Journal of Industrial Information Technology and Application Vol.2 No.2

ISSN(Online) : 2586-0852

Partitioning Large Graphs using Particle Swarm Optimization with Breadth First Search

Srimanth Gadde [1], Robert C. Green II [2,*], William Acosta [3], Vijay Devabhaktuni [1]

[1] University of Toledo, Toledo, OH, USA
[2] Bowling Green State University, Bowling Green, OH, USA
[3] DISH Network, Englewood, CO, USA

Abstract. Processing large graph datasets represents an increasingly important area in computing research and applications. The size of many graph datasets has increased beyond the processing capacity of a single computing node, necessitating distributed approaches. Processing over a distributed system of nodes leads to an inter-partition communication cost problem, negatively affecting the system performance. Previously proposed graph partitioning approaches minimize the inter-partition communication on large graphs, yet require high computation overhead. To address this problem, a robust algorithm combining Particle Swarm Optimization (PSO) with Breadth First Search (BFS) is implemented, which does not require high computation overhead, reduces the inter-partition communication, and maximizes intra-partition communication by achieving an appropriate balance partition. In the first phase (PSO Selection), a core set of vertices are selected and assigned to the computing nodes. In the second phase, the graph is partitioned by placing the remaining nodes through the use of BFS. Experimental results show graph intra-partition performance for canonical graphs representative of real-world data is improved up to 50% in case of Powerlaw graph, up to 33% in case of Random near K-regular graph (with low degree), and up to 16% in case of Random near K-regular graph (with high degree). (Abstract)

Keywords; graph partitioning; particle swarm optimization; breadth first search; inter-partition communication

[*] Corresponding author: greenr@bgsu.edu

Received: 2017.9.28; Accepted: 2017.12.5; Published: 2018.6.30

1. Introduction

Large graph dataset processing has been extensively used for numerous applications such as social networking analysis, data mining, and web graphs [1, 2]. The graphs processed in these applications are becoming increasingly large, requiring graph processing of millions or even billions of vertices. However, as the size of these datasets and their computational demands continue to increase, there is a requirement to distribute the computation beyond a single computer (or CPU). This requirement has given rise to a set of distributed graph processing systems such as Pregel [3], Trinity [4], Haloop [5], Priter [6], Piccolo [7], Spark [8], Giraph [9], and others. These frameworks are designed to process large graph datasets on the distributed system. By employing these frameworks, various graph-based algorithms are computed to process large graph datasets on a distributed system. When these datasets are distributed randomly without partitioning, it increases the vertex-to-vertex communication between the computing nodes. This eventually leads to an inter-partition communication problem that negatively affects system performance [10]. Many graph algorithms are iterative and require significant communication between the vertices, thus making them sensitive to data/vertex placement in the distributed system. The tasks involved in these frameworks for processing the graph datasets generally include iterative mode operations where all the vertices/edges are accessed. Examples of these tasks are PageRank (PR) [11], Single-Source Shortest Path algorithm (SSSP) [12], etc.

This work presents a new graph partitioning algorithm based on previous work [13] which does not require high computation overhead and improves system performance by reducing inter-partition communication through the use of population-based metaheuristic algorithms like particle swarm optimization (PSO) that have rarely been used in the literature [14-16]. The algorithm partitions the graph in two intuitive stages: Random Selection method and PSO with BFS partitioning. In the first stage, the set of vertices are selected randomly from the graph, based on the limit value provided, and these vertices are assigned randomly on the computing nodes. In next stage, these vertices are given as input and BFS is applied on the selected vertices in order to partition the graph. To ensure the appropriate balance partition, the count of vertices on each node is checked at each step with a provided cutoff value. This process partitions the graph entirely without requiring a significant amount of computation overhead. The proposed algorithm is applied to three types of graph topologies including the Power-law graph, Random near K-regular graph (with low degree), and Random near K-regular graph (with high degree). All three graph topologies vary in shape, connections and structure.

2. Methods

A. Particle Swarm Optimization

PSO was introduced by Eberhart and Kennedy in 1995, is a population-based stochastic optimization technique for determining continuous and discrete optimization problems. It is influenced by the social behavior of bird flocking and fish schooling, and is well-known for solving optimization problems. In PSO, agents called "particles" travel in the search space to find the optimal solution. The position of each particle located inside the search space indicates the potential solution to the optimization problem. In PSO, the potential solution can be characterized with the particle's past behavior depending on its position, velocity, and best recorded fitness. Each particle continually updates its velocity and position dependent upon the particle population behavior in order to derive the best solution to the optimization problem. The formulas for updating the position and velocity of the PSO algorithm are shown in (1) - (4) where R, V, and M are the arrays of particles, velocities, and fitness values, *p* is the current particle, *d* is the current dimension, *r* is a random number, G_{best} is the global best, P_{best} is the personal best, c_1 is the cognitive constant, c_2 is the social constant, and Ψ is the constriction factor.

$$R_{p,d} = R_{p,d} + V_{p,d} \quad (1)$$

$$V_{p,d} = \Psi\big(V_{p,d} + rc_1\big(P_{best,p,d} - R_{p,d}\big) + rc_2(G_{best,p,d} - R_{p,d})\big) \quad (2)$$

$$\Psi = \frac{2.0}{\left|2.0-\Phi-(\sqrt{\Phi^2-4\Phi}\right|} \quad (3)$$

$$\Phi = c_1 + c_2 \quad (4)$$

B. Proposed Method

The proposed method uses the PSO algorithm [17] to partition initial vertices between nodes in a distributed system. Once the PSO algorithm has assigned the initial nodes, BFS is used to partition any remaining nodes. This is fundamentally the same process as used in [13], with the random selection of nodes being replaced by the PSO algorithm. PSO is initialized using values of x_{min}= 0, x_{max}= 32, v_{min} = 0, v_{max} = 20, N_p = 50, $C_1 = C_2$ = 6.05, and N_t=100. As vertices are being placed on computational system of 32 nodes, each particle has 32 dimensions, each containing an integer which selects a single, root vertex that is to be assigned to that node. It is this seed node which is used to run the BFS algorithm. The fitness function used minimizes inter-partition communication and is shown in (5) where T_{sent} is the total number of messages sent from all nodes and T_{rx} is the total messages received within all nodes.

$$M = \frac{T_{sent}}{T_{rx}} \quad (5)$$

For experimentation, Giraph is chosen as the graph processing framework since it combines the best features of Pregel and Bulk Synchronous Parallel model. The experiments in this study were performed on a 32-node distributed network for processing the graph datasets using Giraph framework. Table I shows the intra-partition and inter-partition communication before partitioning the graph dataset for various graphs. The presented graph partitioning algorithm was developed in Java jdk1.7.0_03 Enterprise Edition. It was implemented on an Intel Core i5 with a 2.30 GHz processor and 4.00 GB of RAM, running on the Microsoft Windows 7 Service Pack 1. For testing the algorithm, three types of graph topologies were used: Powerlaw graph (this dataset is router topology which holds link directions corresponding to the trace route directions), a randomly generated Random near K-regular graph (with low degree), and a randomly generated Random near K-regular graph (with high degree). Specifically the experiments were performed on a Powerlaw graph (with 190,914 vertices, 1,215,102 edges), Random near K-regular graph with high degree (with 100,000 vertices, 2,500,000 edges) and Random near K-regular graph with low degree (with 100,000 vertices, 400,000 edges). All trials are run 10 times.

TABLE I. INTRA-PARTITION AND INTER-PARTITION COMMUNICATION IN % USING SSSP AND PR BEFORE PARTITIONING

	SSSP		**PR**	
Graph Topology	***Intra-partition communication***	***Inter-partition communication***	***Intra-partition communication***	***Inter-partition communication***
Powerlaw	3.7	96.3	3.7	96.3
Random near K-regular (low degree)	4.1	95.9	3.9	96.1
Random near-K-regular (high degree)	3.9	96.1	4.0	96.0

3. Results

Table II shows the results of three different graph topologies: Powerlaw graph, Random near K-regular graph (with low degree) and Random near K-regular graph (with high degree). Each table contains the following results performed for 10 trials and each trial is executed for 100 iterations: the average improved intra-partition communication and the average reduced inter-partition communication.

In Table II, by using the proposed PSO-based algorithm on the Powerlaw graph, the average intra-partition communication is improved from 3.7% to 48.9% in the case of SSSP and from 3.7% to 49.9% in the case of PR. The average inter-partition communication is reduced from 95.9% to 51.1% in the case of SSSP and from 96.1% to 50.1% in case of PR. For the Random near K-regular graph (with high degree), the average intra-partition communication is improved from 3.9% to 16% in the case of SSSP and from 3.9% to 15.7% in the case of PR. The average inter-partition communication is reduced from 96.1% to 84.0% in the case of SSSP and from 96.0% to 84% in case of PR. When using the proposed algorithm with the high degree graph, the average intra-partition communication is improved from 3.9% to 16% in the case of SSSP and from 3.9% to 15.7% in the case of PR. The average inter-partition communication is reduced from 96.1% to 84.0% in the case of SSSP and from 96.0% to 84% in case of PR.

TABLE II. AVERAGE AND STANDARD DEVIATION OF IMPROVED INTRA-PARTITION AND MINIMIZED INTER-PARTITION COMMUNICATION

	SSSP		**PR**	
Graph Topology	***Intra-partition communication***	***Inter-partition communication***	***Intra-partition communication***	***Inter-partition communication***
Powerlaw	48.9 ± 0.04	51.1 ± 0.05	49.9 ± 0.04	50.1 ± 0.04
Random near K-regular (low degree)	33.1 ± 0.04	66.9 ± 0.04	34.2 ± 0.03	65.8 ± 0.03
Random near-K-regular (high degree)	16.0 ± 0.01	84.0 ± 0.02	15.7 ± 0.01	84.3 ± 0.01

The proposed PSO graph partitioning algorithm gives good results when compared to the stream partitioning methods in [18]. Their PR computations were improved up to 18%-39% whereas we improved up to 50%. A set of vertices are randomly selected (say 10% of total vertices) and the whole graph is partitioned by applying BFS on each selected vertex within the PSO. This process is repeated for several iterations using PSO until the algorithm stopping criterion is met. The presented algorithm can minimize the inter-partition communication as well as improve the intra-partition communication for all the three graph topologies presented by maintaining a good balance partition. From these results it can be concluded that our proposed algorithm is efficient and minimizes the inter-partition communication by maximizing the intra-partition communication.

4. Conclusion

The proposed PSO graph partitioning algorithm gives good results when compared to the stream partitioning methods in [18]. Their PR computations were improved up to 18%-39% whereas we improved up to 50%. A set of vertices are randomly selected (say 10% of total vertices) and the whole graph is partitioned by applying BFS on each selected vertex within the PSO. This process is repeated for several iterations using PSO until the algorithm stopping criterion is met. The presented algorithm can minimize the inter-partition communication as well as improve the intra-partition communication for all the three graph topologies presented by maintaining a good balance partition. From these results it can be concluded that our proposed algorithm is efficient and minimizes the inter-partition communication by maximizing the intra-partition communication.

In this paper, we focused on PSO with BFS using single objective function i.e. maximizing the intra-partition communication. In the future, we will focus on PSO with a multi-objective function where the objective functions include maximizing the intra-partition communication and efficiently perform balance partitioning on each computing node to further maximize the intra-partition communication.

References

[1] U. Kang, E. Charalampos, E. Tsourakakis, C. Faloutsos, “Pegasus: A peta-scale graph mining system implementation and observations”, in IEEE Ninth International Conference on Data Mining, Miami, Florida, pp. 229–238, 2009.

[2] U. Kang, C. Tsourakakis, A. Appel, C. Faloutsos, J. Leskovec, HADI: Fast diameter estimation and mining in massive graphs with Hadoop, Technical Report, Machine Learning Department, School of Computer Science, Carnegie Mellon University, 2008.

[3] G. Malewicz, M. Austern, A. Bik, J. Dehnert, I. Horn, N. Leiser, G. Czajkowski, Pregel: a system for large-scale graph processing, in International Conference on Management of data, ACM, Indianapolis, Indiana, pp. 135–146, 2010.

[4] S. Bin, H. Wang, Y. Li, "The Trinity graph engine," Technical Report 161291, Microsoft Research, 2012.

[5] Y. Bu, B. Howe, M. Balazinska, and M. D. Ernst, "Haloop: Efficient iterative data processing on large clusters," PVLDB, pp. 285–296, 2010.

[6] Y. Zhang, Q. Gao, L. Gao, C. Wang, “Priter: a distributed framework for prioritized iterative computations,” in Proceedings of the 2nd ACM Symposium on Cloud Computing, ACM, New York, New York, pp. 13:1–13:14, 2011.

[7] R. Power, J. Li, “Piccolo: building fast, distributed programs with partitioned tables”, in Proceedings of the 9th USENIX conference on Operating systems design and implementation, USENIX Association, Berkeley, CA, USA, pp. 1–14, 2010.

[8] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica, “Spark: cluster computing with working sets”, in Proceedings of the 2nd USENIX conference on Hot topics in cloud computing, USENIX Association, Berkeley, California, 2010, pp. 10–10.

[9] Apache Incubator Giraph. http://incubator.apache.org/giraph/, 2013.

[10] S. Yang, X. Yan, B. Zong, A. Khan, “Towards effective partition management for large graphs”, in Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD’12, ACM, New York, New York, pp. 517–528, 2012.

[11] L. Page, S. Brin, R. Motwani, T. Winograd, “The PageRank Citation Ranking: Bringing Order to the Web”., Technical Report 1999–66, Stanford InfoLab, 1999.

[12] N. Deo, C.-Y. Pang, “Shortest-path algorithms: Taxonomy and annotation”, Networks 14 pp. 275–323, 2011.

[13] S. Gadde, W. Acosta, J. Ringenberg, R. C. G. II, V. Devabhaktuni, “Achieving Optimal Inter-Node Communication in Graph Partitioning Using Random Selection and Breadth-First Search,” International Journal of Parallel Programming, pp. 772-800, 2016.

[14] J. Kim, I. Hwang, Y.-H. Kim, B.-R. Moon, “Genetic approaches for graph partitioning: a survey”, in Proceedings of the 13th annual conference on Genetic and evolutionary computation, GECCO ’11, ACM, New York, New York, pp. 473–480, 2011.

[15] P. Sanders, C. Schulz, “High Quality Graph Partitioning”, in 10th DIMACS implementation Challenge-Graph Partitioning and Graph Clustering, pp. 1–18.

[16] D. Datta, J. R. Figueira, “Graph partitioning by multi-objective real-valued metaheuristics: A comparative study”, Applied Soft Computing, pp. 3976–3987, 2011.

[17] J. Kennedy and R. Eberhart, "Particle swarm optimization," Neural Networks, 1995. Proceedings., IEEE International Conference on, Perth, WA, 1995, pp. 1942-1948 vol.4

[18] Guo, , et al., "g2: A graph processing system for diagnosing distributed sytems," Proceedings of the 2011USENIX annual technical conference, USENIXATC (2011).