A MPI Parallel Algorithm for the Maximum Flow Problem

JIANG Jincheng[1] WU Lixin[2],

[1]Institution of Spatial Information Science and Technology, Academy of Disaster Reduction and Emergency Management, Beijing Normal University, Beijing 100875, China
Email: jiangjincheng0305@126.com

[2]Institution of Spatial Information Science and Technology, Academy of Disaster Reduction and Emergency Management, Beijing Normal University, Beijing 100875, China
Email: awulixin@263.net

1. Introduction

Network analysis remains one of the most significant and persistent research and application areas in GIS[1]. The problem of computing a maximum flow in network analysis is a fundamental combinatorial problem, with many applications in transportation planning, operations research and resource scheduling, i.e. Numerous serial max-flow algorithms have been developed over the past 50 years to improve complexity bounds. Andrew V. Goldberg[2] summarized the development of the algorithm in 1998. In general, there are two principal categories for solving the maximum problem: augmenting path algorithm [3-5] and push-relabel algorithm[6-9]. The *hipr* algorithm, a highest-level variant of the push-relabel algorithm, was found to be the one with best performance both in theory and practice [8] [9], the complexity has been improved to $\mathrm{O}(n^2\sqrt{m})$. However, the efficiency to solve the maximum flow problem for large-scale network still remains the bottlenecks in practical application.

In recent years, parallel computing has become an effective solution to improve computation speed. While there are few improvements in parallel algorithm for max-flow problem utilizing parallel machines. Most existing parallel algorithms[10-13] are fine-grained parallelism, which need too much interconnection or communication, leading to low speed-up. Consequently, coarse-grained parallelization has its unique advantages to reduce the times of exchanging messages.

In this paper, a parallel algorithm with MPI (Message Passing Interface) for max-flow problem is presented. Assuming a graph of *n* nodes and *m* arcs is partitioned into several regions. The flow inside each region is pushed to its boundary through executing *hipr* algorithm iteratively, and a special method is given to discharge the flow out of the region, reducing message passing by three improvements which are different from common push-relabel algorithm. The features of MPI parallel algorithm include: 1) we do not try to calculate the distances between boundary nodes and sink node directly, but by a distance function which compute the distance from region to sink node, and the relationship between boundary node and its adjacent regions; 2) the method of discharge flow out of region various with the flow on boundary arcs, with the state of nodes in adjacent regions and with the distance of region and sink node; 3) we do not try to push the flow along the path which need too much message passing. We tested our parallel algorithm on several types of networks used in the first DIMACS Implementation Challenge, and found that the parallel algorithm has very good acceleration ratio as to sequential algorithm for most types of sparse networks, even beyond our expectation.

2. Methodology

2.1 Network Partition

As the complexity of the *hipr* algorithm we used in each region is $O(n^2\sqrt{m})$, where *n* is the number of nodes and *m* is the number of arcs, the computing time mainly depends on *n*, not *m*. We partition the network to several regions with approximates number of nodes by BFS (Breadth First Search) starting from sink node. The arcs connecting two regions are taken as boundary arcs, the nodes of boundary arcs are defined as boundary nodes.

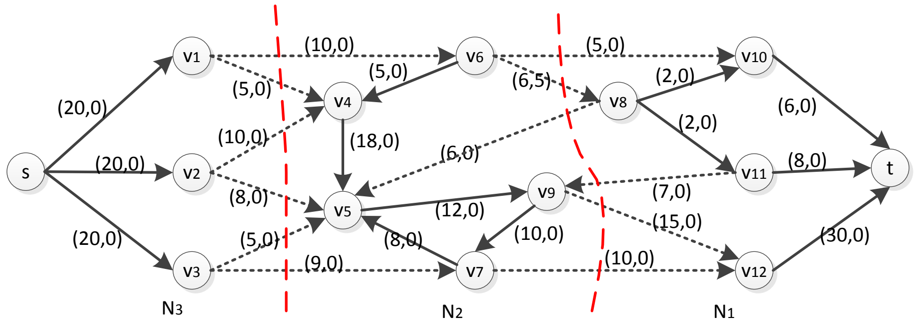


Figure 1. An example of network partition.

2.2 Push Flow Iteratively

Three steps are executed iteratively to push flow to sink until there are no more active nodes (the amount of in-flow larger than out-flow) in all regions except for the source node *s* and the sink node *t*.

(1) label boundary nodes.

There are three types of labels for boundary nodes to choose, the type-I and type-II have chance to push the flow to *t*, while type-III has no chance any more. The method of labelling boundary nodes is as follow: if there are no feasible boundary out-arcs for node v_1 to push the flow to node v_2 (v_2 belongs to different region and the label of v_2 is not type-III), then the label of v_1 is set to type-III. We distinguish the type-I from the type-II by the distance from boundary nodes to *t*, the label of node v_1 with short distance is set to type-I, otherwise to type-II.

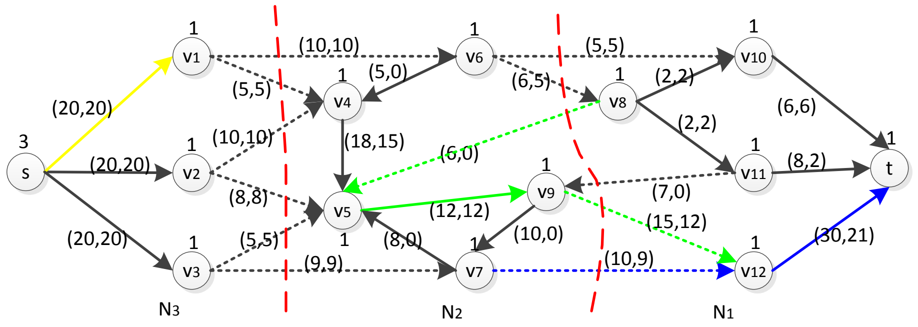


Figure 2. The way to label boundary nodes

As in figure 2, the label of node v_1 will be set to type-III, v_7 to type-I, and v_8 to type-II, respectively.

(2) Push flow inside region to boundary

The *hipr* algorithm is used to push the flow inside region to its boundary, but the features of our method is pushing flow hierarchically, i.e., the flow is pushed to the type-I nodes, then to the type-II nodes and to the type-III nodes lastly.

⑶ discharge flow out of region

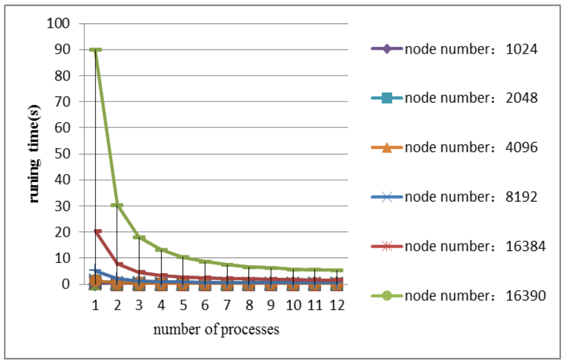
The method of discharging the flow in boundary nodes out of the region is similar to the way as *label boundary nodes* operation. The flow in boundary nodes of type-I and type-II label is pushed along the boundary out-arcs found in *label boundary nodes* operation. We push the flow in the type-III nodes along the boundary in-arcs.

2.3 Region Merger

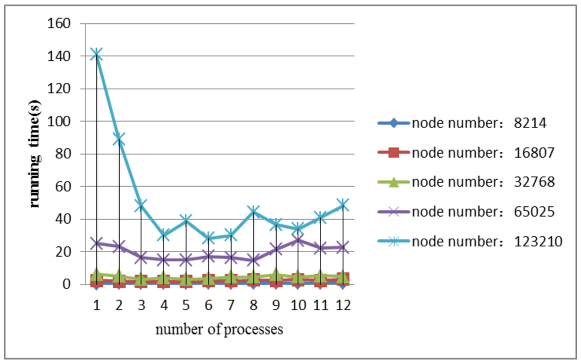
To reduce the cost of message passing, some feasible paths cross boundary are ignored. So it is necessary to judge whether optimal solution has been obtained through BFS from *t*. If not, all regions will be merged to a whole network with the help of MPI, and then the remaining work will be done in sequences.

3. Experiments

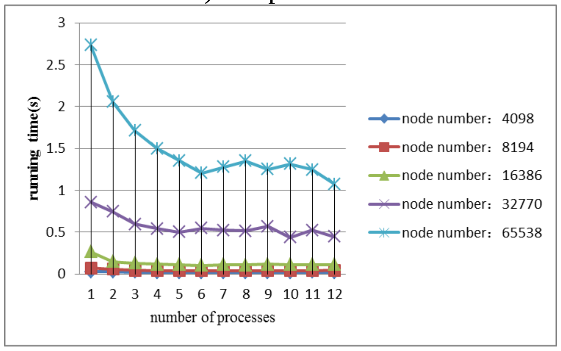
We tested the MPI parallel algorithm on a graphic workstation, which has four 64 bit Quad-Core processors running at 2.4 GHz with 64 GB memory. Visual Studio 2010 is used for coding to generate the executable file. Four types of networks, which were used in the first DIMACS Implementation Challenge and used by many existing max-flow algorithms, were applied to test the performance. The testing result is shown in figure 3:



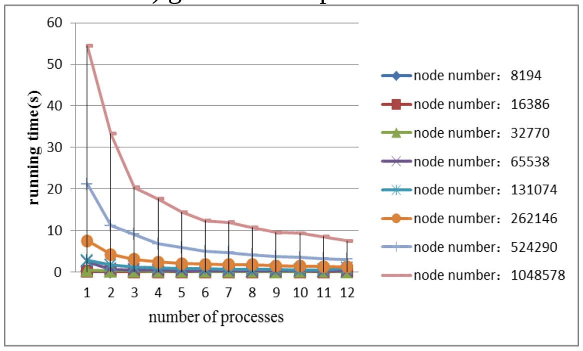
a) AK problem



b) genrmf-wide problem



c) Washington-Line-Moderate problem



d) Washington-RLG-Long problem

Figure 3. MPI performance on different problems

4 Conclusion

We have proposed a new distributed algorithm with MPI for maximum flow problem for large-scale sparse graph. Several effective methods are used to reduce message exchange and additional calculation is added to make sure the optimal solution can be obtained.

Experimental tests on real instances shows that the MPI parallel algorithm performs good parallel efficiency, and sometimes even beyond 100%, the reason is that many unnecessary operations happened in boundary regions are ignored.

The parallel algorithm presented in this paper is of great significance for many transportation and utility applications based on GIS. Especially in the emergency situation, our parallel algorithm plays an important role for resource scheduling.

5. References

[1] Kevin M. Curtin, 2007, Network Analysis in Geographic Information Science: Review, Assessment, and Projections, *CaGIS*. 34: 103–111.

[2] A V Goldberg, 1998, Recent developments in maximum flow algorithms, *Lecture Notes in Computer Science*, 1432: 1-10.

[3] Ford, Jr. L. R., D. R. Fulkerson. 1957. A simple algorithm for finding maximal network flows and an application to the Hitchcock problem. *Canad. J. Math.* 9*:*210–218.

[4] Dinic E. A. 1970. Algorithm for solution of a problem of maximal flow in a network with power estimation. *Soviet Math Doklady*, 11: 1277-1280.

[5] Karzanov A. V. 1974. Determining the maximal flow in a network with a method of preflows. *Soviet Math. Dokl*, 15: 434-437.

[6] J. Edmonds, R.M. Karp, 1972, Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*. 19(2): 248-264.

[7] Goldberg, A. V., R. E. Tarjan. 1988. A new approach to the maximum flow problem. *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, New York, NY, USA, 136-146.

[8] Ahuja, R. K., T. L. Magnanti, J. B. Orlin. 1993. Network Flows: Theory, Algorithms, and Applications. Prentice-Hall, Englewood Cliffs, NJ.

[9] Goldberg, A. V., B. V. Cherkassky, 1997, On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, 19(4): 390-410.

[10] A. Iossa R. Cerulli, M. Gentili, 2008. Efficient Preflow Push Algorithms. *Computers & Oper. Res*, 35(8): 2694–2708.

[11] G. C. Caragea and U. Vishkin, 2011, Brief announcement: better speedups for parallel max-flow. *In Proceedings of the 23rd ACM symposium on Parallelism in algorithms and architectures*, New York, NY, USA, 131-134.

[12] B. Hong and Z. He, 2011. An asynchronous multithreaded algorithm for the maximum network flow problem with nonblocking global relabeling heuristic. IEEE Transactions on Parallel and Distributed Systems, 22(6):1025–1033.

[13] Mujahed Eleyat, Dag Haugland, Magnus Lie Hetland, 2012, Parallel algorithms for the maximum flow problem with minimum lot sizes, *Operations Research Proceedings 2012,* 83-88.