A Study on Rapid Incremental Maximum Flow Algorithm in Dynamic Network

Yuanyuan Wang
School of Computer Science and Engineering
(Southeast University)
Nanjing, China

Jianhui Ling
School of Foreign Languages
(Southeast University)
Nanjing, China

Sihai Zhou, Yundi Liu, Weicheng Liao, Baili Zhang
School of Computer Science and Engineering
(Southeast University)
Research Center for Judicial Big Data
(Supreme Count of China)
Nanjing, China
zhangbl@seu.edu.cn

***Abstract*—Maximum flow is a key measurement for the capacity of a flow network. When malfunction or damage occurs in branches of a dynamic network, it is urgent in many applications to identify whether the damaged network is still able to transfer demanded flow *d*. Obviously, recalculating maximum flow in the new network with the traditional algorithm is not a recommended solution for its higher time-complexity. As is known, in many existing maximum flow algorithms, the search for augmenting paths is critical but highly time-consuming. Thus, this paper proposes an incremental maximum flow algorithm based on augmented routing tree (IMFA_ART), directly producing augmenting paths without complex calculation. The algorithm caches the information of simple paths during calculating maximum flow in the original network. The cached data will be utilized to get the augmenting paths whenever network topology changes, without any complex calculation in residual network. In addition, in order to avoid traversing those invalid simple paths that contain saturated edges, an augmented routing tree is proposed to index all simple paths. With the aid of this tree, the next augmenting paths can be found sequentially to achieve maximum flow by traversing from the root to a leaf. The time complexity is determined by the height of ART, which is far less than the number of augmenting paths. Therefore, the algorithm performance can be improved significantly. Experiments show that IMFA_ART has greater advantage in time performance over Dinic, the most famous maximum flow algorithm.**

Keywords—maximum flow; incremental algorithm; augmented routing tree

I. INTRODUCTION

Maximum flow is one of the classic topics of network optimization, and has many applications in traffic transportation network, electrical power systems, and drainage network. In the past five decades [1], many efficient algorithms have been developed to solve problems with maximum flow. Firstly, Dantzig and Ford-Fulkerson investigated two pseudopolynomial time methods[2,3,4]. After that, Edmonds Karp and Dinic proposed two polynomial time algorithms[2,3,5], on the basis of the algorithm presented by Ford-Fulkerson. In 1998, Goldberg and Rao put forward the binary blocking flow method[6]. This is the first time to decrease the ballpark from nm to $\min\{n2/3, m1/2\}m$. However, it is quite difficult for the algorithm to implement a large amount of network transition. Currently, some researchers have proposed specific algorithms to calculate maximum flow in special networks. For example, Zhang Xianchao focused on how to determine the maximum flow in networks where each edge and node have their capacity limit, and presented a solution to the problem by the minimal cuts[7,8]. This algorithm produces a result within the time O(nlogn). Moreover, in order to get maximum flow quickly, some scholars began to conduct research on approximation methods. Lee, Daitch and Richard et al. who studied undirected networks, put forward many effective approximation algorithms[9-11]. Zhao Shu presented a Contracting Neighbor-node-set Approach[12], on the basis of the entropic space theory. M. Buzdalov applied genetic algorithm to obtain an approximate result. In addition, in order to meet demands in application, researchers have expanded the study on the maximum flow problem, and introduced many new concepts and solutions , such as the minimum cost and maximum flow problem[14,15], the network flow reliability problem[16,17] and the most reliable maximum flow problem[18], the best bipartite graph matching problem[19,20].

In practice, network is often dynamic. That means the edge capacity or the network topology may change over time. As for the change of edge capacity, with a focus on how to achieve the maximum flow at a certain moment in dynamic network, Zhang Ling presented a new method on the basis of the entropic space theory[21], by which the dynamic network is decomposed to many static networks. By analyzing these static networks, a solution to dynamic network problem is produced. When the network topology changes, real-time computation is often required. It is true that recalculating the maximum flow in small networks with traditional algorithms may satisfy this requirement. But it is challenging to recalculate the maximum flow in huge networks due to the higher time complexity. As is known, the incremental algorithm is widely used in various real-time solutions[22-24]. Thus, this paper proposes an incremental algorithm to compute the maximum flow quickly. As searching a next augmenting path is critical and yet highly time-consuming, it is important to simplify or optimize this process to improve the algorithm efficiency. Obviously, original network G contains most of the information in damaged network G', such as nodes, edges, simple paths, etc. Therefore, if the simple paths of the original network are cached, augmenting paths may be achieved directly in cache when

978-1-5386-5059-2/18/$31.00 ©2018 IEEE
DOI 10.1109/IC3.2018.00011

network topology changes, without any complex calculation in residual network. This proposal can effectively simplify the query process of augmenting paths. Moreover, Augmenting routing tree (ART) is proposed in this paper to index all simple paths, for the purpose of avoiding traversing invalid paths with saturated edge(s). Actually, the nodes from the root to a leaf in ART represent augmenting paths. If the height of ART is H (it is far less than the number of augmenting paths), all valid augmenting paths can be obtained within O(H) time through one traversal from the root to a leaf. On the basis of ART, this paper develops an incremental maximum flow algorithm IMFA_ART, which can significantly improve the algorithm efficiency.

This paper is composed of three parts. It starts with the theory of incremental maximum flow algorithm and augmented routing tree, which is immediately followed by the incremental maximum flow algorithm based on ART. Finally, the performance of IMFA_ART is verified through a series of experiments.

II. INCREMENTAL ALGORITHM

A. overview

Most maximum flow algorithms are derived from the method proposed by Ford-Fulkerson. This method is used for the constant search of augmenting paths in residual network, and the accumulation of the augmented flow, until there is no augmenting path in the residual network. The common method of labeling is applied to get augmenting paths in many algorithms, such as Dinic, the shortest path combination algorithm (SPC), and improved SPC. Among them, SPC layers network on the basis of breadth-first traversal to label each node. However, once an augmenting path is located, nodes label should be renumbered by breadth-first traversal. This process is time-consuming. For this reason, ISPC algorithm updates nodes' labels with some auxiliary information to improve algorithm efficiency in the process of searching for augmenting path. However, this algorithm is not able to reduce time complexity. In a world, the process of seeking next augmenting paths is a critical part of the algorithm. Therefore, effectively simplifying or optimizing this process can significantly promote the computational efficiency.

Considering that augmenting paths are derived from the simple paths of network and that simple paths of damaged network are included in original network, this paper proposes a method of caching all original simple paths. Whenever network topology changes, augmenting paths can be searched directly in cache. Moreover, to accelerate the seeking process, ART is proposed to index the next augmenting paths. Therefore, based on ART maximum flow can be quickly produced.

B. ART and its constructing algorithm

The rules of constructing ART are as follows:

1) The tree node has two fields, as shown in Fig. 1. Pvalue indicates the simple path address; NextTreeNode is directed to children nodes.

Pvalue	NextTreeNode[EDGE_NUM]

Fig.1 ART node information

2) The *Pvalue* of the root is directed to the shortest one of simple paths.

3) From the root to a leaf, the length of simple path is increasing.

4)*Saturated_edges* records all saturated edges from the root to the current node. The simple path, to which the next traversal node is directed, doesn't contain any edges in *Saturated_edges*.

5) If path p contains edge e_i, *NextTreeNode[i]* is directed to the child node, which represents the next augmenting path when e_i is saturated; otherwise *NextTreeNode[i]* is NULL.

According to the rules, ART can be constructed as follows:

Algorithm 1. createART

```
Input:    pathlist, edge_num,
                saturated_edges[EDGE_NUM]= {0} , n=0
Output:   root
1)  if(pathlist==NULL) return NULL;
2)  int i=0;
3)  if   the simple path contains the saturated_edges
4)       return CreateART(pathlist->next,edge_num,
                        saturated_edges,n);
5)  else
6)       create tree node *root, root ->pvalue=pathlist;
7)  for(i=1;i<=edge_num;i++)
8)       if this path contain edge i {
9)            saturated_edges [n++]=i;
10)           root ->nextTreeNode[i]=
CreateART(pathlist->next,
                edge_num, saturated_edges, n);
11)           n--;}
12)      else      root ->nextTreeNode[i]=NULL;
13) return root;
```

In the algorithm, all simple paths are saved in *pathlist* in the order of the length of simple paths. According to the rules of ART, each of the nodes from the root to a leaf represents an augmenting path. Therefore, ART saves all possible valid augmenting paths. For a graph with $|E|$ edges, the network has $|E|$ augmenting paths at most. But in practice, the height of ART H is far less than $|E|$. For example, Fig. 2 is a network with 5 nodes and 9 edges, where e_i/c represents the edge label and its capacity. Table 1 shows all simple paths in Fig. 1, and Fig. 3 is the corresponding ART.

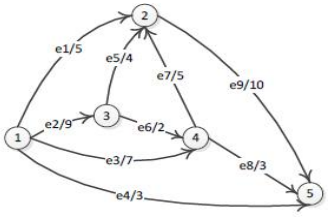

Fig. 2. a network of V5E9 Table 1. all path vectors of Fig. 1

TABLE 1. ALLPATH VECTORS OF FIG.1

Path	Vector
P1	(0,0,0,1,0,0,0,0,0)
P2	(1,0,0,0,0,0,0,0,1)

P3	(0,0,1,0,0,0,0,1,0)
P4	(0,1,0,0,1,0,0,0,1)
P5	(0,1,0,0,0,1,0,1,0),
P6	(0,0,1,0,0,0,1,0,1)

C. IMFA_ART algorithm

On basis of augmented routing tree created by algorithm 1, an incremental maximum flow algorithm is proposed, by which the next augmenting paths can all be found within *O(H)* time to achieve the maximum flow, through one traversal from the root to a leaf. The processes are as follows:

Algorithm 2. MFIA_ART

Input: root, edge_n, edge_capacity *Ce, fail_edge e_i
Output: max flow of network when e_i is fail

```
1) Int maxflow=0, augflow=0,saturated=0;
2) set the fail edge's capacity Ce[i]=0;
3) PathTreeNode *p=root;
4) While(p){
5)    augflow=GetPathAugFlow(Ce,p->pvalue, edge_n);
6)     saturated=UpdateEdgeCap(Ce,augflow,p->pvalue );
7)    maxflow+=augflow;
8)    p=p->nextTreeNode[saturated];}
9) Return maxflow
```

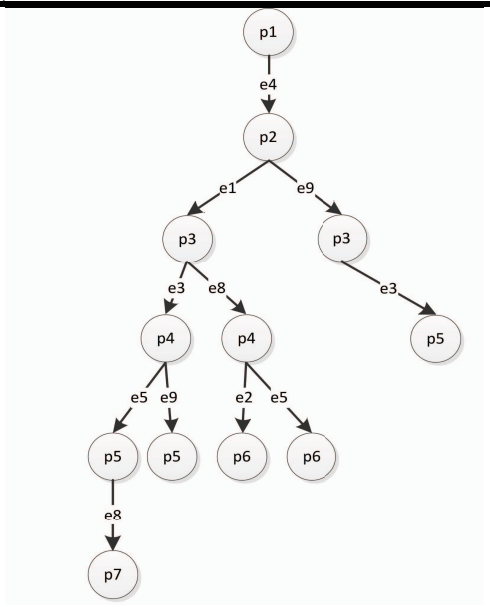

Fig. 3. ART of V5E9

In this algorithm, a pointer *p* is directed to the root node firstly, and then the augmented flow value of path represented by root is obtained by calling the function of *GetPathAugFlow*. After that, *UpdateEdgeCap* is called to update the remaining capacity of each edge and then return to the *saturated* which saves the saturated edge. Next, the algorithm seeks the next node according to *saturated*. The above steps are repeated until the pointer *p* is NULL. For example, in Fig.2, at first, when p is directed to p1, the augmented flow is 3 and the saturated edge is e_4. According to e4, p is directed to p2, and then aug-flow 5 and saturated edge e_1 is obtained. The rest is p3 (left)->p4(right)->p6. If the height of ART is H, augmenting paths can be found within *O(H)*, without traversing all simple paths in cache or undergoing complex calculation process in residual network as is down other algorithms.

III. EXPERIMENTS

A. Environment and the Data Set

In this section, a series of experiments were conducted to test the accuracy and performance of the algorithm. The experiments platform is a PC (Intel Core, CPU i7-3700, 3.4GHz, RAM 8GB, 64 bit windows 7 operating system). Algorithms are coded with C/C++ in visual studio 2010. All data sets used in the experiment are generated by the NETGEN that is a directed graph generator cited from literature [25]. The specific graphs include the following categories:

1) 8 graph groups are generated by NETGEN: V6E10, V8E14, V10E18, V12E22, V14E26, V16E34, V18E40, V20E56. VnEm represents a graph with n nodes and m edges. Each group has 10-20 graphs. The capacity of every edge is randomly generated. The selection rule of source *s* and sink *t* is derived from the literature [20]. According to the rule, two nodes with most simple paths are selected as the source *s* and sink *t*.

2) Another 8 graph groups are generated by NETGEN. These graphs have same node number and different dense *d (d=(2*|E|)/(|V|*(|V|-1))* [26]). In this paper 16 is given as the number of nodes, the density ranges from 0.2 to 0.6. Each group has 5-10 different graphs. Source and sink selection rule is same as the first category.

B. Performance of the Proposed Algorithm

In this experiments, the classic maximum flow algorithm Dinic is adopted for comparison. Firstly, time consumption and space consumption of algorithm are compared in different size graphs. Then, the time consumption of the algorithms is compared in different density graphs.

Experiment 1. With the first category data, we compare the algorithms performance in different graph size.

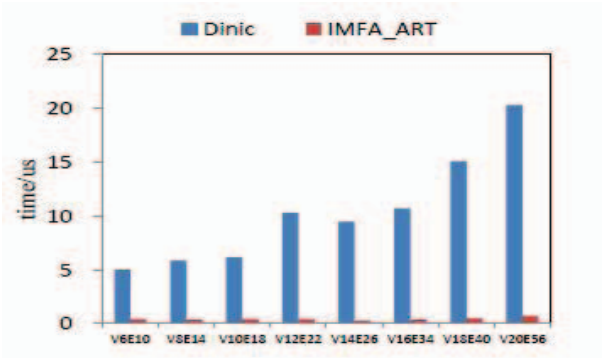

Fig. 3(a). The time consumption of algorithms

Fig 3(a) shows the time consumption of Dinic and IMFA_ART. Firstly, it can be found that IMFA_ART has greater advantage in time performance over Dinic. Secondly, the time consumption of these algorithms rises with the increase of graph size, because the number of simple paths and augmenting paths will increase as a graph grows bigger. As a result the time for seeking augmented paths and traversing ART becomes longer, which results in the decreased performance of the algorithms. However, the time consumption of IMFA_ART rises slowly when graph size grows, because IMFA_ART only traverses up to H nodes, which is far less than the number of simple paths.

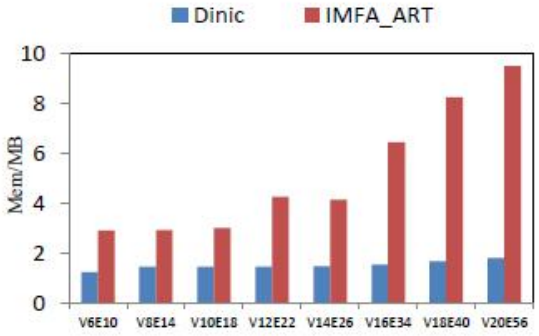

Fig. 3(b). The space consumption of algorithms

Fig. 3(b) indicates the space consumption of two algorithms. It can be found the space of Dinic is smaller and increases slowly, and that the space consumption of IMFA_ART is much bigger, as the result of our strategy of "trading space for time". Meanwhile, the width of ART will grow rapidly with increasing number of simple paths, so the space consumption of IMFA_ART increases concurrently.

Experiment 2. With the second category data, we compare the performance of the two algorithms in different graph density. In this experiment there is a fixed number of nodes and the density varies from 0.2 to 0.6.

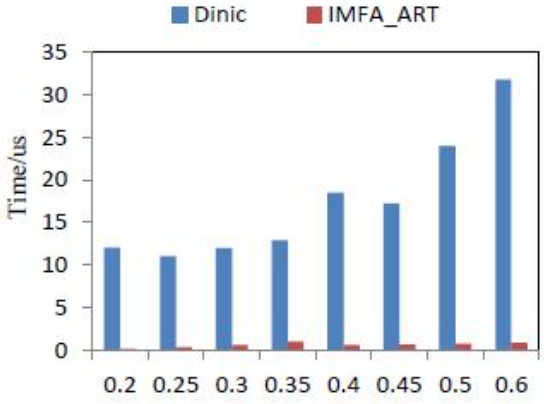

Fig. 4. Algorithms time consumption in different density

Figure 4 shows that the time consumption of algorithms increases along with the density. The increasing density means more edges. As a result, the number of simple paths and augmenting paths increases correspondingly, which leads to more time consumption. However, IMFA_ART algorithm is far superior to Dinic in time consumption regardless of the changing graph density.

IV. CONCLUSION

In a dynamic network, once a network edge is damaged, it will take long time to compute the maximum flow with the traditional algorithm. However, in this case, the maximum flow can be calculated rapidly with our algorithm by means of ART which indexes all simple paths to get all augmenting paths directly. Experiments demonstrate that time consumption of IMFA_ART algorithm has been considerably reduced in the search of the next augmenting paths. In conclusion, our incremental algorithm has great advantage in time complexity, which results from necessary cached data and reasonable index structure. To optimize our algorithm, the research focus will be given onto the decrease of the space consumption by means of combining same nodes in ART.

ACKNOWLEDGMENT

This work was supported by the Fundamental Research Funds for the Central Universities under project NO. 3209007304

REFERENCES

[1] A. V. Goldberg, "Recent Developments in Maximum Flow Algorithms (Invited Lecture)," Scandinavian Workshop on Algorithm Theory Springer-Verlag, 1998, pp. 1-10.

[2] Cormen, Thomas T, C. E. Leiserson, and R. L. Rivest, Introduction to algorithms. Beijing: MIT Press, 2002.

[3] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, "Network Flows: Theory, Algorithms and Applications," Journal of the Operational Research Society, vol. 45, no. 11, 2005, pp. 294-356.

[4] L. R. F. Jr, D. R. Fulkerson, A. Ziffer. "Flows in Networks," Mathematics of Computation, vol. 18, no. 4, 2009, pp. 319-331.

[5] J. Edmonds, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," Springer-Verlag New York, Inc, vol. 19, no. 2, 1972, pp. 248-264.

[6] A. V. Goldberg, and S. Rao, "Beyond the flow decomposition barrier," J Assoc Comput Mach," vol. 45, no. 5, 1998, pp. 783-797.

[7] Zhang Xianchao, Wan Yingyu, and Chen Guoliang. "Approaches to the Minimum Cut Problem in a Class of Practical Networks," Journal of Software, vol. 14, no. 5, 2003, pp. 885-890.

[8] Zhang Chaoxian, Jang He, and Chen Guoliang, "Minimum Cuts and Maximum Flows in Directed Planar Networks with Both Node and Edge Capacities," Chinese Journal of Computers, vol. 29, no. 4, 2006, pp. 543-551.

[9] YT Lee, S Rao, and N Srivastava, "A new approach to computing maximum flows using electrical flows," Forty-fifth ACM Symposium on Theory of Computing, 2013, pp. 755-764.

[10] S.I. Daitch and D.A. Spielman, "Faster approximate lossy generalized flow via interior point algorithms," ACM symposium on Theory of computing, 2008, pp. 451-460.

[11] Richard Peng, "Approximate undirected maximum flows in O(mpolylog(n)) time," Twenty-Seventh Annual ACM Symposiums on Discrete Algorithms, 2016, pp. 1862-1867.

[12] Zhao Shu, Xu Xiansheng, and Hua Bo, "Contraction network for solving maximum flow problem," ACM SIGKDD Workshop on Mining Data Semantics, 2012.

[13] Megha Sharma, and Diptesh Ghosh, "Speeding up the estimation of the expected value of maximum flow through reliable networks," IEEE Transactions on Reliability, vol. 62, no. 1, 2013, pp. 105-115.

[14] Makhlouf Hadji, and Djamal Zeghlache, "Minimum cost maximum flow algorithm for dynamic resource allocation in clouds," IEEE Fifth International Conference on Cloud Computing, 2012, pp. 876-882.

[15] T.H Szymanski, "Achieving minimum-routing-cost maximum-flows in infrastructure wireless mesh networks," IEEE Wireless Communications & Networking Conference, 2012, pp. 2031-2036.

[16] Zhao Peixin, and Zhang Xin, "A Survey on Reliability Evaluation of Stochastic-Flow Networks in terms of Minimal Paths," International Conference on Information Engineering & computer science, 2009, pp. 1-4.

[17] Chin-Chia Jane, Jsen-Shung Lin, and John Yuan, "Reliability Evaluation of a Limited-Flow Network in Terms of Minimal Cutsets," IEEE Trans. Reliability, vol. 42(3), pp. 354-368.

[18] Cai Wei, Zhang Baili, and Lv Jianhua. "Algorithm of the Most Reliable Maximum Flow on Uncertain Graph," Chinese Journal of Computers, vol. 35(11), 2012, pp. 2371-2380.

[19] Neha Bora, Swati Arora, and Nitin Arora, "An Efficient Algorithm for Calculating Maximum Bipartite Matching in a graph," International Journal of Advanced Computer Research, vol. 3(10), 2013.

[20] Gu Tianlong, Chang Liang, and Xu Zhoubo. "A Novel Symbolic Algorithm for Mximum Weighted Matching in Bipartite Graphs," International Journal of Comunications, Network and System Sciences, vol.4(2), 2012.

[21] Zhang Ling. "The Concept of Max-Flow and Its Properties in Dynamic Networks," PR & AI, vol. 26, no. 7, 2013, pp. 609-614.

[22] Mirayh Kas, Matthew Wachs, Kathleen M. Carley, and L. Richard Carley, "Incremental algorithm for updating betweenness centrality in dynamically growing networks," International Conference on Advances in Social Networks Analysis & Mining, 2013, pp. 33-40.

[23] Rachel Ben-Eliyahu-Zohary. "An Incremental Algorithm for generating all Minimal Models. Artificial Intelligence," vol.169, no. 1, 2005, pp. 1-22.

[24] MC Hsieh, YS Lee, and SJ Yen. "An efficient Incremental Algorithm for Mining Web Traversal Patterns," IEEE International Conference on E-business, 2005, pp. 274-281.

[25] Klingman D, Napier A, and Stutz J. "NETNEN: A program for generating large scale capacitated assignment, transportation and minimal cost flow network problems," Management Science, vol. 20, no. 5, 1974, pp. 814-821.

[26] Johnson D. "Finding all the elementary circuits of a directed graph," SIAM Journal on Computing, vol. 4, no. 1, 1975, pp. 77–84.