Stable Matchings on Bipartite Graphs and Hall's Theorem

Jayme Prenosil

Nebraska Wesleyan University
jprenosi@nebrwesleyan.edu

May 14, 2014

Abstract

We look at bipartite graphs and how to find stable matchings on them using the Gale-Shapley algorithm and perfect matchings with Hall's theorem. We also discuss applications of the Stable Matching Theorem and Hall's Theorem.

1 Preliminary Definitions

We use the standard terminology found, for example, in Bona [2]. A graph G is **bipartite** if the vertex set of G can be split into the disjoint sets A and B so that each edge of G is adjacent to one vertex of A and one vertex of B. For example, if the vertex set of $G = \{a_1, a_2, a_3, b_1, b_2, b_3\}$, A=$\{a_1, a_2, a_3\}$, and B=$\{b_1, b_2, b_3\}$ and every edge in G is connected to one vertex in A and one vertex in B (like in Figure 1), then G is bipartite.

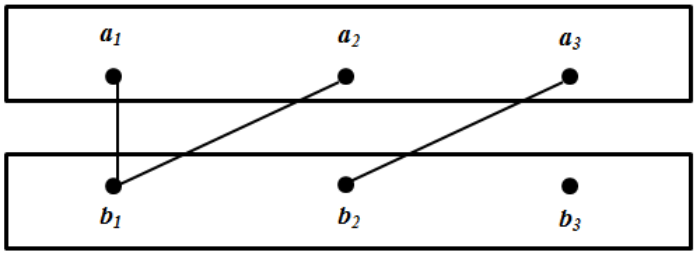

Figure 1: Bipartite Graph.

If we let G be any graph and S be a set of edges in G so that no two edges in G have a vertex in common, then we can say that S is a **matching** in G. If each vertex in G is covered by an edge in S, then we call S a **perfect matching**. Figure 2 is an example of a perfect matching in G.

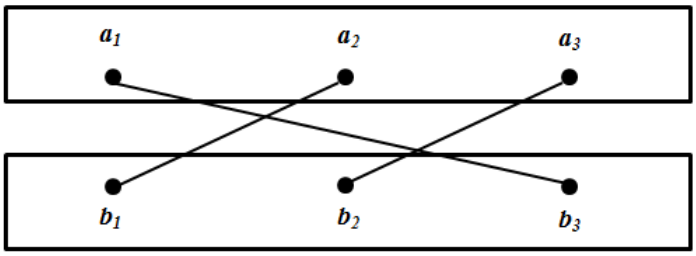

Figure 2: Perfect Matching on Bipartite Graph.

2 Stable Matching Theorem

There are many mathematical problems and results that involve the use of matchings on bipartite graphs. One such problem is called the Stable Matching Problem. This addresses the problem of finding a stable matching between two sets of elements given a set of preferences for each element. If we have a vertex set A and a vertex set B, a matching M is stable if there do not exist edges $ab, a'b' \epsilon M$ with $a, a' \epsilon A$ and $b, b' \epsilon B$ such that a prefers b' to b and b' prefers a to a'.

In 1962, David Gale and Lloyd Shapley proved that when we have two disjoint vertex sets of equal size that it is always possible to find a stable matching. A stable matching can be found using the *Gale-Shapley Algorithm.*

- Gale-Shapley Algorithm: Let every $a \in A$ propose to the first $b \in B$ on its list. Any b with exactly one proposal accepts it provisionally. Any b with more than one proposal chooses the one higher on its list. Any rejected a proposes to the next b on its list. Again, any b accepts the offer that's highest on its list. The algorithm continues until every a has been accepted or has been rejected by every b on its list at which point every pair is finalized.

With two equal size vertex sets, the Gale-Shapley algorithm always produces a stable matching. The proof of this follows.

Proof. We shall prove that the Gale-Shapley Algorithm produces a stable matching. Assume on the contrary that the Gale-Shapley Algorithm produces an unstable matching. Then there exists two elements, a and b', who are not matched by the algorithm such that a prefers b' to b, its assigned partner, and b' prefers a to a', its assigned partner. Then a proposed to b' before b since b' is before b on a's preference list. But an element only rejects another element if they have already received a proposal from an element they prefer. So if an element rejects a proposal, it perfers the final matching to the rejected element. This implies that b' prefers a' to a, which is a contradiction, thus the Gale-Shapley Algorithm always produces a stable matching. [3] □

Now that we have shown that the Gale-Shapley Algorithm always produces a stable matching with two equal sized vertex sets, we can apply this algorithm to a real world application, job searching. Let's assume that we have a set, $A = \{a_1, a_2, a_3, a_4\}$, of applicants and a set, $J = \{j_1, j_2, j_3, j_4\}$, of jobs. Each applicant has a preference list of jobs and similarly, each job has a preference list of applicants. These preference lists are below in figure 3.

Job 1: A2, A1, A3, A4	**Applicant 1: J1, J3, J2, J4**
Job 2: A4, A1, A2, A3	**Applicant 2: J3, J4, J1, J2**
Job 3: A1, A3, A2, A4	**Applicant 3: J4, J2, J3, J1**
Job 4: A2, A3, A1, A4	**Applicant 4: J3, J2, J1, J4**

Figure 3: Preference Lists of Applicants and Jobs.

Using these preference lists, we can utilize the Gale-Shapley algorithm to find a stable matching of applicants with jobs. During Round 1, j_1 proposes to a_2, j_2 proposes to a_4, and j_3 proposes to a_1. All of the applicants accept provisionally since they do not have any prior offers.

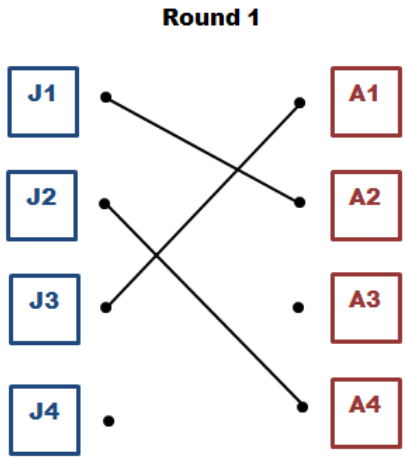

Figure 4: Round One.

Then j_4 proposes to a_2 who is already matched to j_1. But since j_4 is ranked higher on a_2's preference list, a_2 accepts j_4's offer.

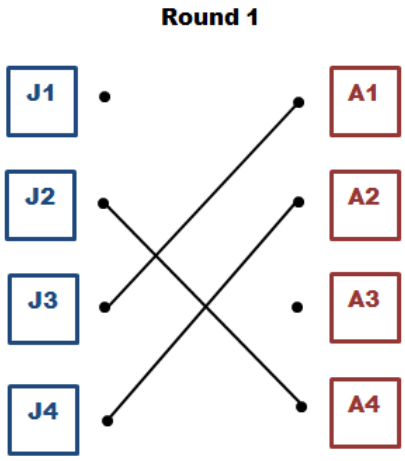

Figure 5: Round One.

Now j_1 is left unmatched and proposes to the next applicant on its preference list, a_1. Since a_1 prefers j_1 to j_3, a_1 accepts j_1's offer. j_3 is now left unmatched.

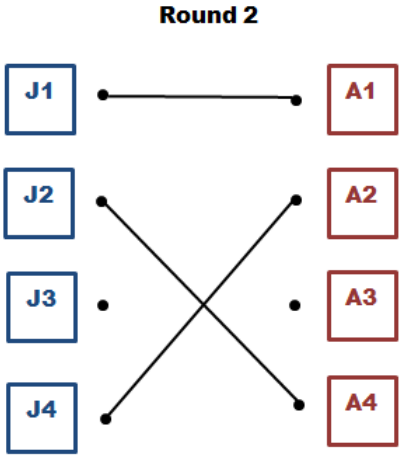

Figure 6: Round Two.

Applicant a_3 is the second choice on j_3's preference list, so j_3 proposes to a_3. Since a_3 is currently unmatched, a_3 automatically excepts.

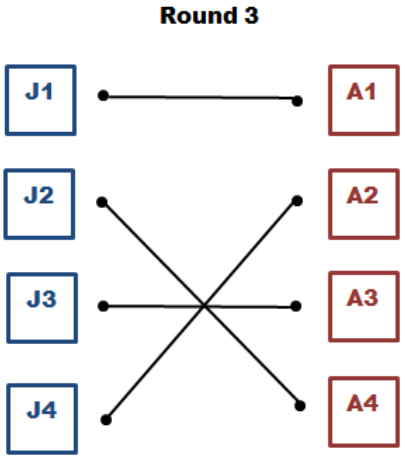

Figure 7: Round Three.

Now that every job is matched to exactly one applicant, we are done with the algorithm and each applicant accepts the jobs they were provisionally matched to. We now have a stable matching of jobs and applicants. This algorithm would work with vertex sets of any size!

3 Hall's Theorem

We discussed earlier that if every vertex on a bipartite graph is covered by an edge, then it is a perfect matching. How do we know when there exists a perfect matching? Hall's Theorem answers this.

Theorem 1. *Hall's Theorem Let $G = (X, Y)$ be a bipartite graph. Then X has a perfect macthing into Y if and only if for all $T \subseteq X$, the inequality $|T| \leq |N(T)|$ holds.*

Where $N(T)$ is the set of all neighbors of the vertices in T. In other words, $y \in Y$ is an element of $N(T)$ if and only if there is a vertex $x \in T$ so that xy is an edge.

We can use Hall's Theorem to solve a problem involving a deck of cards, which results in a pretty surprising outcome.

Example Problem: We have a normal deck of 52 cards. Divide this deck into 13 arbitrary piles of 4 cards. Prove that it is always possible to get exactly A, 2, 3, ..., J, Q, K by picking one card from each pile. [1]

Proof. This problem can be transformed into a bipartite graph. One set of 13 vertices will represent the piles of 4 cards and the other set of 13 vertices will represent cards A, 2, 3,..., J, Q, K. We notice that each vertex has degree four. We want to find a perfect matching. Take a set S, a subset of the piles, 4*$|S|$

edges (4 from each vertex) leave this set. The receiving vertices have also degree four so

$$|N(S)| \geq \frac{\text{number of edges}}{4} = \frac{4|S|}{4} = |S|$$

otherwise there is a vertex with degree greater than 4. Therefore, by Hall's Theorem, a perfect matching exists and we will always end up with exactly A, 2, 3, ..., J, Q, K.

□

Now that we have walked through an example utilizing Hall's Theorem, we can prove that Hall's Theorem actually ensures that a perfect matching exists.

Proof. Let $G = (X, Y)$ be a bipartite graph. First we shall prove that X has a perfect matching into Y only if $|T| \leq |N(T)|$ holds for all $T \subseteq X$. We will assume that $|T| > |N(T)|$. We then know that T can't be matched to $N(T)$ since T contains more vertices than $N(T)$, or the set of neighbors of T. This also means that we can't match T into Y because this would result in a matching of T into $N(T)$. Now, this means that we can not find a matching of X into Y because this matching would contain a matching of T into Y since by assumption, $T \subseteq X$. Thus, we can only have a perfect matching of X into Y if $|T| \leq |N(T)|$. Now we shall prove that if $|T| \leq |N(T)|$ holds for all $T \subseteq X$, then X has a perfect matching into Y. We shall prove this by induction on $|X|$. We can assume that we know the statement holds for all nonnegative integers less than $|X|$ and we shall prove it for $|X|$. We also assume that the inequality $|T| \leq |N_G(T)|$ holds for all $T \subseteq X$. We then have two cases to look at.

1. First, we shall assume that for every $T \subset X$, $|T| < |N_G(T)|$ holds. We let x and y be connected vertices, with $x \in X$ and $y \in Y$. We define graph $G' = G - x - y$ and let A be any nonempty subset of $X - x$. We then know that $|A| < |N_G(A)|$ which implies that $|N'_G(A)| \geq |N_G(A)| - 1 \geq |A|$. Our induction hypothesis then implies that we can match $X - x$ into $Y - y$ in G'. If we add edge xy to this matching, we get a perfect matching of X into Y.

2. Now, we shall assume that we have a subset $B \subset X$ so that the equality, $|B| = |N_G(B)|$ holds. We can split G into two subgraphs, G_1 and G_2, and show that both subgraphs satisfy the induction hypothesis on their own. We let G_1 be the subgraph defined by $B \cup N(B)$. Let G_2 be the graph of G minus the vertices of G_1. Let's choose any subset $T \subseteq B$. Then we know that $N_G(T) \subseteq N_G(B)$ which implies that $N_{G_1}(T) = N_G(T)$ and thus $|N_{G_1}(T)| = |N_G(T)| \geq |T|$. We now want to show that G_2 satisfies the induction hypothesis. We choose any subset $U \subseteq X - B$. Then $N_G(U \cup B) = N_{G_2}(U) \cup N_G(B)$. We know that $|N_{G_2}(U)| = |N_G(U \cup B)| - |N_G(B)| \geq |U \cup B| - |B| = |U|$ since it's a union of disjoint sets. We can apply the induction hypothesis to both subgraphs G_1 and G_2 which means that B can be matched into $N_G(B)$ and $X - B$ can be matched into $Y - N_G(B)$.

Thus, we have proven that X can be matched perfectly into Y. $\square$

References

[1] Bansal, Nikhil, *Graphs and Algorithms*, published electronically at http://www.win.tue.nl/ nikhil/courses/2012/2WO08/lecture3.pdf, 2012.

[2] Miklos Bona, *A Walk Through Combinatorics*.

[3] Shorey, R., *The Algorithm*, published electronically at http://www.sccs.swarthmore.edu/users/06/rshorey/math/alg.html, 2010.