2015 Seventh International Symposium on Parallel Architectures, Algorithms and Programming

Efficient Heuristic for Placing Monitors on Flow Networks

Chang Liu[1,2], Guiyuan Jiang[3], Jigang Wu[4], Meixia Zhu[1], Lijia Tu[1]

[1] School of Computer Science and Software Engineering, Tianjin Polytechnic University, Tianjin 300387, China
[2] State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
[3] School of Computer Engineering, Nanyang Technological University, 639798, Singapore
[4] School of Computer Science and Technology, Guangdong University of Technology, Guangzhou 510006, China
Email: {lc.jasmine94,ljtusummer}@gamil.com, gyjiang@ntu.edu.sg, asjgwucn@outlook.com, ilryx8292@126.com

***Abstract*—Network flow monitoring is a crucial procedure for optimizing the network infrastructure for better application performance or security issue. In this paper, we address the problem of Flow Edge-Monitoring (denoted as FEM), for which we need to find $k(k > 0)$ edges in an undirected graph $G = (V, E)$, so that if we place k flow monitors on these edges to measure the flow along them, we can maximize the total number of edges for which the flow information can be uniquely determined according to the flow conservation property. This problem has been proved to be NP-hard, and previously reported approach suffers from higher running time complexity. In this paper, we develop efficient heuristic algorithm $ALGHEU$ to significantly accelerating the FEM problem without compromising on solution quality, i.e. without obviously increasing the number of required monitors. We also customize and Ant Colony algorithm, which runs slow but produce near optimal solution, to evaluate our heuristic algorithm $ALGHEU$. The experimental results show that, the proposed approach can significantly accelerate the previous approach by 47.67 times, while the solution quality remains nearly as good as the previous algorithm.**

***Index Terms*—Network flow, flow edge monitoring, monitor placement, algorithm**

I. Introduction

Accurate and rapid monitoring of network flow is a critical issue for network management in terms of detecting cornucopia exceptions and enhancing network security. With enhanced visibility into the flow information, we can proactively manage the network to reduce outages, solve problems faster and ensure efficient and cost-effective operations, provide better application performance and enhanced security guarantee. Moreover, the network flow information is of great useful to unearth the potential of network resources, save the cost of network interconnection, and provide evidence for network optimization and network planning adjustment.

As an important way of network traffic measurement, the flow edge-monitoring (FEM) technique surveilles the network flow by placing monitors on certain edges. Specifically, in an undirected graph $G = (V, E)$, each edges possess an unknown flow ψ, the FEM problem is to place $k(k > 0)$ edges in order to maximize the ability of flow monitoring. Note that, besides the k edges on which the monitors are placed on, another subset B of edges can also be monitored by the k monitors according to the flow conservation property. The objectives of flow edge-monitoring is to find the best subset of k edges so as to maximize the total number of edges whose flow information can be determined. Our proposed approach can also be used to find the minimum edge subset such that placing monitors on these edges can monitor the entire network.

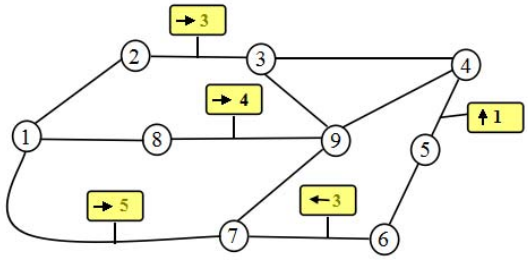


Fig. 1. A undirected graph G with five monitors.

For example, Fig.1 shows a undirected graph covered by monitors. Obviously, in this case, there are k = 5 monitors which are represented by rectangle boxes, where the flow direction and flow quantity are depicted inside the box. Clearly, we can directly obtain the information for flows ψ(2,3), ψ(4,5), ψ(6,7), ψ(7,1) and ψ(8,9). Moreover, according to flow conservation property, we can deduced the information for flows ψ(1,2), ψ(1,8), ψ(5,6) and ψ(7,9). Consequently, we can monitor 9 edges in graph G in total using five monitors.

It is reported that most problems arise in flow monitoring on general graphs are NP-hard [1, 2]. Ritter H et al. [9] propose an effective flow control mechanism for Wireless Ad-Hoc Networks, where the utilized flow conservation property is based on the research data stream technology [10]. A similar problem was studied by Gu and Jia [3] who considered a traffic flow network with directed edges. They observed that $m-n+1$ monitors are necessary to determine the flow on all edges of a strongly connected graph, and that this bound can be achieved by placing flow monitors on edges in the complement of a spanning tree. In this paper, we try to minimize the number of

2168-3042/15 $31.00 © 2015 IEEE
DOI 10.1109/PAAP.2015.12

required monitors instead of focusing on theoretical analysis. Khuller et al. [4] studied an optimization problem where pressure meters may be placed on nodes of a flow network. An edge whose both endpoints have a pressure meter will have the flow determined using the pressure difference, and other edges may have the flow determined via flow conservation property.

The flow monitoring problem is also related to the classical k-cut and multi-way cut problems [5–7], where the goal is to find a minimum-weight set of edges that partitions the graph into k connected components. From this point of view, our monitoring problem can be regarded as maximizing the number of connected components obtained from removing the monitor edges and the resulting bridge edges, where a bridge is a edge whose removal will increase the number of connected components.

To the best of our knowledge, the only work that solve the same problem is reported in [8]. However, the work focuses on theoretical analysis and the proposed algorithm for placing k monitors in a flow network suffers from higher running time complexity. In this paper, we develop efficient algorithm to accelerate the problem of placing monitors on k edges to maximize the ability of flow monitoring.

The rest of this paper is organized as follows. We introduce the related definitions and important notations, and formulate the flow edge-monitoring problem in section II. The proposed algorithm is introduced in section III, and section IV presents the experimental results with analysis. We conclude our work in Section V.

II. Problem Description

Before formally present the Flow Edge-Monitoring problem, we first introduce the formal notations and definitions utilized in this paper. We consider an directed graph $G = (V, E)$, where V is the set of vertices, and E is the set of edges. (u, v) indicates the edge connecting the vertices u and v. Let $D(v)$ be the degree of each vertex v. Moreover, we use $a, b, c, d, ...$ to denote edges. For simplicity, we assume the input graph is connected.

A flow on (v_i, v_j) is a function ψ represent a flow value and a direction from v_i to v_j, denoted by $\psi(v_i,v_j)$. We assume that flow ψ satisfies two following conditions: (i)$\psi(u, v)$ = -$\psi(u, v)$, it is anti-symmetrical.(ii)ψ satisfies the flow conversation property, that is, $\sum_{u,v\in E} \psi(u, v)$ = 0.

Clearly, if we place a monitor on edge (u, v), we can certainly determine the value and direction of the flow going through edge (u, v). Moreover, we can also monitor more edges according to the flow conservation property. In previous works, Flow Edge-Monitor problem was investigated using the following definitions.

Definition 1. *$Br(G)$ is the set of bridges in graph G, and that some flow ψ is given for all edges in some set $M \subseteq E$, and not for other edges. They observed that: for $(u, v) \in E - M$, $\psi(u, v)$ is uniquely determined from the flow preservation property if and only if $(u, v) \in Br(G - M)$.*

Definition 2. *$gain(G, M) = |M \cup Br(G - M)|$ means the total number of edges for which the flow can be determined if place monitors on the edges in M. Thus,the Flow Edge-Monitor Problem can be defined formally as follows:*

Problem*(Flow Edge-Monitoring) Given a graph $G = (V, E)$ and an integer $k > 0$, find a set $M \subseteq E$ with $|M| \leq k$ that maximizes $gain(G, M)$.*

Definition 3. *Given an undirected graph $G = (V, E)$, $D(v) \geq 2$ for arbitrary vertex $v \in$V, only if to execute process as follows can cover total edges, that can determine S as G's weak vertex cover set:*

1) *Cover edges related to S.*
2) *If edges related to v have been covered achieve $D(v) - 1$, then cover the rest edges of v.*
3) *Repeat (2), until no more new edges can be covered.*

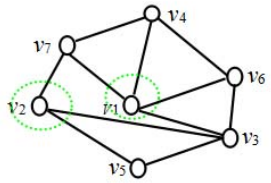


Fig. 2. Weak vertex cover set.

As Fig.2 shows, $\{v_1,v_2\}$ is the minimum weak vertex cover set. Analogously, the goal for flow Edge-Monitor problem is to find a minimum edge set, and the thought of our algorithm is based on it.

III. Proposed Algorithms

In this section, we present our strategies and algorithms in detail. A proposed heuristic algorithm, named ALGHEU, is described in section III-A. In our approach, we consider the Flow Edge-Monitor problem under the *lowest-numbered* policy. In *lowest-numbered* policy, when we require to choose nodes or edges according to the given conditional, and among these nodes or edges satisfy the condition, choose the smallest number.

The notations utilized in this section are defined as follows. Let $D(v)$ be the degree of the vertex v, D_{min}(v) is the vertex with minimum degree among total vertexes. The edges related to v denoted by $E_{(v)}$. Set S refer the consequence of monitor placing, T refer the count of placed-monitors in the end.

A. Proposed Heuristic Algorithm

The algorithm ALGHEU determines whether to place a monitor on edges $E(u, v)$ according to the follow rules.

1) First calculate the degree for each node by performing a traversal of the graph G. Select the minimum degree node v from vertex set V. Let us illustrate the reason to select minimum degree of vertexes. In the case that the minimum degree corresponds to more than one vertices, we pick the vertex conform *lowest-numbered* policy;
   a) In Fig.3, this kind of graph, require one monitor simply, make capable to surveil the network overall. Feature for the graph is each vertex's degree is

2, it also means that, if a vertex's $D(v)-1$ edges be monitored, then the rest be monitored based on the flow conversation property. This theory is stretched by the minimum weak vertex cover of a graph(we have referred in section II). Thus, our aspiration are enable to find more vertexes with degree 2. In a real network, however, almost every nodes' degree more than 2. We substitute it for minimum degree. Experiments demonstrate that this substitution not unprofitable compared to previous algorithm.

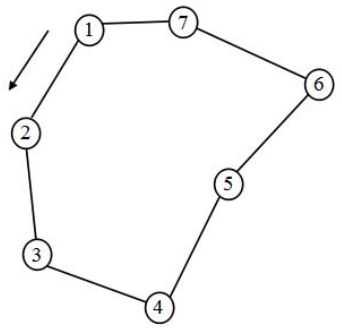


Fig. 3. Graph can be covered by ne monitor .

2) A vertex u ($u \in V$) which is linked to vertex v ($v \in V$) is called a neighbor of v. All neighbors of v ($v \in V$) form the candidate set $N_{(v)}$. Then, we select a vertex from the set $N_{(v)}$ and place monitor between two nodes. Our ALGHEU algorithm consider the minimum degree in $N_{(v)}$ is the best choice;
3) Place a monitor on $E(u, v)$, and cover it, that is, add that edge into set S. Let $E^{'}{}_{(v)}$ be covered edges linked to vertex v. Then, decide whether the quantity of $E^{'}{}_{(v)}$ and $E^{'}{}_{(u)}$ equal to $D(v)-1$ and $D(u)-1$ respectively, if
$$E^{'}_{(v)} = D(v) - 1. \quad (1)$$
Then, covered the rest of edge linked to v, that is, add it into S. And the same with vertex u;
4) Update each node and circulate this process until there is no uncovered edges.

We now analyze the running complexity of algorithm ALGHEU. Assume the graph network contains $|N|$ nodes and $|E|$ edges. Apparently, calculating the $D(v)$ for each node v can be down in $O(|E|)$. Find the vertex u of minimum $D(u)$ from vertex v's neighbor set runs in $O(N)$ time. And then place monitor on the edge, the decision for vertex v can be done in $O(1)$. Determine whether the $D(i)-1$ edges of the vertex have been covered, can be done in $O(N^2)$ time. Each of the above operations will repeats k times. Therefore, the running time complexity of the ALGHEU is $k \cdot (O(N) + O(N) + O(1) + O(N^2))$ = $O(k \cdot N^2)$.

IV. Simulations

In this section, we compare our algorithms with the previous 1-Greedy algorithm [8]. First, we evaluate our algorithms on networks with varying size in section IV-A. Then, in section IV-B, we examine the impact of placing monitors on the number of needed monitors and running time. The experiments have been performed on an Intel(R) Core(TM) i5-2450M processor.

We randomly build some graphs with different number of internal nodes, and generate edges as a probability P_e = 0.03. We use Java language to program and the IDE is eclipse. Meanwhile, we develop a customized Ant Colony Optizimation(ALGACO) as a contrast to demonstrate the performance of our ALGHEU.

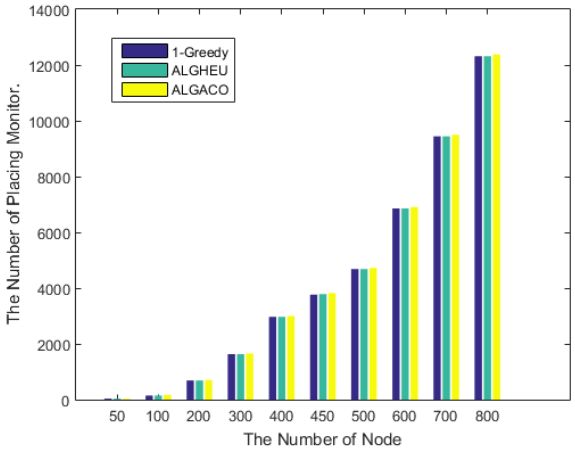
In ALGACO, The maximum iteration number(ant number) of algorithm is set to 5. The *transfer probability* is determined by *pheromone concentration*, we donate *transfer probability* as follows.
$$P_{ij} = \frac{1}{2}(\frac{1}{D(i)} + \frac{1}{D(j)}). \quad (2)$$
And node-choosing relies on the selection strategy based on roulette of ACO. Update *transfer probability* when an ant complete the traversal of n nodes. Among the 5 results, the average result and the best case result will be collected and compared.

A. The impact of network size

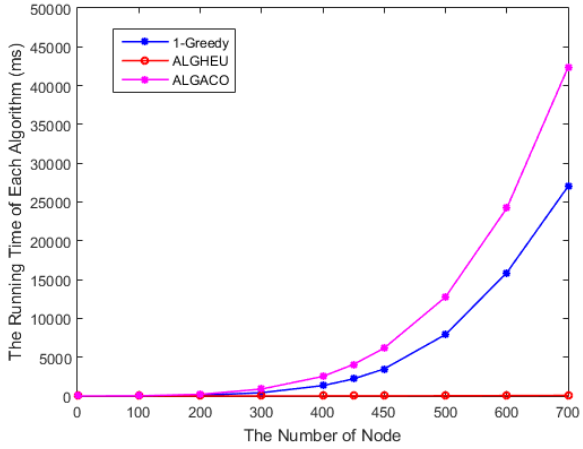
For each network size N ($N \in \{50, 100, 200, 300, 400, 500,600,700,800\}$). We execute the algorithms of Section III, previous 1-greedy algorithm and the ALGACO as a contrast on each graph. For each case (with fixed network size N), we calculate the average number of monitors that used and the average running time in each solution. The experimental results are shown in Fig. 4.

Fig. 4 compares the three algorithms, in terms of the number of monitors that used to test flow and the running time that cost. It can be seen from the figure that the number of monitors, by 1-Greedy and our NEWHEU presented in Section III is closed. Our ALGACO as experimental comparison, is much more than the optimal solution. Furthermore, when the N(the number of node)< 50, solution of 1-Greedy is optimal than the other. For example, on networks with 400 nodes, NEWHEU and 1-Greedy use 2790 and 2791 new monitors, while the ALGACO solution produces 3001 monitors. The running time cost is shown in Fig. 4. In general, for all cases considered, time cost of NEWHEU is lower than 1-Greedy and ALGACO. And with the increase of nodes, from 50 to 800, the percentage of running time cost of 1-Greedy increases 18010.7 times, of ALGACO increases 58.7 times, and of NEWHEU increases 9842.1 times.

Despite the slight improvement in solution quality, our algorithms run much faster than the 1-Greedy algorithm. Table I shows the concrete running time comparison on networks with varying size. On networks with 600 nodes and 7457 edges, our algorithms can find nearly optimal solution in 32.2 milliseconds (by NEWHEU), while the 1-Greedy algorithm runs in 7912.7 milliseconds.



(a) comparison on the number of required monitors



(b) comparison on the running time

Fig. 4. Comparison among the three algorithms, `1-Greedy`, `NEWHEU` and `ALGACO`, on the number of monitors and the running time of three algorithms.

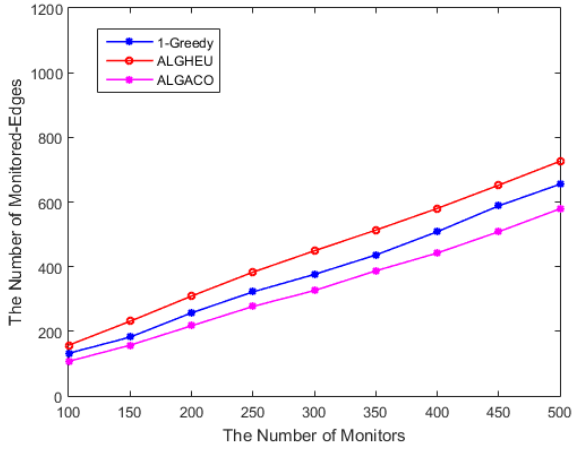


Fig. 5. The number of covered-edges by three algorithm, on networks with different number of monitors, where N = 500.

TABLE I
The running time comparison of three algorithms on networks with different number of node(ms).

node	edge	`1-Greedy`	`ALGACO`	`NEWHEU`
50	76	1.5	4.3	1.2
100	245	12.2	20.3	2.4
200	889	93.6	205.5	0.4
300	1933	417.2	888.7	6.7
400	3369	1337.5	2518	11.6
450	4233	2198.5	4064.8	14.4
500	5183	3482.1	6162.3	20.6
600	7457	7912.7	12749.1	32.2
700	10142	15881.4	24200.1	50.3
800	13116	27017.5	42364.1	71.6

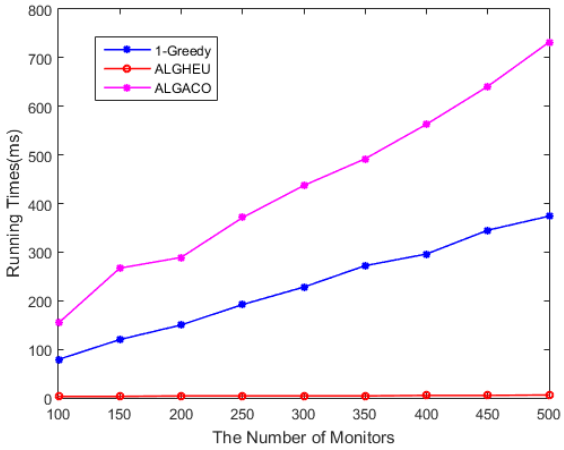


Fig. 6. The running time comparison of the three algorithms, on graphs with different number of monitors, where $N = 500$.

B. The impact of placing monitors

In this section, we evaluate our algorithms on networks with varying amount of monitors. First, we randomly build a graph with $N = 500$ nodes. For each graph, we randomly add different number monitors. The number of mornitors ranges from 100 to 500. The three algorithm `1-Greedy`, `NEWHEU` and `ALGACO`, are executed and compared on the same input instances. All the results are averaged over 10 graphs instances.

Fig. 5 shows the number of covered-monitors executed by our algorithm over `1-Greedy`. From this figure, we can find that, for all cases considered in this experiment, the number of covered-edges executed by algorithm `1-Greedy` less than by `NEWHEU` in average.

Fig. 6 compares the three algorithms in terms of running time on networks with varying amount of monitors. The running time is less than 0.01 second by `NEWHEU` and 0.228 seconds by algorithm `1-Greedy` in average. `ALGACO` as a contrast showed 0.438 seconds.

V. Conclusions

Network flow monitoring is a crucial procedure for optimizing the network infrastructure for better application performance or security issue. The existing reported approaches

suffers from higher running time complexity, and focuses on theoretical analysis. In this paper, we have designed efficient heuristic to accelerate the algorithm of placing monitors to a subset of edges so as to maximize the ability of monitoring, which is particularly useful for large scale networks. Experimental results show that our proposed heuristic can significantly accelerate the previous work without obviously increasing the number of required monitors. On graph networks with 500 nodes and 500 monitors, the two algorithm, `1-Greedy` and our `ALGHEU`, can monitor almost the same number of links, while our algorithms can run as much as 49.67 times faster than `1-Greeedy`.

ACKNOWLEDGMENT

This work was supported by the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant No. 20131201110002, and the Key Laboratory of Computer Architecture Opening Topic Fund Subsidization under Grant No. CARCH201303.

REFERENCES

[1] Scheibelhofer, O., D. M. Koller, P. Kerschhaggl, and J. G. Khinast, "Continuous powder flow monitoring via near-infrared hyperspectral imaging," Proceedings of the *Proceedings of 2012 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pp. 748-753, 2012.

[2] Arattano, M., M. Cavalli, F. Comiti, V. Coviello, P. Macconi, and L. Marchi, "Standardization of methods and procedures for debris flow seismic monitoring," *Engineering Geology for Society and Territory*, vol. 3, pp. 63-67, 2015.

[3] Gu, W., Jia, X., "On a traffic control problem," Proceedings of the *Proceedings of 8th International Symposium on Parallel Architectures, Algorithms and Networks (I-SPAN)*, pp. 510-515, 2005.

[4] Khuller, S., Bhatia, R, Pless, R., "On local search and placement of meters in networks," *SIAM journal on computing*, vol. 32, no. 2, pp. 470-487, 2003.

[5] Dahlhaus, E., Johnson, D.S., Papadimitriou, C.H., Seymour, P.D., Yannakakis, M., "The complexity of multiway cuts (extended abstract)," Proceedings of the *24th ACM Symposium on Theory of Computing (STOC)*, pp. 241-251, 1992.

[6] Goldschmidt, O., Hochbaum, D.S., "Polynomial algorithm for the k-cut problem," Proceedings of the *29th Annual IEEE Symposium on Foundations of Computer Science(FOCS)*, pp. 444-451, 1988.

[7] Saran, H., Vazirani, V.V., "Finding k-cuts within twice the optimal," *SIAM journal on computing*, vol. 24, no. 1, pp. 101-108, 1995.

[8] Chin, F., Chrobak, M., Yan, L., "Algorithms for placing monitors in a flow network," *Algorithmica*, vol. 68, no. 1, pp. 1-15, 2014.

[9] Ritter H, Winter R, Schiller A J. "A Partition Detection System for Mobile Ad-Hoc Networks," *First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, vol. 23, pp. 489C497, 2004.

[10] CHANG Jian-long, YAN Ying, GONG Xue-qing,DAI Dai, ZHOUAo-ying. "SMART:a system for online monitoring large volumes of network traffic," *Journal of Shandong University(Natural Science)*, vol.42, pp. 27-31, 2007.