R. Rastogi, Efficient Monitoring Schemes for IP Networks, Research Report, Bell Labs, 2001.

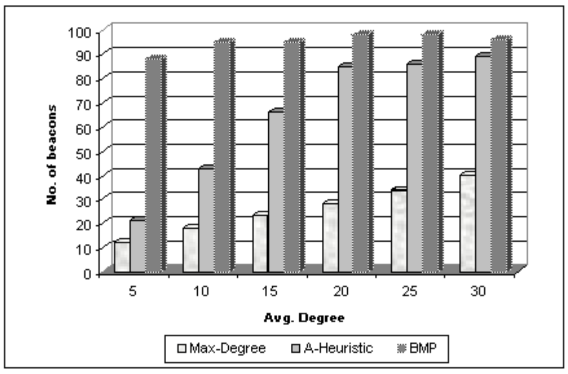
[3] Y. Breitbart, C. Chan, M. Garofalakis, R. Rastogi and A. Silberschatz, Efficiently Monitoring Bandwidth and Latency in IP Networks, *In Proceedings of IEEE INFOCOM,* (2001), 11 pages.

[4] Y. Breitbart, F.F. Dragan, H. Gobjuka, Effective Network Monitoring, *In Proceedings of ICCCN,* (2004), 394–399.

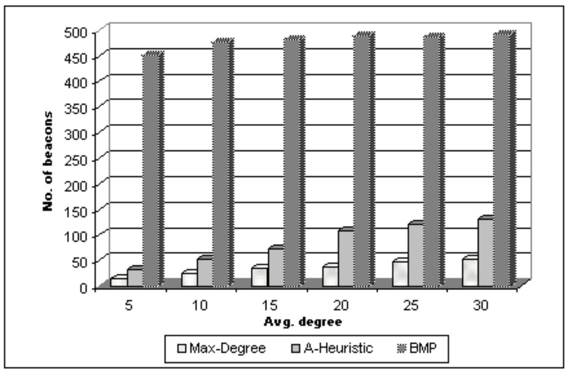
[5] V. Chvatal, A greedy heuristic for the set-covering problem, *Math. of Operation Research,* 4 (1979), 233–235.

[6] M. Faloutsos, P. Faloutsos, and C. Faloutsos, On Power-Law Relationships of the Internet, *In Proceedings of ACM SIGCOMM,* (1999), 251–262.

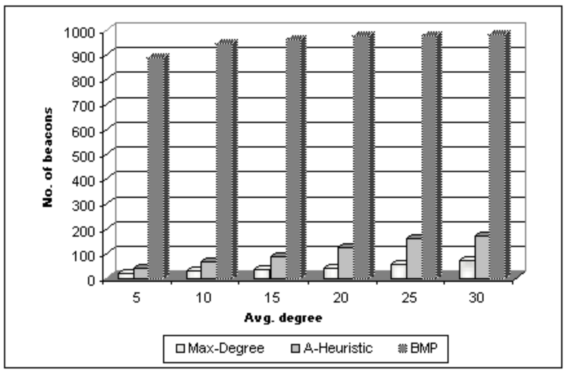
[7] M. Garey, and D. Johnson, Computers and Intractability, A Guide to the Theory of NP-Completeness, *W.H. Freeman and Co, twenty-third printing,* (2002).



(a)

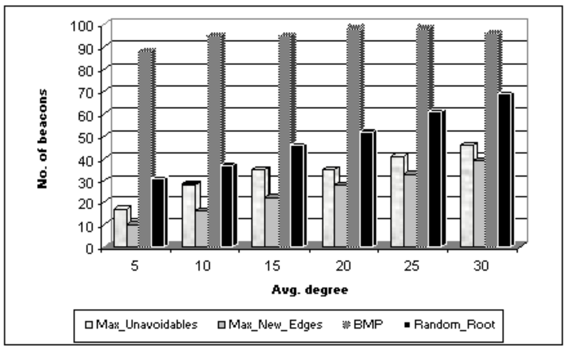


(b)

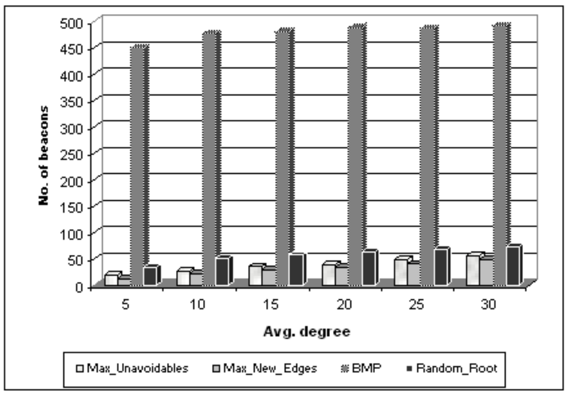


(c)

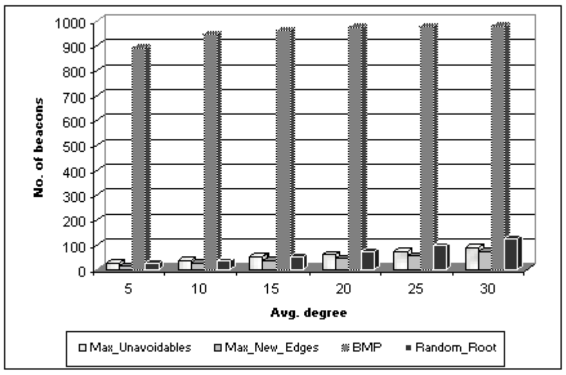
Figure 10. Number of beacons required by the A_Heuristic and BMP methods to monitor all network links when the network size was: (a) 100 nodes, (b) 500 nodes, and (c) 1000 nodes.



(a)



(b)



(c)

Figure 11. Number of beacons required by the Max_Unavoidables, Max_New_Edges, Random_Root and BMP methods to monitor all network links when the network size was: (a) 100 nodes, (b) 500 nodes, and (c) 1000 nodes.

[8] M. GONEN, Y. SHAVITT, A $\Theta(\log n)$-approximation for the set cover problem with set ownership, *Information Processing Letters ,* (2009), 183–186.

[9] A. GUBIN, W. YURCIK, L. BRUMBAUGH, PingTV: A Case Study in Visual Network Monitoring, *In Proceedings of the Conference on Visualization,* (2001), 421–424.

[10] 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), 204–209.

[11] S. JAMIN, C. JIN, Y. JIN, D. RAZ, Y. SHAVITT, L. ZHANG, On the Placement of Internet Instrumentation, *In Proceedings of IEEE INFOCOM,* (2000), 295–304.

[12] R. KUMAR AND J. KAUR, Practical Beacon Placement for Link Monitoring Using Network Tomography, *In Proceedings of Internet Measurement Conference,* (2004).

[13] R. KUMAR AND J. KAUR, Practical Beacon Placement for Link Monitoring Using Network Tomography, *In IEEE Journal on Selected Areas in Communication (J-SAC), special issue on ”Sampling the Internet: Techniques and Applications*, (Dec 2006).

[14] J. MOULIERAC AND M. MOLNAR, Active monitoring of delays with asymmetric routes, *IRISA Institut de recherche en informatique et systemes aleatoires,* (July, 2005), 16 pages.

[15] K. SUH, Y.NG GUOY, J. KUROSE AND D. TOWSLEY, Locating network monitors: complexity, heuristics, and coverage, *UMass Computer Science Techincal Report 2004-61,* (July, 2004).

[16] J. WALZ, B. LEVINE, A Hierarchical Multicast Monitoring Scheme, *In Proceedings of NGC on Networked Group Communication,* (2000), 105–116.