Large-scale graph max flow acceleration algorithm based on GraphChi

Wei Wei [1, a], Ruqing Zhang [2, b] and Yongxin Zhang[1, c]

[1] School of Information Science and Engineering, Henan University of Technology, Zhengzhou 450000, China;

[2] School of Information Science and Engineering, Henan University of Technology, Zhengzhou 450000, China.

[3] School of Information Science and Engineering, Henan University of Technology, Zhengzhou 450000, China.

[a]nsyncw@163.com, [b]zhangruq_haut@163.com, [c]Dxin_haut@yeah.net

Abstract

Max flow is an important problem of graph calculation, and many practical scenarios can be effectively solved if they transformed into max flow problems. The max flow has been discussed from many perspectives, but there are still some problems. Aiming at the problem that graph segmentation in some distributed systems has high computational complexity, this paper presents a max flow acceleration algorithm based on GraphChi framework. First, use the cut vertex in the original graph to construct an overlay graph and divide the original graph into several subgraphs, given source and sink vertex pair, determine the unique path in the overlay graph, then parallel computethe max flow of each subgraph on the GraphChi, finally find the minimum value of the max flow of these subgraphs, which is the max flow of the original graph. The experiments in TIGER/Line show that the proposed algorithm can significantly shorten the max flow calculation time of large-scale graphs, and has a good acceleration effect.

Keywords

Graph calculation; max flow; GraphChi.

1. Introduction

With the development of computer technology, many practical problems can be handled by the method of solving the max flow problem. In recent years, many experts and scholars have adopted different methods to solve the max flow problem [1-4].

[5]constructed a reduced graph by merging the vertices of the original graph to calculate the max flow, but it can't achieve the parallel computation to accelerate the max flow. [6] randomly divided the original graph into multiple subgraphs by the edges in the graph, and combined the max flow of each subgraph to got the approximate max flow. [7] proposed simple-path locality based max-flow acceleration algorithm, by building an overlay and calculating the max flow of the necessary nodes, the computation time was reduced to about 4.3 times that of other algorithms. [8] used the Edmonds-Karp algorithm to solve the max flow problem, the built-in pregel interface in Spark was used to improve computation speed, but the communication cost of the algorithm was relatively large.

To address above problems, we propose max flow acceleration algorithm based on the parallel graph computing framework GraphChi[9] to process large-scale graphs. First, use the cut vertex in the original graph to construct an overlay graph and divide the original graph into several subgraphs, after given the source and sink vertex, determine the unique path in the overlay graph and each subgraph

on the path, then use the parallelism of GraphChi to carry out parallel computation of multiple subgraphs, finally the minimum value of the max flow of these subgraphs is taken as the max flow in the original graph. The results show that the algorithm to compute the max flow in parallel on the GraphChi framework can significantly accelerate the calculation of the max flow in large-scale graphs, reduce a large number of redundant calculations, and effectively reduce the time complexity.

2. Model

Given a graph $G=(V,E)$, where V is the set of vertices, E is the set of edges. For vertex pair (s,t), find feasible flow $\hat{f}(s,t)$ in the graph so that the flow from the vertex s through the intermediate vertex v to t is the maximum flow, thus $\hat{f}(s,t)$ is called the max flow , and this problem is called the max flow problem. According to the above, the max flow mathematical model established can be expressed as

$$\hat{f}(s,t)=\max\left(\sum_{v\in\delta^{+}(s)} f(s,v)\right) \tag{1}$$

The mathematical model of the constrained feasible flow between s and t can be expressed as:

$$s.t.\begin{cases}\sum_{v\in\delta^{+}(s)} f(s,v)=\sum_{v\in\delta^{-}(t)} f(v,t)\ (a)\\ 0\le f_e\le c_e\quad \forall e\in E\ (b)\\ \sum_{w\in\delta^{-}(v)} f(w,v)=\sum_{w\in\delta^{+}(v)} f(v,w)\ \forall v\in V-\{s,t\}(c)\end{cases} \tag{2}$$

Where $\delta^{-}(v)=\{w\in V|(w,v)\in E\}$, $\delta^{+}(v)=\{w\in V|(v,w)\in E\}$.

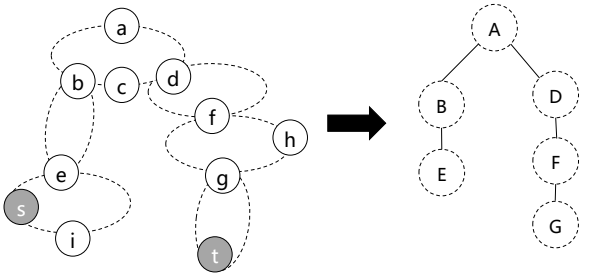
According to equation (2), the value of outflows from s is equal to that of flows into t; The flow of each edge is non-negative and the feasible flow is not more than the capacity flow; The total flow from w to v is equal to the total flow from v to w except the vertex pair (s,t).

Through the above discussion, the mathematical model established according to the max flow theory and the constraint conditions of the feasible flow $\hat{f}(s,t)$ can realize the calculation of the max flow.

3. Algorithm

3.1 Construct Overlay Graph and subgraphs

Construct an overlay graph and divide the original graph into several subgraphs by using the cut vertex in the original graph is shown in Figure 1.



(a)original graph (b) overlay graph

Fig.1 Overlay graph construction

The algorithm uses DFS to traverse from the root vertex of the original graph. At the beginning of the algorithm, each neighbor vertex of the current vertex is traversed. If hasn't been visited yet, push the vertex pair into stack and the value of is incremented by 1, and then the DFS traversal is

start from . As the vertex are traversed continuously, the value is update to be traceable to the smallest through the not-father vertex. When traversal to the leaf node and backtrack, if has been visited and , infer is the cut vertex and will be mapped to in the overlay. If the top element of the stack is either or , pop the top element of the stack, add the vertex that are not into the subgraph and maps to vertex in the overlay graph. After map all cut vertex to the overlay, the corresponding links are established in the overlay graph according to the relationship between the cut vertex in the original graph. Save the overlay graph and the subgraphs to calculate the max flow between the source vertex and sink vertex.

3.2 Parallel Computation Of The Max Flow Based On The Graphchi

The diagram of parallel computing max flow based on GraphChi is shown in Fig.2. The path formed by the vertex E\B\A\D\F\G is the unique path in the overlay graph after given the source vertex s and sink vertex t, and each vertex in the overlay graph corresponds to a subgraph in the original graph.

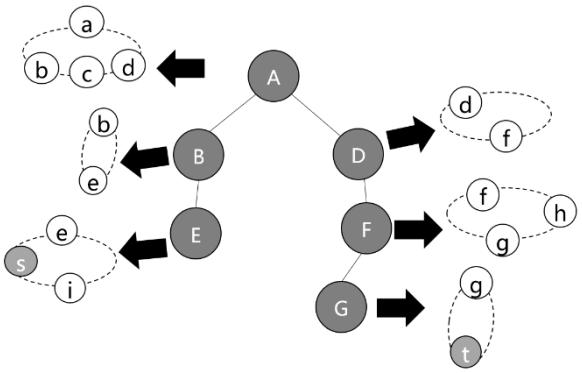


Fig.2 Diagram of parallel calculating the max flow on GraphChi

Given the vertex pair (s,t) in the original graph, according to algorithm 1, s and t are respectively in the subgraph IG_u and IG_v, and they correspond to vertex U and V in the overlay graph respectively. If s and t are in the same subgraph, the max flow between s and t can be directly calculated. If s and t are not in the same subgraph, determine the unique path between U and V in the overlay graph, and find the corresponding subgraph based on the path. In the process of building an overlay graph, U corresponds to the cut vertex u in the original graph and V corresponds to the cut vertex v in the original graph, the subgraphs on the path was submitted to the GraphChi to calculate the max flow between s and u, between the cut vertex in each subgraph and between v and t in parallel. The minimum value of the max flow of each subgraph is taken as the max flow of the original graph.

4. Experiment

In order to evaluate the max flow for large-scale graph calculation based on the GraphChi, this experiment was carried out with i5-4200@1.6GHz CPUs and 8GB memory, the operating system is Windows10. We use 5 real-world graphs in United States Road Networks (TIGER/Line), among them the number of vertices in these graphs ranges from 264346 to 1207945.

According to the above experimental environment and data sets, experiment the execution time of the max flow calculation for different algorithms. During the experiment, 50 sets of vertex pairs (s,t) are randomly selected on each graph to calculate the max flow. For the algorithm to compute the max flow in parallel on the GraphChi, because the overlay graph and subgraph on each dataset only need be generated once, the total time for the 50 sets of (s,t) to generate the overlay graph and subgraph on a data set, as well as the time to find and locate the subgraph and determine the path in the overlay graph, and the time for parallel computing the max flow of each subgraph on the GraphChi, the average of the total time is taken as the average execution time of the algorithm on each data set. The traditional serial algorithm takes the average time of 50 groups for calculate the max flow as the

execution time of the algorithm on each data set. The experimental results of the average execution time of the two algorithms on 5 data sets are shown in table 1. The second column X represents the average execution time of the graph max flow acceleration algorithm based on the GraphChi, the unit is ms; The third column Y represents the average execution time of the max flow calculated by the traditional serial algorithm on the same data set, the unit is ms.

Table 1 Comparison of execution time of different algorithms

Vertex number	X/ms	Y/ms
264346	24936.48	74809.14
321270	18112.06	69361.08
435666	54478.60	178770.56
1070376	60661.48	262532.20
1207945	68526.70	344944.52

5. Conclusion

Considering the problem of high computational complexity of the max flow algorithm, an acceleration algorithm for constructing the overlay graph by using the cut vertex of original graphs and parallel computing of multi-subgraphs on the GraphChi is proposed. The experimental results show that the parallel computing max flow algorithm on the GraphChi can effectively shorten the calculation time of the max flow and improve the calculation speed of the max flow compared with the traditional serial algorithm.

References

[1] Liang C, Yu F R, Zhang X. Informationcentric network function virtualization over 5g mobile wireless networks[J]. IEEE Network, 2015, 29(3):68-74.
[2] Kosut O. Max-flow min-cut for power system security index computation[C]. 2014 IEEE 8th Sensor Array and Multichannel Signal Processing Workshop (SAM). IEEE, 2014.
[3] Yin T L, Rao S, Srivastava N. A new approach to computing maximum flows using electrical flows[C]. ACM Symposium on Theory of Computing. ACM, 2013:755-764.
[4] Yin T L, Rao S, Srivastava N. A new appro- ach to computing maximum flows using electrical flows[C]. ACM Symposium on Theory of Computing. ACM, 2013:755-764.
[5] Scheuermann B, Rosenhahn B. Slimcuts: Graphcuts for high resolution images using graph reduction [C]. International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition, 2011:219–232.
[6] BENCZUR A A, KARGER D R. Randomized Approximation Schemes for Cuts and Flows in Capacitated Graphs[J]. Siam Journ- al on Computing, 2015, 44(2): 290–319.
[7] Wei W , Liu Y , Zhang R. SPLMax: Exploiting the Simple Path Introduced Locality for Maximum Flow Acceleration[J]. IEEE Communications Letters, 2018, PP(99):1-1.
[8] Ramesh V, Nagarajan S, Jung J J, et al. Max-flow min-cut algorithm with application to road networks[J]. Concurrency & Computation Practice & Experience, 2017, 29(2):12-22.
[9] Kyrola A, Blelloch G, Guestrin C. GraphChi: large-scale graph computation on just a PC[C]. Usenix Conference on Operating Systems Design and Implementation. 2012:31-46.
[10] ROMAN T, INSTITUTE J S. Limits To Parallel Computation: P-Completeness Theory[J]. IEEE Concurrency, 1999, 7(1): 79-79.
[11] Johnson D S. A Catalog of Complexity Classes[M]. Handbook of theoretical computer science (vol. A). MIT Press, 1991.
[12] United States Road Networks (TIGER/Line) (http://www.dis.uniroma1.it/challenge9/data/tiger/)