Introduction to Maximum Matching in Graphs

Omar Ghaleb*

***Abstract*— This work discussed the idea of maximum matching in graphs and the main algorithms used to obtain them in both Bipartite and General graphs.**

I. INTRODUCTION

Matching in graphs has been an important topic in computer science and has a lot of applications for solving optimization problems. It is used to solve the task assignment problem and stable matching in marriage problem or roommates matching. This can be generalized to solve any problem that can represented as a graph. In this paper, we will discuss the general matching idea in bipartite and the concept of using blossoms in general graphs.

II. PRELIMENARIES

A. Definition 1:

$G = (V, E)$ is an undirected graph where$|V| = n$ is the number of vertices and $|E| = m$ is the number of the edges.

B. Definition 2:

Given a graph $G = (V, E)$, M is a matching in$G$ if it is a subset of$E$ such that no two adjacent edges share a vertex.

C. Definition 3:

M is a maximum matching if and only if it has the maximum cardinality or the maximum possible number of edges.

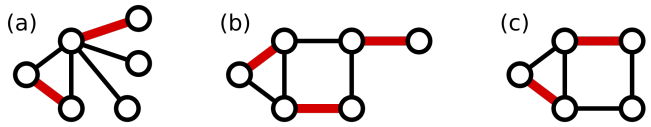

Fig. 1. Shows different maximum matchings

And as we can see from the following figure, maximum matching is not unique.

D. Definition 4:

A vertex is called free or exposed if it is not included in any matching.

E. Definition 5:

An Alternating path is a path whose edges are alternating between being in M and free edges.

*COMP 5703 Project Seminar

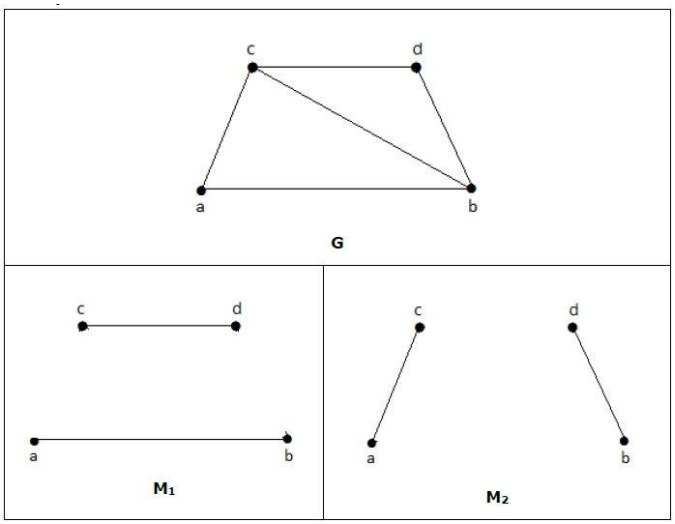

Fig. 2. An Example of two different maximum matching for the same graph

F. Definition 6:

An Augmenting path is and alternating path found starting from an exposed or free vertex and ends with and exposed one and can add an extra edge to the matching M.

Fig. 3. An Example of augmenting path

III. MAXIMUM MATCHING IN GRAPHS

A. Problem Statement

A graph $G = (V, E)$ is provided as an input. The goal is to find a matching that has the maximum possible cardinality, which is the maximum number of edges such that no two matched edges same the same vertex. We have four possible problems:

- Max cardinality matching in bipartite graphs.
- Max cardinality matching in general graphs.
- Max weighted matching in bipartite graphs.
- Max weighted matching in general graphs.

What we will discuss here are the first two types of the problem.

B. Approach for Solving Maximum Matching

To solve the maximum matching problem, we need an algorithm to find these maximum matching. The main idea is to find augmenting paths in the graph which will add an extra matching to the existing current matching.

- Theorem 1(Berges Matching):
  A matching M is maximum if and only if it has no augmenting paths.
- Proof:
  This comes from the idea of taking symmetric difference of two matchings M and the augmenting path P. we can see that an augmenting path is used to produce a bigger matching M than the current matching M.
  Now if we consider M a bigger matching than M in G. then we can see that the symmetric difference between them will produce a graph that will contain augmenting paths.

This theorem will be the basis of the proposed algorithms for finding the maximum matching in graphs.

IV. Maximum Cardinality in Bipartite Graphs

In this section, we will talk about the fastest known deterministic algorithm for finding the maximum matching in bipartite graphs. This algorithm is known as the Hopcroft-Karp Algorithm (1973). It runs in $O(|E|\sqrt{(|V|)})$.

The algorithm goes as follows:

- Maximum Matching (G, M)
- $M \leftarrow \phi$
- while ($\exists$ an augmenting path P in the maximal set of augmeting paths)
  $M \leftarrow M \bigoplus P$
- return M

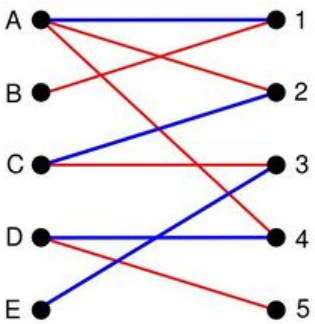

Fig. 4. An Example of bipartite graph

We can see that the algorithm runs in phases to find augmenting paths. First, it does breadth first search to divide the vertices into two sets. Then, starting from the free vertices in the graph, it starts looking for augmenting path of odd lengths using depth first search starting from lengths $(1, 3, 5, \ldots)$ until we don't have augmenting paths.

A. Analysis

The running time of the algorithm is in $O(|E|\sqrt{(|V|)})$. Each phase takes $O(m)$ since we must traverse each edge at most once while we are performing the breadth first search. Now we prove that it needs $\sqrt{n}$ phases. We need the following lemmas:

- $Lemma1$:
  $M' \bigoplus M$ has $|M'| - |M|$ vertex-disjoint augmenting paths
- $Lemma2$:
  The length of each augmenting path $\geq 2\sqrt{n} + 1$

Now let M be a matching after $\sqrt{n}$ phases. and let M be maximum matching. By $lemma1$ and $lemma2$, the number of augmenting paths in $M' \bigoplus M \leq \sqrt{n}/2$. Since each augmenting path increases the size of M by at least 1, then we have at most $\sqrt{n}/2$ more phases until we reach a maximum matching. Hence, we need only $O(\sqrt{n})$ phases to reach maximum matching.

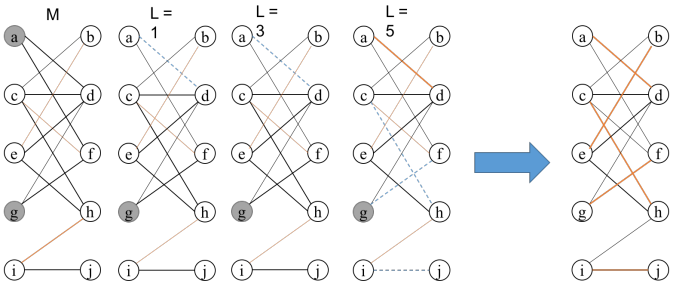

Fig. 5. An Example of Hopcroft and Karp algorithm

V. Maximum Cardinality in General Graphs

Just like the bipartite problem, what we want to find is the maximum matching in the graph. But here we encounter a problem which complicates the searching for such matching, this is due odd-length cycles in the graph. These cycles do not exist in the bipartite graphs. To solve this issue Edmonds (1965) presented an algorithm that runs in $O(n^4)$. After that a lot of improvements have been added to his algorithm with the following running time:

- $O(n^3)$ (Gabow, 1976):
- $O(nm)$ (Kameda and Munro, 1974)
- $O(n^{5/2})$ (Even and Kariv, 1975)
- $O(\sqrt{n}m)$ (Micali and Vazirani, 1980)

Micali and Vazirani managed to design an algorithm that runs in the same time as in the best time for bipartite graph. But only Edmonds main algorithm will be discussed. The other algorithms use the main concept Edmond has in his algorithm. To tackle his algorithm, we need to define something called Blossom.

A. Defenition 7:

Given $G = (V, E)$ and a matching M of G, a blossom B is a cycle in G consisting of $2k + 1$ edges of which exactly k belong to M, and where one of the vertices v of the cycle (the base) is such that there exists an alternating path of even length (the stem) from v to an exposed vertex w.

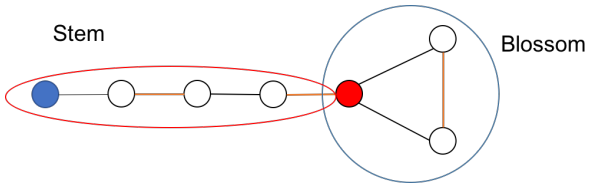

Fig. 6. An Example of a blossom and a stem

B. Edmonds Lemma:

Let G' and M' be obtained by contracting a blossom B in (G, M) to a single vertex. The matching M of G is maximum if and only if M' is maximum in G'.

The new algorithm for the general graph will apply the main concept of finding augmenting paths, but now it will also look for blossoms and contract them to a single vertex and look for augmenting paths after the contraction.

$EmdondsAlgorithm$:

- Maximum Matching (G, M)
- Find augmenting paths, and shrink blossoms if found
- $M \leftarrow \phi$
- while ($\exists$ an augmenting path P in the maximal set of augmeting paths)
  $M \leftarrow M \bigoplus P$
- return M

The implementation of finding augmenting paths here is different that bipartite graphs. Here we try to find also blossoms and contract them.

- We do a traversing for alternating path just like the bipartite graph
- Mark exposed vertex and at the even distance from it as (e)
- Mark vertices at odd distances as (o)
- We have a blossom if we have two even vertices adjacent. The running time for this algorithm is $O(n^2m)$

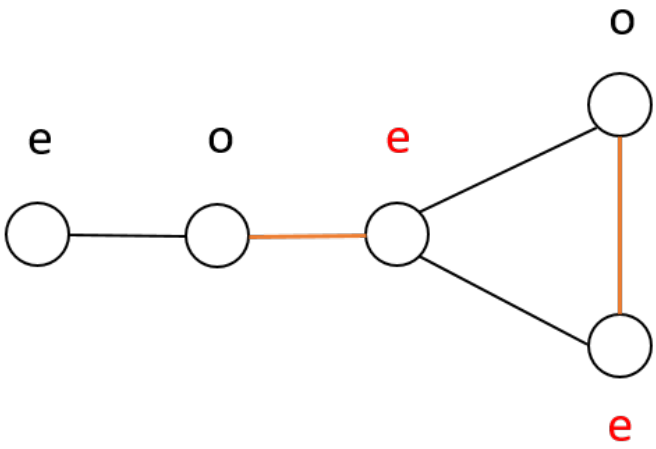

Fig. 7. An Example of How vertices are labeled

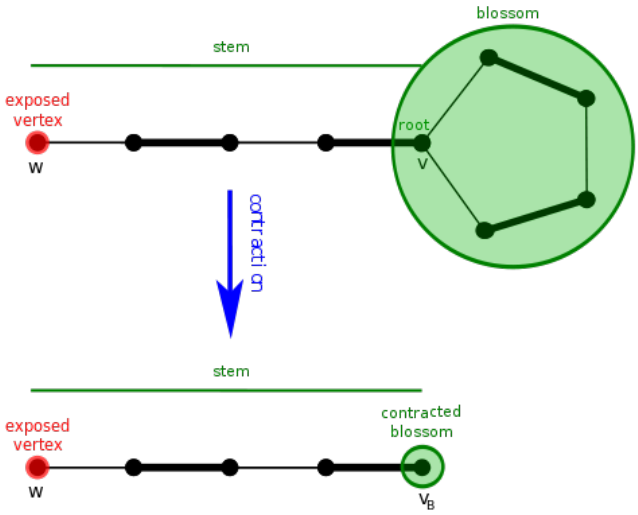

Fig. 8. An Example of contracting blossoms

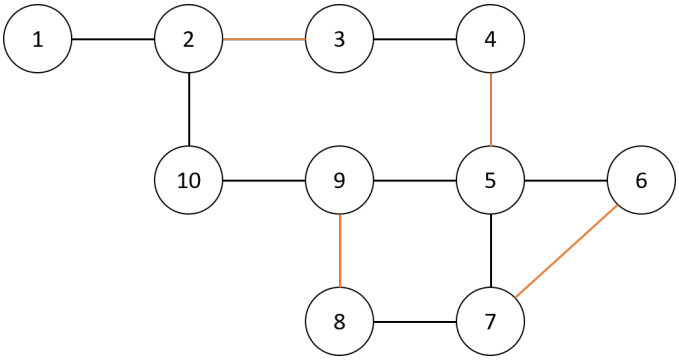

Fig. 9. An Example of How the graph looks like before the algorithm with $|M| = 4$

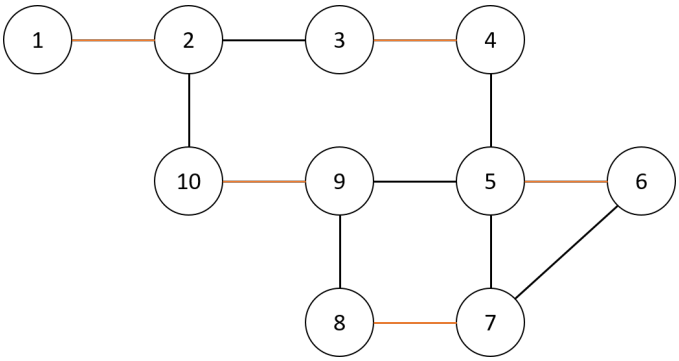

Fig. 10. The graph after running the algorithm with $|M| = 5$

C. Proof of Correctness:

The existing or non-existing of an augmenting path is preserved during the blossom shrinking since it has already the maximum matching inside of it. Assume there exist an augmenting path, then the algorithm will find this path and label both ends Even. Since for each matched edge has one vertex even and another odd, then we will have an edge where two end vertices are even. And the algorithm will traverse this edge before the algorithm terminates, so it will either find a blossom or an augmenting path.

References are important to the reader; therefore, each citation must be complete and correct. If at all possible, references should be commonly available publications.

REFERENCES

[1] Edmonds, Jack (1965). "Paths, trees, and flowers". Canad. J. Math. 17: 449467. doi:10.4153/CJM-1965-045-4.

[2] Norbert Blum, A New Approach to Maximum Matching in General Graphs.

[3] Galil Z.: Efficient Algorithms for Finding Maximum Matching in Graphs, Computing Surveys, 1986, 23–38.

[4] Tarjan, Theory of Algorithms,2002..

[5] Norbert Blum, Maximum Matching in General Graphs Without Explicit Consideration of Blossoms Revisited, 2016.

[6] Peterson and Loui, The General Maximum Matching Algorithm of Micali and Vazirani , 1988.

[7] Endika Bengoetxea, The graph matching problem, PhD Thesis, 2002.

[8] Dexter C. Kozen, The Design and Analysis of Algorithms, 1992.