International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 11 | Nov 2019 www.irjet.net p-ISSN: 2395-0072

Comparison for Max-Flow Min-cut Algorithms for Optimal Assignment Problem

Saurabh Purkar[1], Rushikesh Borse[2], Shubham Dodmise[3], Akash Dhayatadak[4], Snehkumar Shahani[5], Reena Kharat[6]

[1,2,3,4]Student, Dept of Computer Engineering, Pimpri Chinchwad College of Engineering, Maharashtra, India
[5]Professor, Dept of Computer Engineering, Pimpri Chinchwad College of Engineering, Maharashtra, India
[6]Senior Software Engineer, Persistent Systems, Maharashtra, India

---

Abstract - *Events like Smart India hackathon require individuals to travel to distant place that is "centers" for final project demonstration. For this there is need to reduce the travel time and distance so that no team/individual is at disadvantage in terms of travelling distance and time. It is beneficial if center allocation for such large-scale event is done by minimizing total distance travelled. This can be solved by formulating a graph and then using a class of algorithms called Max-flow Min-cut algorithms to identify the optimal allocation. If individuals get proper centers as per this algorithm it would further result in minimized travel cost. For this we have used a class of algorithms that fall into Max-flow Min cut algorithms. We have evaluated different algorithms in this group and recommended best algorithm that provides optimized allocation in shortest amount of time to teams to respective centers in an efficient manner.*

Key Words: **Hackathon, Center Allocation, Max-flow Min Cut, Minimized Cost, Optimized Allocation**

1. INTRODUCTION

In this paper, we attempt to address the problem of exam center allocation to individuals and teams using different Max-flow Min-cut algorithms. Max-flow Min-cut algorithms. The motivation to address this problem is to optimize the travelling cost and time that incurs to organizers and participant respectively. There are different algorithms which can be used to allocate nodal center. But each algorithm has a different cost associated with it so while deciding algorithm there are different factors that need to be considered. There are factors such as complexity of implementation, execution time are considered. There are some algorithms which give locally optimized solutions but to increase accuracy and efficiency of solution we consider different approaches and algorithms. This paper mainly includes algorithms like Hungarian algorithm [2][3][4][5], Ford Fulkerson algorithm [9], Edmond Karps[8] algorithm, Auction algorithm[10] and their comparative study with the help of program implementation. These different algorithms work best in different scenarios and must be selected according to requirement and type of data. Performance of the algorithm is also an important factor in selecting the algorithm. As a result, the outcome will be optimized allocation of nodal center according to distance.

1.1 Base Concepts

1. Matching:

For a given graph G = (V, E), a matching can be a subgraph of graph G i.e H such that H= (V*,E') where E' is a subset of E & is a set of edges that do not have common vertices i.e. every node has either zero or one degree.

2. Maximum matching:

A matching is called maximum matching if it is a matching that contains the maximum number of edges matching maximum number of nodes.

3. Minimum Weight matchings:

The matching which is performed to minimize overall weight of graph.

4. Bipartite matching:

A bipartite graph consists of 2 disjoint sets of vertices V1 & V2. Bipartite matching aims at matching vertices in V1 to vertices V2 on one-to-one basis.

1.2 Problem Statement

Consider a set of teams $t_1, t_2, t_3, t_4 \ldots\ldots t_n$, n $\in$ N. Consider a set of domains (group of problem statements that fall under the same group or organization) $D_1, D_2, D_3, \ldots\ldots\ldots D_m$, m belongs N Consider a set of nodal enters as $C_1, C_2, C_3, \ldots\ldots\ldots C_m$, m $\in\in$ N. To ensure that number of nodal centers are equal to domains we need to club/combine the small domains into one. Each domain is assigned a subset of set of teams. Consider m subsets of set of teams assigned to $D_1, D_2, D_3 \ldots\ldots\ldots D_m$. Let the subset be denoted by $T_1, T_2, T_3, \ldots\ldots\ldots T_m$ such that T_1 is allocated to D_1, T_2 is assigned to D_2, ..., Dm is assigned to D_m. These subsets should follow the given condition:

$$T_1 \cap T_2 \cap \ldots \cap T_m = \emptyset$$

This means if a team t_i is allocated to some domain D_j it cannot be assigned again to another D_k. Let distance from each team's home institute to each nodal centre be represented by w_{ij} where i = team number j = centre where

competition for domain D_j will be hosted. For example, w_{12} is the distance of team T_1 from nodal centre NC_2 Weightage of each node from team subset is calculated by using distance of set of teams T_1, T_2……. T_m from every nodal centre by using summation function, which is described as follows:

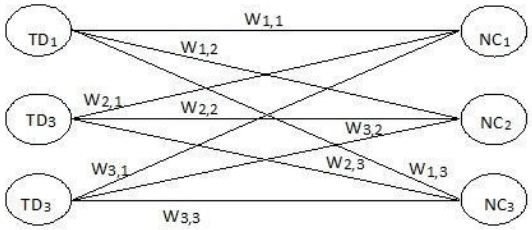

Fig.:1

This results in complete bipartite graph where T_1, T_2, T_3 represent the set of teams assigned to D_1, D_2, D_3 respectively. Each $W_{i,j}$ represents distance of T_i to C_j which is calculated using summation function defined earlier.

We have m subsets of team T_1, T_2……. Tm and m nodal centers as C_1, C_2……. Cm. The algorithm is expected to identify optimal assignments of these m subsets to m nodal centers uniquely. The cost associated with assigning a particular subset to nodal center is given by $W_{i,j}$ which represents cost of i^{th} subset to j^{th} nodal centers.

1.3 Mathematical model:

The cost matrix can be defined as adjacency matrix of m*m, where $W_{i,j}$ denotes cost of assignment of i^{th} subset to j^{th} nodal center.

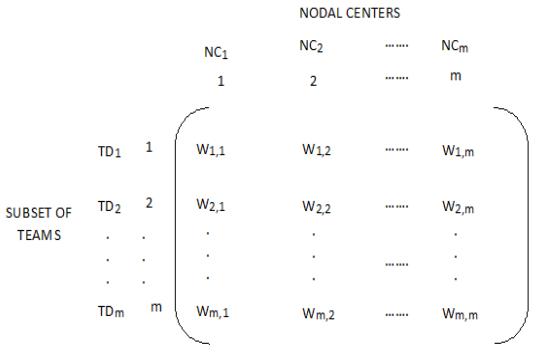

Fig.:2

An assignment is choosing m elements from above matrix such that no two elements lie in the same row/column. An optimal assignment is the one which will follow the above constraint and minimize the cost associated with.

2. ALGORITHMS

This section gives detailed analysis of various possible algorithms for solving the problem statement.

2.1 Hungarian Algorithm [2][3][4][5]

The Hungarian matching algorithm [2]is also called as the Kuhn-Munkkes [2] algorithm. The Hungarian algorithm solves the assignment problem in $O(n^3)$ time, where n is the number of vertices in one partition of bipartite graph. We use Hungarian algorithm to find minimum weights matching in bipartite graph which is represented using adjacency matrix. The graph is a complete bipartite graph. The algorithm operates on the concept that if a number is added or subtracted from all of the entries of any one row or column of a cost matrix, then an optimal assignment for the resulting cost matrix is also an optimal assignment for the original cost matrix. Hence, by using Hungarian algorithm we can solve assignment problem in $O(n^3)$ time which otherwise would need O(n!) when solved using brute force approach [2]. The transportation problem as in [6] deals with the cost of moving ships to desired location. There will be a certain number of ships to be moved from one location to another. We deal with the problem of choosing this quota so that the total cost of moving the ships is as small as possible. There is version of Hungarian algorithm described in [5] which works with changing cost or edges. Whenever there is a change in edge or cost this generates a new problem. But this dynamic version of Hungarian algorithm solves this problem efficiently by repairing the initial solution obtained before the cost changes thus saving the time to compute whole problem again.

2.2 Ford Fulkerson [9]

While assigning nodal center to participating teams we have to take care that unique nodal centers should get allocated to group of teams. This problem and structure can be mapped into bipartite graph. Bipartite graph is a type of graph in which all vertices can be split into two sets U and V such that every edge connects a vertex u ∈ U with a vertex v ∈ V, and there are no edges between vertices in the same group. U and V will be nodal center and group of teams respectively. While assigning group of teams uniquely to nodal center concept of bipartite matching helps us to do that. Bipartite matching is a concept in which it is take care that, we choose edges such that no two edges share same endpoint. In a maximum matching maximum number of edges are matched to each other if dynamically some edge is added into given edges it is no longer matching. There can be more than one maximum matchings for a given bipartite graph. To solve problem of bipartite matching there are different steps in first step we have to build flow network for converting problem into flow network problem we add sink and source at both ends. From source lines are drawn to all the vertex in U. And to connect sink all the vertex in V are connected by edges. Constant flow with constant capacity is assumed to

solve this problem. And in second step we have to find maximum flow. There are a number of techniques available to solve this maximum flow problem. By using Ford Fulkerson algorithm, we find the maximum flow and balance the network so that we can get unique relation between vertex from U to V.

Ford Fulkerson algorithm [9] is used to find the maximum flow. By using this algorithm, we can balance the network. For balancing the network, we have to solve the problem two sets of nodal center and group of teams is given input to algorithm in form of matrix. The output of this algorithm will be uniquely assigned set of nodes from U and V. While checking assignment we have to keep track of it using data structure like array. We have to check whether matching is possible for given matrix. So first we have to check whether particular element from U and V are connected or not. After checking the possibility of matching, a node is added in set of assigned nodes. In assigned node it is set of nodes i.e. unique assignment of nodal centers to group of teams.

Solution generated using Ford Fulkerson algorithm is locally optimal solution as one nodal center gets assigned to another nodal center it is considered as final assignment. But there can be group of teams which are not visited and have a minimum sum of distance. In such case there is no optimal allocation. So, this method is not much useful for efficient solution [9].

2.3 Edmonds-Karp Algorithm [8]

The Edmonds-Karp Algorithm [8] is the advanced version of the Ford-Fulkerson algorithm. Just like Ford-Fulkerson, Edmonds-Karp is also an algorithm that deals with the max-flow min-cut problem. Edmonds-Karp provides a full specification. This means that it specifies that breadth-first search should be used to find the shortest paths during the intermediate stages of the program. The search order of augmenting paths is well defined along with residual graphs, this are the two important concepts to understand when finding the max flow of a network.

Augmenting paths are the path defined from the source to the sink that can currently take more flow. In this process, flow is monotonically increased. So, there are times when a path from the source to the sink can take on more flow, and that is known as augmenting path.

COMPLEXITY: Edmonds-Karp algorithm made few advancements on the Ford-Fulkerson algorithm. Ford-Fulkerson works on principle that the flow increases by at least 1 in every iteration. Each iteration takes O(|E|) time. So, all Ford-Fulkerson can promise is that the maximum flow is found in O(|E| .f*) , where f* is the maximum flow itself.

Edmonds-Karp Algorithm removes the dependency on maximum flow for complexity, making it much better for graphs that have a large maximum flow. Edmonds-Karp improves the runtime of Ford-Fulkerson, which is O(|E| .f*) , to O(|V| .|E|^2) . This improvement is important because it makes the runtime of Edmonds-Karp independent of the maximum flow of the network.

2.4 Auction Algorithm [10]

The auction algorithm [10] is a parallel relaxation method for solving the classical assignment problem. The term "auction algorithm" refers to several variations of a combinatorial optimization algorithm which speculates assignment problems and network optimization problems with linear and nonlinear cost. An auction algorithm has been used in a path finding to determine the best distance on a set of distance offered from multiple locations. It is an iterative procedure, so the name "auction algorithm" is related to a path auction, where multiple nodes are compared to determine the best distance and hence giving the best optimal solution. The original form of the auction algorithm is an iterative method to find the optimal path and an assignment that maximizes the net benefit in a bipartite graph, the maximum weight matching problem (MWM).

The auction algorithm has excellent computational complexity and is reputed to be among the fastest for solving single commodity network optimization problems. In addition, the original version of this algorithm is known to possess a distributed nature particularly suitable for distributed systems, since its basic computational primitives (bidding and auctioning) are localized rather than relying on queries of global information. However, the original version that is intrinsically distributable has a pseudo-polynomial time complexity, which means that the running time depends on the input data.

Even if the total benefit with the auction algorithm is monotonically increasing with each iteration, in the Hungarian algorithm the total benefit strictly increases with each iteration. Hence, we get more accurate result using Hungarian algorithm.

2.5 Genetic Algorithm [11]

Genetic Algorithm is a heuristic search algorithm. It is based on the ideas of natural selection and genetics. These are intelligent exploitation of random search provided with historical data to direct the search into the region of better performance in solution space. It is commonly used to generate high-quality solutions for optimization problems and search problems. It simulates the process of natural selection which means those species who can adapt to changes in their environment are able to survive and reproduce and go to next generation. In simple words, they simulate “survival of the fittest” among individual of consecutive generation for solving a problem. Each generation consists of a population of individuals and an individual represents a point in search space and possible solutions. An individual is represented as a string of character/integer/float/bits. This string is analogous to the

Chromosome. Genetic algorithms are based on an analogy with genetic structure and behavior of chromosomes of the population. Following is the foundation of GAs based on this analogy –

1. An Individual in population compete for resources and mate
2. The individuals who are successful (fittest) then mate to create more offspring than others
3. Genes from "fittest" parent transform throughout the generation, that is sometimes parents create offspring which is better than either parent.
4. As a result, each successive generation is more suited for their environment.

A genetic algorithm is applied to find a maximum flow from the source to sink in a weighted directed graph, where the weight associated with each edge represents its flow capacity in one direction. The maximum flow problem is one of several well-known basic problems for combinatorial optimization in weighted directed graphs. Because of its importance in many areas of applications such as computer science, engineering and operations research, the maximum flow problem has been extensively studied by many researchers using a variety of methods (Tarjan 1983, McHugh 1990, Mazzoni 1991). The maximum flow problem appears to be more challenging in applying genetic algorithms than many other common graph problems because of its several unique characteristics. For example, a flow at each edge can be anywhere between zero and its flow capacity, rather than being a fixed distance. Also, the total inflow and outflow at each vertex must balance. The fitness function is defined to reflect two characteristics - balancing vertices and the saturation rate of the flow. Starting with a population of randomized solutions, better and better solutions are sought through the genetic algorithm. Optimal or near optimal solutions are determined with a reasonable number of iterations compared to other previous GA applications.

3. EXPERIMENTAL EVALUATION:

Python 3.0 was used as the programming and development environment. Ford Fulkerson Algorithm, Edmond Karp algorithm and Hungarian algorithm were developed using Python development environment. Then Smart India Hackathon dataset of 2019 and centers were allocated to the individual teams

4. RESULTS:

The algorithms were compared for time required for execution. Table 1 shows the time taken by Hungarian Algorithm and Ford Fulkerson algorithm.

Algorithm	Execution Time
Hungarian Algorithm	0.004000186920166 second
FordFulkerson Algorithm	0.002996683120728 second

5. CONCLUSION

It can experimentally be concluded that Hungarian algorithm takes least time and hence is best suited to provide the optimized solution for center allocation from the algorithms considered in the Literature Review. Hungarian algorithm provides globally optimized result in shortest time. The time complexity of Hungarian algorithm is O(n^3). Hence we have developed the tool for identifying the best center using Hungarian algorithm.

REFERENCES

1) Yonatan Aumaan, Yuval Rabani," An O(log k) approximate min-cut max-flow theorem and approximation algorithm" In Proceedings of 1998 Society for Industrial and Applied Mathematics.

2) Harold W. Kuhn, "The Hungarian Method for the assignment problem", Naval Research Logistics Quarterly, 2: 83–97, 1955. Kuhn's original publication.

3) Harold W. Kuhn, "Variants of the Hungarian method for assignment problems", Naval Research Logistics Quarterly, 3: 253–258, 1956.

4) Yan Zeng, Xuesong Wu, Jianwen Cao," The Research and Analysis of Hungarian Algorithm in the Structure Index Reduction for DAE", In Proceedings of 2012 11th International Symposium on Distributed Computing and Applications to Business, Engineering & Science.

5) G. Ayorkor Mills-Tettey, Anthony Stentz, M. Bernardine Dias," The Dynamic Hungarian Algorithm for the Assignment Problem with Changing, Costs", Robotics Institute Carnegie Mellon University Pittsburgh, Pennsylvania 15213.

6) J. Munkres, "Algorithms for the Assignment and Transportation Problems", Journal of the Society for Industrial and Applied Mathematics, 5(1):32–38, 1957 March.

7) Jonker, R.; Volgenant, A. (December 1987). "A shortest augmenting path algorithm for dense and sparse linear assignment problems.

8) Kalyan Kumar Mallick, Aminur Rahman Khan, Mollah Mesbahuddin Ahmed, Md. Shamsul Arefin, Md. Sharif Uddin." Modified EDMONDS-KARP

Algorithm to Solve Maximum Flow Problems" Open Journal of Applied Sciences, 2016, 6, 131-140.

9) Myint Than Kyi, Lin Lin Naing "Application of Ford-Fulkerson Algorithm to Maximum Flow in Water Distribution Pipeline Network" International Journal of Scientific and Research Publications, Volume 8, Issue 12, December 2018

10) D. P. Bertsekas" An Auction Algorithm for the Max-Flow Problem" Journal of Optimization Theory and Applications: Vol. 87, No. 1, Pp. 69-101, Oct 1995

11) Christopher R. Houck, Jeffrey A. Joines and Michael G. Kay," comparison of genetic algorithms, random restart and two-opt switching for solving large location-allocation problems" Department of Industrial Engineering, North Carolina State University, Raleigh, NC 27695-7906, U.S.A