CS-510 Algorithms Homework 3 - Bipartite Matching **Fall 2019**

With the following set of problems we will develop a combinatorial algorithm for matching in bipartite graphs. This is a classic combinatorial optimization problem with many applications. The algorithm is very simple and simply beautiful.

Matching in general graphs have many applications but matching in bipartite graphs also has applications in scheduling, Online ad Allocation, Computational Chemistry and Economics and Game Theory. After we learn basic definitions, we will model problems in these domains as matching problems in appropriately defined bipartite graphs.

1 Bipartite Graphs and their Characterization

Definition (Bipartite Graphs:)**.** *A graph $G = (V, E)$ is bipartite if V can be partitioned into two sets L and R, such that all edges in E have exactly one endpoint each in L and R.*

Often bipartite graphs are denoted with its (two) parts identified, i.e. $G = (L, R, E)$. A graph may not be specified as a bipartite graph when it is input. But one can easily find out if the graph is bipartite. If indeed the graph is bipartite, then one would employ different specialized algorithm for it, rather than using algorithms designed for general graphs (in many cases the problem is simpler on bipartite graphs).

Problem 1. *Given a graph $G = (V, E)$, give an $O(|V| + |E|)$ time algorithm to determine if G is bipartite.*

If the graph is bipartite, then its bipartition can be very easily found.

Problem 2. *Let $G = (V, E)$ be a bipartite graph, briefly describe a linear time algorithm to identify the parts of V. In other words express G as $G = (L, R, E)$.*

Before we develop a characterization for bipartite graphs, it will be helpful to do some warm up exercises.

Problem 3. *Let $G = (L, R, E)$ be a bipartite graph, what is the minimum and maximum possible degree of each vertex (give lower and upper bounds)*

Problem 4. *What is the maximum number of edges a bipartite graph $G = (L, R, E)$ can have?*

An analog of the Handshaking Lemma for bipartite graphs $G = (L, R, E)$ is stated as follows

$$\sum_{v \in L} deg(v) = \sum_{v \in R} deg(v)$$

Problem 5. *Prove the handshaking lemma for bipartite graphs.*

Definition (d-regular graph)**.** *A graph is called d-regular if every vertex of it has degree d.*

Problem 6. *Suppose $G = (L, R, E)$ is a d-regular bipartite graph. Prove that $|L| = |R|$*

Problem 7. *Prove that C_8 (a cycle graph on 8 vertices) is bipartite*

Problem 8. *Prove that C_5 is not bipartite*

Problem 9. *Prove that if G contains a C_5 as a subgraph, then G is not bipartite*

Problem 10. *Prove that if G contains a C_k (for k an odd integer) as a subgraph, then G is not bipartite*

Indeed, it turns out that odd cycles are the only reason for graph being not bipartite

Problem 11. *Prove that if G does not contain any odd cycle, then G is bipartite*

Together, the last two problems gives us a complete characterization of bipartite graphs, i.e. we get a necessary and sufficient condition to check bipartiteness.

Theorem 1. *A graph is bipartite if and only if it does not contain any odd cycle.*

2 Matching and Perfect Matching

Definition (Matching)**.** *For $G = (V, E)$, a matching M is a subset of edges ($M \subseteq E$), such that every vertex $v \in V$ is incident on at most one edge in M*

Problem 12. *Prove that the above definition of matching is equivalent to the following one*

Definition (Matching-Alternative-Definition)**.** *For $G = (V, E)$, a matching in G is a 1-regular subgraph of G.*

The matching problem becomes interesting only when some notion of optimization comes into play. To this end we define the size of matching

Definition. *The size of a matching M is the cardinality of edges in M*

Problem 13. *Let $G = (V, E)$ be a graph with $|V| = n, |E| = m$. Given some obvious lower bounds on the size of matching in M. Briefly describe graphs where these bounds are achieved (show the corresponding matchings)*

Problem 14. *Prove that every non-empty graph has a matching of size* 1

Problem 15. *Make a graph on $n \geq 100$ vertices with $n - 1$ edges that has no matching of size more than* 1

Definition (Perfect Matching)**.** *A matching M in $G = (V, E)$ is called a perfect matching if every vertex is* "matched"*, i.e. every vertex in V is incident to exactly one edge in M*

In other words, a perfect matching is a 1-regular spanning subgraph of the graph.

Problem 16. *What is the size of a perfect matching in a graph on n vertices? What if n is odd?*

Problem 17. *Make a small example of graph on* 6 *vertices to show that perfect matching does not have to be unique. In your example show all perfect matchings?*

The number of perfect matchings in a graph is a very interesting quantity and there is a deep theory and vast literature about it (mainly of interest to the enumerative combinatorics and algebraic graph theory community).

As a warm up ask your self the following questions.

Problem 18. *How many perfect matchings does a path graph on $n = 2k$ vertices have? How about a cycle graph on $n = 2k$ vertices?*

3 Matching in Bipartite Graphs

There are many optimization problems of matching in bipartite graphs depending on the application.

Perfect Matching Problem Given a bipartite graph $G = (L, R, E)$, find a perfect matching in G.

As we discussed above bipartite graphs do not necessarily have perfect matching, in that case one just seeks to find the largest matching.

Maximum Cardinality Matching or Maximum Matching Problem Given a bipartite graph $G = (L, R, E)$, find a matching in G of maximum size.

Clearly, the Maximum Matching problem is more general and perfect matching problem is its special case. This is what we will focus on in this problem set.

In some cases the bipartite graph would have weights on edges. These weights could represent the profit obtained by matching the two end points or could be the cost associated with the matching. Such graphs are denoted by $G = (L, R, E, w)$, where $w : E \to \mathbb{R}$ is a function on edges of G.

Depending on the interpretation of edge weights we seek to solve either of the following two problems. In these setting weight or cost of the matching is defined to be the sum of weights of the edges in the matching. Formally, weight or cost of a matching M is defined as:

$$w(M) = \sum_{e \in M} w(e).$$

Maximum Weight Perfect Matching Problem Given a bipartite graph $G = (L, R, E)$ with weights on edges. Find a perfect matching in G that has the largest weight.

Minimum Cost Perfect Matching Problem Given a bipartite graph $G = (L, R, E)$ with weights on edges. Find a perfect matching in G that has the least cost.

Problem 19. *In the last two problems, we require perfect matchings. While there is no guarantee that a perfect matching even exists. Briefly, describe how can we change the input graph so as a perfect matching in the modified graph correspond to a maximum weight or minimum cost matching in the original graph*

Remark. *Note that all these problems are very interesting and have a huge number of applications in general (non-bipartite) graphs too. There are beautiful algorithms and theory for this general problem. See for instance the Tuttes theorem for a necessary and sufficient condition for existence of perfect matching in general graphs. Edmond's 'Blossom Algorithm' for the maximum cardinality matching problem. The randomized algorithm by Mulmuley, Vazirani, Vazirani for the maximum weight matching problem. A web search should give you material on these.*

4 Applications of Matching

Assignment Problem You are given n agents available (e.g machines) and m tasks to be performed (e.g. computer processes). Each agent a_i can perform a subset (of size m_i)of the m tasks.

Problem 20. *Model the assignment problem with a graph, clearly identify vertices and edges. Briefly discuss the degrees of vertices*

Problem 21. *Suppose any agent can be assigned at most one task (out of the one it can perform) and a task can be assigned to at most one agent (who can perform it). If our goal is to get as many tasks accomplished as possible, what does this problem reduce to? (briefly discuss)*

Problem 22. *In the above problem, if the number of agents is equal to the number of tasks, what problem is this?*

Problem 23. *Suppose there is a known cost incurred when agent a_i performs task t_j. We have equal number of tasks and agents, model this by a known problem?*

Problem 24. *Suppose there are more agents than there are tasks. How would you modify the input so as we can still use a solution to the above problem in this case too.*

Problem 25. *Suppose there are more tasks than there are agents. How would you modify the input so as we can still use a solution to the above problem in this case too.*

This assignment problem is actually being used by Uber to assign drivers to rides, resource allocation in cloud and fog computing. Content delivery networks (and proxies) use it to match users to servers.

Matching and bipartite matching has been used extensively in a wide variety of applications. In many two-sided markets scenario it is used by Economists (e.g. resource allocation). Perhaps the most widely used graph algorithm is the Gale-Shapley algorithm for stable matching (also called *Pareto efficient matching*) for matching applicants to colleges, residents to hospitals, kidneys to recipients (Search for Kidney Exchanges). In switch scheduling algorithm is essentially finding an optimal way of matching input ports in switches to output ports. Read chapter 1 of Klienberg and Tardos text.

A great application is the so-called Ad allocation. The main problem a search engine faces (their revenues largely depends on it) is to deciding which ads to show (on which slot, with search results on a particular keyword). This advertisers to slots matching is a maximum matching problem with edge weights depending upon expected revenues, predicted click-rates, keywords, user locality, suitability of ads and many other factors. However, given the size of the problem the algorithm we are going to study are not suitable for this. In this case an online algorithm is used to approximately find a good matching in changing graphs. There are also many applications in bioinformatics and cheminformatics.

Chessboard Puzzle We want to study which chessboards can be covered domino tiles (See Figure 2). Every (remaining) square of the chessboard has to be covered by exactly one tile.

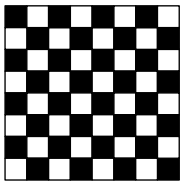

Figure 1: 8×8 chessboard

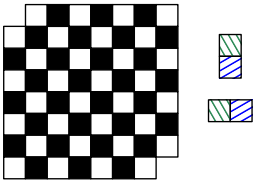

Figure 2: 8×8 chessboard with two corners removed

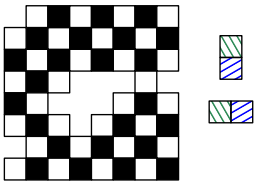

Figure 3: 8×8 chessboard with two several squares removed

Problem 26. *Make a graph with a vertex for each (remaining) square on the board. Two vertices are adjacent if the corresponding squares are adjacent on the boards. Two square are adjacent if they share a border. Argue that this graph is bipartite? Show the bipartition.*

Problem 27. *Give a characterization of edges in this graph in terms of tiling.*

Problem 28. *Describe a matching in this graph in terms of tiling.*

Problem 29. *Model the problem of whether a given board can be tiled with a matching problem in this graph.*

5 Characterization of Perfect Matching in Bipartite Graphs

In this section we try to characterize bipartite graphs that will admit a perfect matching. We will also devise a simple way to check whether a given matching is the maximum matching in the graph.

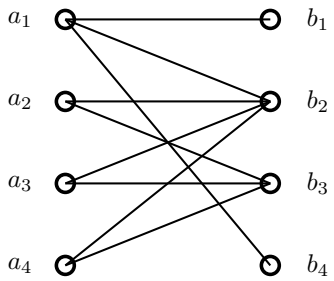

Figure 4: A bipartite graph

Problem 30. *What is the size of maximum matching in the graph in Figure 5. List all maximum matchings.*

Problem 31. *Explain the reason why there is no perfect matching in this graph. This is the most critical part of this section. Spend some time on this. Try to make as general statement as possible and verify your hypothesized reason for a few more graphs.*

To get a more precise way to describe the reason for a graph not to have a perfect matching we define neighborhood of sets of vertices. Recall that in $G = (V, E)$, for $v \in V$, $N(v)$ the neighborhood of v is defined as $N(v) := \{w : (v, w) \in E\}$.

We extend this notion of neighborhoods to sets of vertices. For $A \subseteq V$, let $N(A)$ (referred to as neighborhood of the subset A) is defined as

$$N(A) := \{w : \exists\, v \in A \text{ such that } (v, w) \in E\}$$

In other words

$$N(A) := \{w : \exists\, (v, w) \in E \text{ for some } v \in A\}$$

Problem 32. *Verify that* $N(A) = \bigcup_{u \in A} N(u)$

The notion of neighborhood of vertices and that of sets of vertices automatically extends to bipartite graphs, except that neighborhood is always a subset of the other part.

Problem 33. *Prove that if $G = (L, R, E)$ has a perfect matching, then for every $A \subseteq L$, we must have $|A| \leq |N(A)|$*

It turns out that this condition is also sufficient for existence of perfect matching.

Theorem 2 (Hall's theorem)**.** *$G = (L, R, E)$ has a perfect matching if and only for every $A \subseteq L$, we have $|A| \leq |N(A)|$*

We will derive an algorithmic proof of the Hall's theorem from the algorithm that we are going to develop. Actually we will derive it from another general result. That will make you familiar with the magic of duality.

Definition (Vertex Cover)**.** *In a graph $G = (V, E)$, a vertex cover C is a subset of V, such that every edge $e \in E$ is incident to at least one vertex in C. In other words, there is no edge in E with both endpoints in the subset $V \setminus C$.*

Simply put, vertex cover is a set of vertices that cover all edges.

Problem 34. *Prove that V is a vertex cover.*

Problem 35. *Suppose we order all vertices in V as $v_1, v_2, \ldots, v_n$. Suppose all edges are written as (v_i, v_j) such that $i < j$. Let $C = \{v_i : (v_i, v_j) \in E\}$. In other words, C contain all the lower indexed endpoint of each edge. Prove that C is a vertex cover*

Problem 36. *Let $G = (L, R, E)$ be a bipartite graph. Prove that each of L and R is a vertex cover*

Problem 37. *What is the maximum number of edges that a graph G on n vertices can contain, if it has a vertex cover of size 1. Identify the vertex cover in this case*

Problem 38. *Characterize the smallest vertex cover in a path graph P_n, the cycle graph C_n (you must cover both cases whether n is odd or even)*

In the following we see how vertex cover is related to matching.

Problem 39. *Suppose a graph G has a vertex cover of size t. Prove that the maximum matching in G is of size at most t*

Problem 40. *Suppose a graph G has two vertex covers of size t_1 and t_2. Prove that the maximum matching in G is of size at most* $\min\{t_1, t_2\}$

Thus the size of a vertex cover is an upper bound on the size of maximum matching. The smaller the vertex cover size, the tighter the bound is.

Problem 41. *Prove that size of maximum matching is at most the size of the smallest cover. This is the so-called* 'weak duality' *of matching.*

One would hope that smallest of the upper bounds will be **equal** to the size of largest matching. Indeed, such an equality holds for bipartite graphs. This is called *'strong duality'* or simply duality. We will see more of it in optimization, where the dual problem itself is a very useful optimization problem.

Theorem 3 (Kőnig - Egervary theorem)**.** *If $G = (L, R, E)$ is a bipartite graph, then the cardinality of maximum matching is equal to the cardinality of minimum vertex cover.*

We will prove this theorem, use it to prove Hall's theorem and testify that a matching that our algorithm found is actually the maximum matching.

Problem 42. *Let $G = C_3$ (a triangle), find the largest matching in G and the smallest vertex cover in it. Are they equal? If not, then does it violate the duality stated in the Kőnig theorem.*

There is a dual of maximum matching in general graphs is well known, but beyond the scope of this homework.

We will prove Hall's theorem using the Kőnig theorem. Recall that we only need to prove sufficiency, i.e. $\forall\ A \subseteq L\ |N(A)| \geq |A| \implies G$ has a perfect matching.

Problem 43. *State the contrapositive of this statement*

Let M^* be the maximum matching in G. Suppose $|M^*| < |L|$.

Problem 44. *Let C^* be the minimum vertex cover in G. Show that $|C^*| < |L|$*

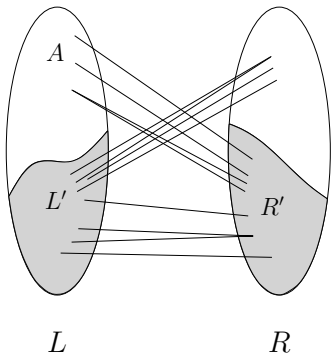

Figure 5: A bipartite graph

Problem 45. *Let $L' = C^* \cap L$ and $R' = C^* \cap R$. Let $A = L \setminus L'$ (the vertices in L that are not part of the cover), as represented in Figure 5. Show that $N(A) \subseteq R'$.*

Problem 46. *Show that $|A| > |M^*| - |L'|$*

Problem 47. *Show that $|N(A)| \leq |M^*| - |L'|$*

This completes the proof of Hall's theorem.

6 Algorithm for maximum cardinality bipartite matching

In the following we will derive an algorithm for maximum cardinality bipartite matching problem. Let $G = (L, R, E)$ be the input bipartite graph

Problem 48. *Devise a greedy algorithm to find a matching in G.*

Problem 49. *Give a small bipartite graph on 2 vertices in each side to show that this greedy algorithm doesn't produce the maximum matching.*

Problem 50. *Show that the greedy algorithm will always produce a matching that is at least half as good, i.e. if M^* is the maximum matching, then the greedy algorithm outputs a matching of size at least $|M^*|/2$.*

Given a matching M, a vertex is said to be **free** if no edge in M is incident on it, it is called matched otherwise (See Figure 6)

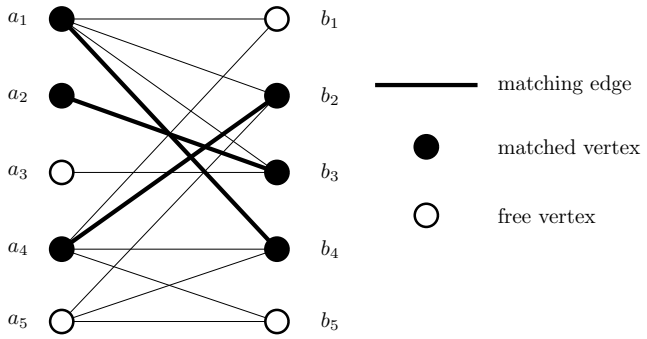

Figure 6: A bipartite graph and a matching in it

Problem 51. *Suppose M is a maximum matching in G, can there be an edge from a free vertex to a free vertex*

Recall that a path is a sequence of vertices $v_0, v_1, \ldots, v_k$ such that for $0 \leq i \leq k-1, (v_i, v_{i+1}) \in E$.

Definition. *Given a matching M, a path P is called an* **alternating path with respect to** *M or M-***alternating path** *if P alternates between edges in M and edges in $E \setminus M$.*

Definition. *Given a matching M, an M-alternating path P is called an* **augmenting path with respect to** *M or M-***augmenting path** *if the first and last vertex of P is free vertex (w.r.t M)*

Problem 52. *Identify some augmenting paths in Figure 6 and some non-augmenting alternating paths*

Problem 53. *Argue about the parity of lengths of augmenting paths. Can an augmenting path have even length?*

Problem 54. *Can endpoints of an augment path be in the same part?*

Problem 55. *Suppose there are k edges of M in an M-augmenting path, P. How many non-matching edges must be in P.*

Problem 56. *Using this last property of the augmenting paths suggest a strategy to increase the matching.*

For a matching M and M-augmenting path P, let $M' = M \oplus P = (M \setminus P) \cup (P \setminus M)$.

Problem 57. *Apply the augmentation process to the M-augmenting path a_5, b_4, a_1, b_1 in Figure 6.*

This concept of augmenting path is extremely useful, it not only is the basis of the algorithm but also provides a proof of optimality.

Problem 58. *Let M be a matching in G, such that there is an M-augmenting path P in G. Can M be a maximum matching?*

We argued that absence of augmenting path is necessary for a matching to be maximum. Next we prove that it is also sufficient. We prove that if the M is not a maximum matching in G, then there must be an M-augmenting path in G.

Suppose M is not a maximum matching. Let M^* be a maximum matching.

Problem 59. *Let $Q = M \oplus M^*$. Can Q be empty? Can M and M^* have same number of edges in Q? Which one will have more edges in Q?*

Problem 60. *Show that any vertex v of G is incident to at most one edge in $M \cap Q$ and at most one edge in $M^* \cap Q$.*

Problem 61. *Consider the subgraph $G|_Q$ of G composed of edges in Q. Argue about structure of connected components of $G|_Q$*

Problem 62. *Using the above facts, argue that there must be an M-augmenting path in G.*

We summarize the above in the following theorem.

Theorem 4. *A matching M is maximum if and only if there is no M-augmenting path in G.*

This motivates the following algorithm. Assume (possibly with renaming) that $|L| \leq |R|$.

Algorithm Algorithm for Maximum Matching

```
M ← ∅                          ▷ Initialize with an empty matching
P ← getAugmentingPath(G, M)
while P is not empty do
    M ← M ⊕ P
    P ← getAugmentingPath(G, M)
```

Begin with a matching M_0 (an empty matching or greedy maximal matching). In iteration i find a M_{i-1}-augmenting path (from a free vertex in L to a free vertex in R) and augment M_{i-1} to get M_i. Repeat until there is no augmenting path, at that point the matching is maximum by the above theorem.

Problem 63. *Give a dry run of this algorithm for maximum matching on the graph in Figure 5*

Problem 64. *If we initialize with an empty matching, what is the maximum number of iteration the above algorithm would perform*

The problem is now which path to augment through and how to find an augmenting path. Since this is just finding a path (from a free vertex in L to a free vertex in R) we will use DFS or BFS for the purpose. One needs some adjustment to the algorithm to alternately select matching and non-matching edges and check for free and matched vertices for termination.

There is a really cool trick that instead modify the input graph w.r.t a matching and then run DFS or BFS to find an odd length path.

Problem 65. *Given $G = (L, R, E)$ and a matching M, construct a directed graph D by orienting the edges in E (according) to M such that any path from a vertex in L to some vertex in R will be an M-augmenting path in G.*

Problem 66. *Give this directed graph construction for the graph and matching in Figure 6.*

Problem 67. *Let $|L| + |R| = n$ and $|E| = m$. Briefly describe an $O(n + m)$ algorithm to perform one augmentation of a matching M.*

Problem 68. *Given this augmentation, what is the total runtime of the overall algorithm for finding matching in bipartite graph.*

Given this algorithm we can now provide the algorithmic proof of the Kőnig theorem.

Let M^* be the matching returned by the above algorithm. That is M^* is a maximum matching and there is no M^* augmenting path in G. Let P be the set of vertices in G that are reachable from some free vertex in L in the directed graph constructed w.r.t M^*. The following figure shows the vertices in P w.r.t a maximum matching

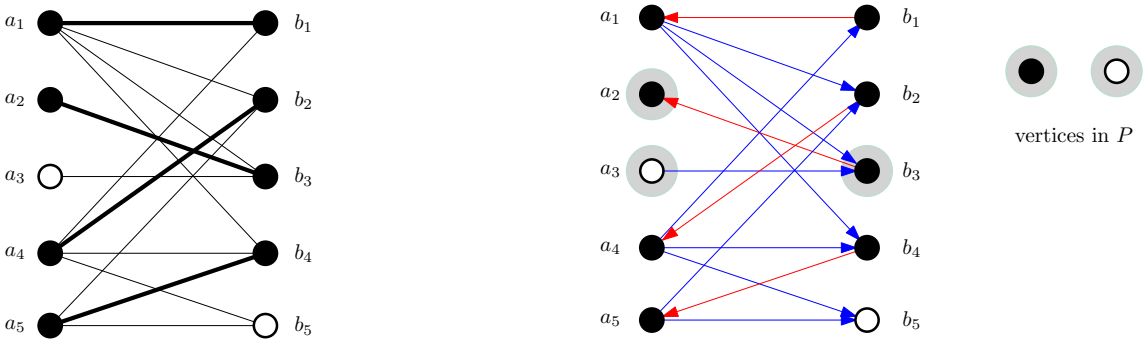

Figure 7: The directed graph w.r.t the matching. Red edges (matched edges) are directed from R to L, and blue edges (unmatched edges) are directed from L to R

Let $C^* = (L \setminus P) \cup (R \cap P)$.

Problem 69. *List vertices in C^* for the graph given in Figure 7.*

Problem 70. *Prove that C^* is a vertex cover in G. Assume that an edge (a, b) is not covered by C^*, argue about the location of a and b (w.r.t P). Consider both cases whether or not (a, b) is an edge in M^*.*

Problem 71. *Show that $|C^*| \leq |M^*|$. Argue that all vertices in $L \setminus P$ and $R \cap P$ are matched (i.e. all vertices of C^* are matched). Next show that there cannot be any edge in $(a, b) \in M^*$ such that $a \in L \setminus P$ and $b \in R \cap P$ (in other words there cannot be any matching edge in C^*).*

Problem 72. *Briefly argue that we now have a proof of the Kőnig theorem.*