Indian Institute of Information Technology Vadodara (Gandhinagar Campus)

Design Project Report-2021

on

DESIGN FOR BIPARTITE GRAPH

Submitted by

Gaini Om Prakash(201951061)

Ganugachintala Raja Sekhar(201951062)

Dharavath Rajeev Gandhi(201951186)

Jammula Jyothi Swaroop Kumar(201951189)

Under the supervision of

Dr. Manasi Kulkarni

Abstract-

In our daily life we come across some matching problems. As the companies are increasing rapidly So, the number of jobs and the people applying is also increasing. Here with the bipartite matching we can easily assign the jobs to employees who are eligible for it. And also with the weighted graph we can find the minimum cost to complete a task. We can solve many problems like marriage problem , assignment problem, minimum matching, maximum matching .Basically all these matching problems can be solved by using graphs. We decided to give a one solution for all matching problems. Here the user can give the input in the form of bipartite graph and the tool outputs the required matching. Moreover ,it can also detect if the graph is bipartite or not.

I. INTRODUCTION

The primary objective of this project is finding the matching algorithms for a given bipartite graph. A bipartite graph (or bigraph) is a graph whose vertices can be divided into two disjoint and independent sets U and V such that every edge connects a vertex in U to one in V. In this project, GUI is designed using Javaswing. It is used to design it for giving input in the form of graph. Major algorithms are implemented to solve some bipartite matching problems. The user is required to give the input bipartite graph in the form of edges and weights using the GUI and select the appropriate algorithm according to his need, the desired matching will be displayed .A set of items connected by edges, each item is called a vertex or node. A graph is a set of vertices and a binary relation between vertices and adjacency. Bipartite graph is a whose vertices can be divided into two disjoint and independent sets U and V such that every edge connects a vertex such that every edge connects a vertex in U to one in V. A matching is a set of edges, such that no two edges share the same vertex. A matching is said to be maximum matching if it has maximum number of edges. A matching M of graph G is said to be perfect if every vertex is connected to exactly one edge.

II. LITERATURE SURVEY

In the process of developing this project various researches were come to light by different researchers. However , they aimed for different applications. Here some of the details of those papers.

Authors of [1]. It present an organized social graphical view on resource allocation and the extended to multi-objective

resource of wireless networks. It Provided a systematic approach for multi-objective optimization and then applied bipartite matching to enhance the design of wireless networks. Author of [2] Introduced a new representation of documents and co-occurrence concepts in the documents. They introduced a novel document clustering approach that overcome those problems by combining a semantic-based bipartite graph representation.

III. THE PRESENT INVESTIGATION

The present tool was designed using eclipse Ide which we used java as backend and JavaSwing Framework as frontend. This tool is a simple and lite desktop application so that it becomes user friendly. User can add vertices using add vertices button and can join the vertices using edges by entering vertices in text field . For weighted matching user must join edges by entering their weightage in weightage text field.

Maximum Matching :

In graph theory a maximum matching is a matching of maximum number of edges such that no two edges share any end points. There are many real world problems that can be formed as bipartite matching example if there are M job applicants and N jobs. Each applicant is applied for one or more number of jobs but job applicant can be appointed for only one job.to assign the jobs to applicants such that many applicants as possible get jobs.

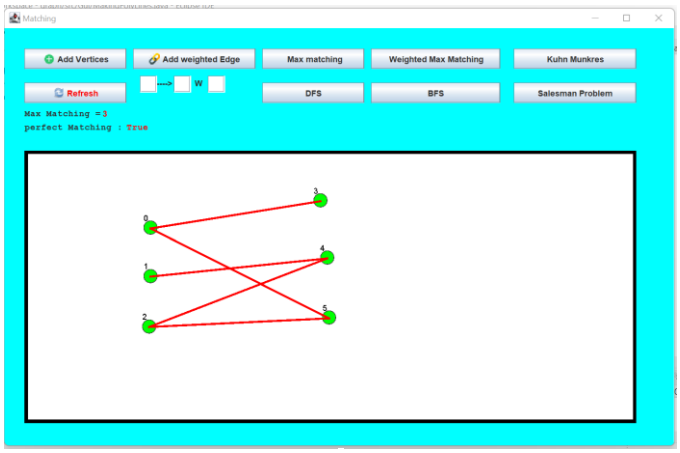

Fig-1.Maximum Matching

Assignment Problem:

The Hungarian maximum matching algorithm, also called the Kuhn-Munkres algorithm, that can be used to find maximum and minimum weight matchings in bipartite graphs, which is sometimes called the assignment problem. A bipartite graph can easily be represented by an adjacency matrix, where the weights of edges are the entries.

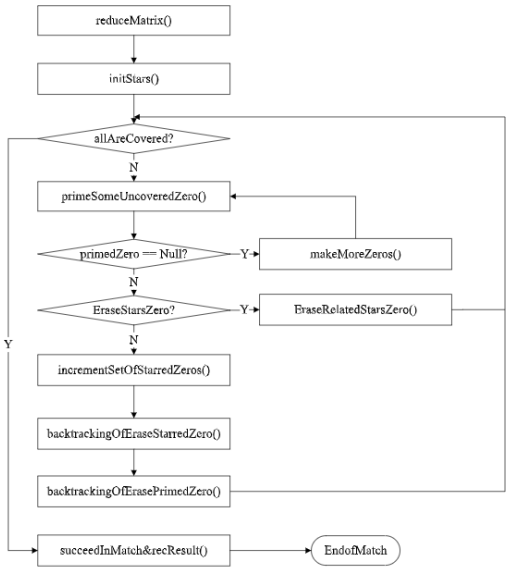

Fig-2.Flow chart of Hungarian Algorithm

The Kuhn-Munkres (KM) algorithm is a combinatorial optimization algorithm that is widely utilized to solve many real-world problems. Assignment problem is if there are number of agents and number of tasks. Any agent can be assigned to perform any task, incurring some cost and the cost may vary. We have to assign the agents to tasks in such a way that total cost of the assignment is minimized. For this mostly faced common problem we have used Kuhn-munkres algorithm to solve.

Minimum weighted matching.

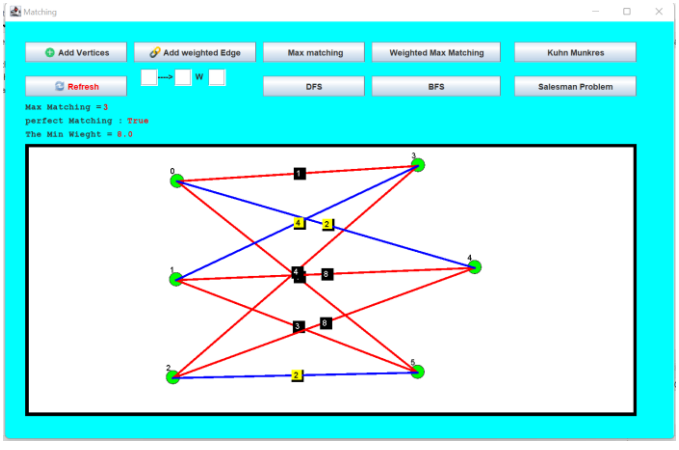

Fig-3.Minimum Weighted Matching

For maximum weighted matching.

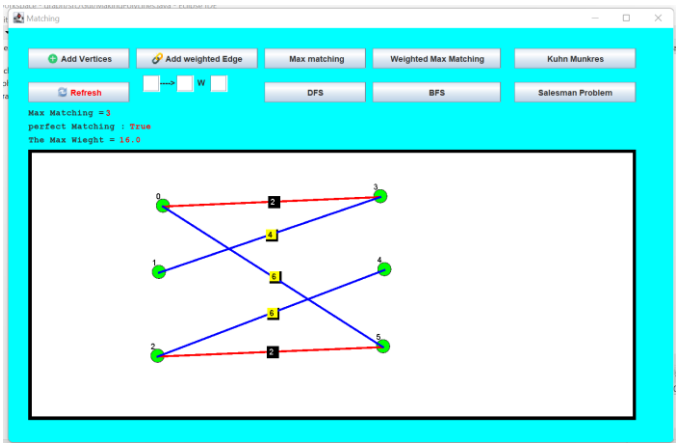

Fig-4.Maximum Weighted Matching

Apart from implementing matching algorithms, we aim to make this tool so that other graph algorithms too can be implemented and visualized.

Travelling Salesman Problem:

Travelling salesman problem is to travel in a shortest possible route that visits each city exactly once and returns to the original city. given set of cities and the distance between each pair of cities. terms of sub-problems.

Consider vertex '0' as the starting and ending point. Since the route is cyclic, we consider vertex '0' as a starting point and we will generate all possible permutations of cities which are (n-1)!.Find the cost of each permutation and keep track of the minimum cost permutation and return the permutation with minimum cost. Here we use a dynamic approach to calculate the cost function Cost(). Using recursive calls, we calculate the cost function for each subset of the original problem. To calculate the cost(i) using Dynamic Programming, we need to have some recursive relation in terms of sub problem.

Breadth First Search:

Breadth first search is a graph traversal algorithm that starts traversing the graph from root node and explores all the neighbouring nodes. Then, it selects the nearest node and explore all the unexplored nodes. The algorithm follows the same process for each of the nearest node until it finds the goal.

BFS can be implemented by using following steps:

1. Begin the search algorithm, by knowing the key which is to be searched. Once the key/element to be searched is decided the searching begins with the root (source) first.
2. Visit the contiguous unvisited vertex. Mark it as visited. Display it (if needed). If this is the required key, stop. Else, add it in a queue.
3. On the off chance that no neighboring vertex is discovered, expel the first vertex from the Queue.
4. Repeat step 2 and 3 until the queue is empty.

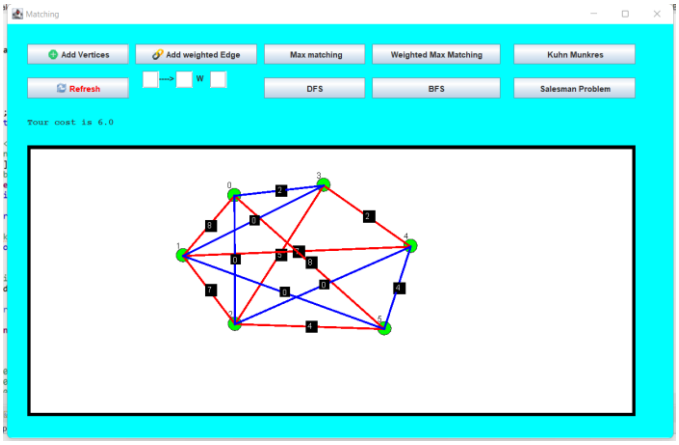

Fig-5.Travelling Salesman problem.

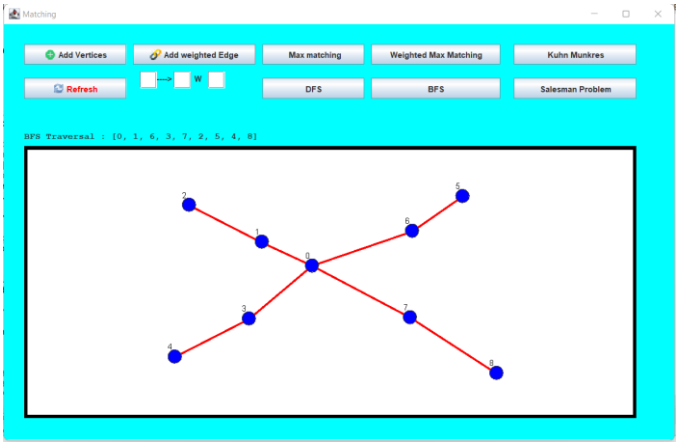

Fig-6.Breadth first Search.

Depth First Search:

Depth first search (DFS) algorithm starts with the initial node of the graph G, and then goes to deeper and deeper until we find the goal node or the node which has no children. The algorithm, then backtracks from the dead end towards the most recent node that is yet to be completely unexplored.

The data structure which is being used in DFS is stack. The process is similar to BFS algorithm. In DFS, the edges that leads to an unvisited node are called discovery edges while the edges that leads to an already visited node are called block edges.

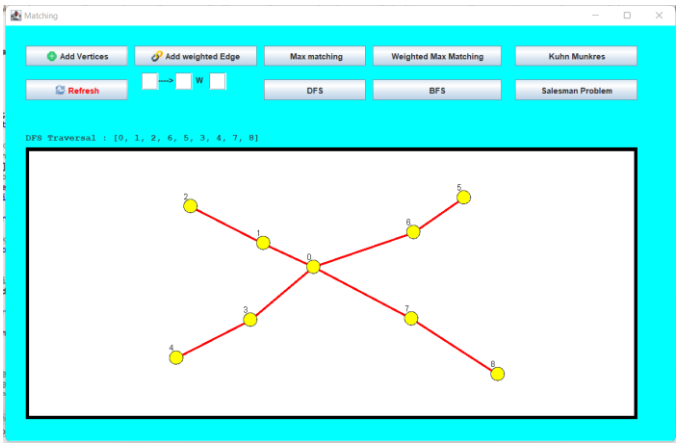

Fig-7.Depth first Search.

IV. Results and Discussions

We Included some of the major algorithms to solve major bipartite matching problems. The user is required to give the input bipartite graph in the form of edges and weights using the GUI and select the appropriate algorithm according to his need , the desired matching will be displayed.

V. Conclusions

The computation of assignment costs is a significant contributor to the overall running time of finding a solution to the assignment problem. However, our techniques show that by utilizing lower or higher costs and pruning rules, it is possible to terminate sooner while computing fewer expensive exact costs.

A combinatorial optimization algorithm used to find maximum and maximum weight matchings, which can be used to solve the assignment problem. Other than matching problems we had implemented Breadth First Search, Depth First Search and Travelling Salesman problem. For applications of DFS and BFS in relation to specific domains, such as searching for solutions in artificial intelligence or web-crawling, the graph to be traversed is often either too large to visit in its entirety or infinite . In such cases, search is only performed to a limited depth due to limited resources, such as memory or disk space, one typically does not use data structures to keep track of the set of all previously visited vertices. The Travelling Salesman Problem (TSP) is one of the best known NP-hard problems, which means that there is not guaranteed to get the optimal route and no exact algorithm to solve it in polynomial time.The method which would definitely obtain the optimal solution of TSP is the method of exhaustive enumeration and evaluation.

In future we can add all the matching algorithms So, that it becomes a one stop solution for matching problems.

VI. FUTURE SCOPE

This project is very much useful in our daily life. This can be further developed using mostly used script languages and which can make more attractive. Many challenges will be carried out to increase reliability and efficiency. This tool can also be developed using javascript language. The knowledge is ever expanding and so are the problems which the mankind strives to solve. In this spirit, it is hoped that current activity will lead further enhancements.In future we have plan to implement some other matching algorithms like Max-cardinality bipartite matching, Cardinality Matching of Convex Graph and if we find any better data structure or algorithm which might increase efficiency and speed ,we will update our tool so that changes can be added to the project in order to make it more fast and efficient and more convient to use .

Acknowledgment

We express our special gratitude to our honorable IIIT Vadodara staff who gave us an opportunity to work on this project. Being the students of IIIT Vadodara, it was our pleasure working on this project and it made us learn lot of new things. We would like to thank our project guide, Dr .Manasi Kulkarni, who helped us to complete this project and guided us continuously.

VII. References

[1] Socially Enabled Wireless Networks Resource Allocation via Bipartite Graph Matching. (n.d.).

[2] Yoo, I., Hu, X., & Song, I.-Y. (2006). Integration of Semantic-based Bipartite Graph Representation and Mutual Refinement Strategy for Biomedical Literature Clustering.

[3] Pankaj K Agarwal and R Sharathkumar. 2014. Approximation algorithms for bipartite matching with metric and geometric costs. In STOC. 555–564J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[4] Khuller, S., Mitchell, S. G., & Vazirani, V. V. (1994). On-line algorithms for weighted bipartite matching and stable marriages. *Theoretical Computer Science*, *127*(2), 255-267.R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[5] Zhu, H., Liu, D., Zhang, S., Zhu, Y., Teng, L., & Teng, S. (2016). Solving the many to many assignment problem by improving the Kuhn–Munkres algorithm with backtracking. *Theoretical Computer Science*, *618*, 30-41.

[6] Dwivedi, Varshika, Taruna Chauhan, Sanu Saxena, and Princie Agrawal. "Travelling salesman problem using genetic algorithm." *IJCA Proceedings on Development of Reliable* algorithm. *Information Systems, Techniques and Related Issues (DRISTI 2012)* 1 (2012): 25.

[7] (Halim, n.d.) Halim, D. S. (n.d.).*Travelling Salesman Problem* Retrieved from visualgo: https://visualgo.net/en/tsp?slide=1

[8] (Halim, n.d.) Halim, D. S. (n.d.). *Data structures and algorithms*. Retrieved from visualgo: https://visualgo.net/en