Stable Matchings

Abel Romer

November 1, 2020

Abstract

We introduce the concept of a *matching* in a graph and explore a sub-category of matchings called *stable matchings*. We provide some motivating examples and applications of the concept, and address the problem of identifying stable matchings in arbitrary complete graphs. We introduce the Gale-Shapley algorithm for finding such matchings. We explore properties of the set of solutions to the stable matching problem and introduce the concept of a distributive lattice. We conclude by asking other questions regarding stable matchings and related concepts.

1 Introduction

A matching in a graph G is a subset M of $E(G)$ such that all edges $e \in M$ are vertex-disjoint. Matchings can be thought to belong to a class of problems in graph theory called *extremal problems*. These are problems that deal with maximization and minimization. In the case of matchings, we are often concerned with those matchings that are *maximum*–that is, those that contain the largest number of edges. Matchings that cover all the vertices of G are said to be *perfect matchings*.

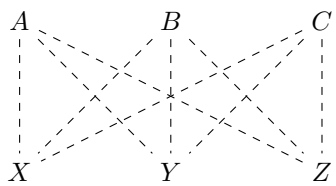
In this paper, we address a special kind of perfect matching, called a stable matching. Stable matchings are pairings in complete bipartite graphs $K_{n,n}$, optimized according to preference lists assigned to each vertex. The concept is typically introduced by the following problem:

Problem 1.1 (The Stable Pairing Problem)**.** *Suppose we have six dancers A, B, C, X, Y and Z. Each of A, B and C are leads, while each of X, Y and Z are followers. Each lead is to be uniquely matched with one follower. Additionally, each lead has ranked preferences for each follower, and each follower has ranked preferences for each lead. Suppose, or example, the following circumstance:*

- A: $\{x, y, z\}$
- B: $\{z, y, x\}$
- C: $\{y, z, x\}$
- X: $\{c, b, a\}$
- Y: $\{c, b, a\}$
- Z: $\{b, a, c\}$

How might we go about matching leads with followers? What criteria should we use to determine if a particular matching is suitable?

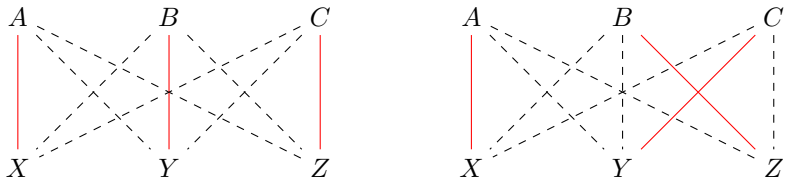
/noindent We begin by representing our problem as the complete bipartite graph $K_{3,3}$:



Each dashed edge represents a potential pairing we could add to our stable matching. We define a matching to be *stable* as follows:

Definition 1.1 (Stable Matching)**.** *A perfect matching M in a complete bipartite graph $K_{n,n}$ is said to be stable if there are no two vertices $u \in X$, $v \in Y$ such that u prefers v over its current partner, v prefers u over its current partner, and $uv \notin M$.*

Consider the following two matchings:



The matching on the left is not stable. This is because the preference list for vertex $B = \{z, y, x\}$ shows that B would prefer to be paired with Z *and* the preference list for vertex $Z = \{b, a, c\}$ shows that Z would prefer to be with B. By making this adjustment, we obtain the graph on the right. Note that although one vertex (X) is matched with its least preferred partner, no two vertices would prefer each other over their current pairing.

At this point, we might wish to ask a number of questions. These include:

- Which graphs have stable matchings?
- How do we determine whether a stable matching exists?
- Are there classes of stability? Can a stable matching be *more* stable?
- What is the use of stable matchings? Does this problem have practical value?

We will take a brief moment to address the question of applicability below before exploring the other, more theoretical questions.

1.1 Applications

Applications of the theory of stable matchings to the practical world are, perhaps unsurprisingly, wide-ranging. Two of particular note regard its application to user-server assignment on the Internet [1], and what is called the kidney matching problem [2]. We briefly formulate these problems below.

1.1.1 User-Server Assignment

In basic terms, when we visit a webpage, a request is sent from our computer to a server somewhere else in the world to send us packets of data that our web browser then renders into Facebook or Twitter or YouTube. The server that receives these requests is rarely one computer sitting some data center in Tucson or Abu Dhabi, but rather a distributed network of computers. This type of network is called a *content delivery network* or CDN. When building a CDN, engineers must consider the question of how to efficiently direct requests to servers. This is a formulation of the stable matching problem.

The aim is to match users accessing content from a CDN to servers providing that content. Users prefer to be directed to servers that can deliver content quickly and successfully (without data loss). Servers prefer users according users' IP addresses or internet services providers (among other things) [1]. The set of users and servers may then be assigned preference lists, allowing the problem to be addressed by the Gale-Shapley algorithm (see below).

1.1.2 Kidney Matching Problem

This is indeed, as one might expect, the problem of matching kidneys to people with renal failure. For those interested in the medical side of things, the first successful kidney transplant took place on December 23, 1954 and the patient lived for eight years after the operation. It is widely accepted that kidneys from living donors produce appreciably more successful results (87% 5-year success rate from living donors, compared to 80%

success from deceased donors), and living donors are comparatively uncommon. Until 1986, living donors who were found incompatible with their particular recipient would be outright rejected. This eliminated a pool of potential donors who might have otherwise been paired to different patients. In 1986, transplantation expert Felix Rapaport proposed a kidney exchange program for living donors [2]. This necessitated a process for matching patients and compatible donors.

In the kidney matching problem, there are a number of patients P_1 to P_n and kidneys k_1 to k_n, and each patient has ordered preferences for kidneys. Additionally, there is a set of patient-kidney pairs representing those matchings that are incompatible [2]. The problem can be solved using something called the top trading cycle algorithm; while this problem is not exactly the same as the stable matching problem, there is theoretical overlap between the two.

2 Properties of Stable Matchings

We address the question of whether a stable matching exists for some graph $G = K_{n,n}$ with an arbitrary preference list P. As it turns out, mathematician-economists and Ph.Ds David Gale and Lloyd Shapley proved in 1962 that a stable matching exists on all such graphs G, and provided an algorithm for finding it [3].

2.1 Gale-Shapley Algorithm

We present an overview of the algorithm and some simple examples, followed by a proof that it does indeed identify a stable matching.

2.1.1 Overview

The algorithm proceeds through a number of rounds.

- In the first round, each of the leads "proposes" to their preferred partner. [What we really mean is "ask to dance" instead of "propose", but "propose" is more concise.] Note that at this point it is possible that a follower receives multiple propositions. The followers (if proposed to) then provisionally accept the proposition of the lead they most prefer.
- In each of the following rounds, each unmatched lead proposes to the follower s/he most prefers to whom s/he has *not yet proposed*. Then each follower either updates their current partnership (if s/he has received a proposition from a more preferable lead), or accepts the proposition of the lead s/he most prefers.
- This process continues until either (a) every lead proposes to every follower, or (b) every lead (and consequently every follower) finds him/herself provisionally matched.

Let's consider some examples:

Example. *Let's begin by applying this algorithm to our example problem (Problem 1.1):*

1. *First, A, B, and C propose to their preferred partners. In this case, X, Y, and Z all receive propositions. They each provisionally accept.*
2. *None of A, B, and C remain unmatched, so we are finished.*

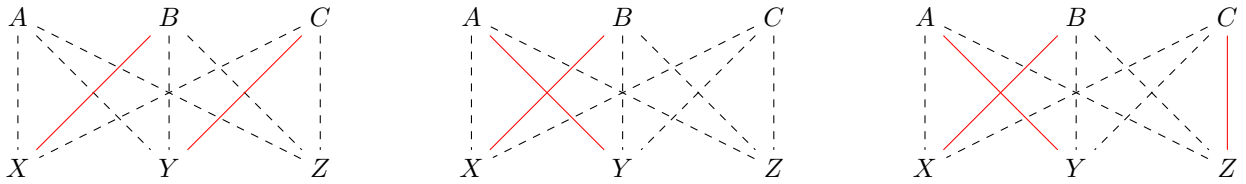
While this example is perhaps too simple, it does illustrate the point that if each lead most prefers a distinct follower (that is, no two leads most prefer the same follower), then finding a stable matching is as simple as pairing the leads to their top preferences. Let's examine a more complicated case:

Example. *Consider the following case:*

- A: $\{x, y, z\}$
- B: $\{x, y, z\}$
- C: $\{y, z, x\}$
- X: $\{b, c, a\}$
- Y: $\{a, b, c\}$
- Z: $\{b, a, c\}$

1. *A proposes to X, B proposes to X, and C proposes to Y. X accepts B's proposition, Y accepts C's proposition, and Z remains un-proposed to.*
2. *B and C are now provisionally matched, and no longer propose. A is single, and so proposes to their second choice: Y. Y, much preferring A over C, ditches C. The status of X and Z remains unchanged.*
3. *C is now unmatched, and proposes to their second choice: Z. Z, having received no other propositions, accepts.*
4. *All parties are now provisionally matched.*

This sequence is illustrated below:



2.1.2 Proof

We show that the Gale-Shapley algorithm (1) terminates, (2) creates a perfect matching (i.e. every lead is paired with a distinct follower and vice versa), and (3) each edge in this matching is stable.

1. Consider a matching to be made between n leads and n followers. As no lead may propose to the same follower twice, after n rounds, each lead has either found a partner, or proposed to every available follower and been rejected. In either case, the algorithm terminates.
2. We proceed by contradiction. Suppose there exists a lead l who remains unmatched when the algorithm terminates. Then there must also exist an unpaired follower f, since there are the same number of leads and followers, and each person is paired with exactly one other person. (Furthermore, note that any follower who receives a proposition will ultimately end up matched. This is because followers always accept one of the propositions they receive.) Therefore, f must not have received a proposition. However, in order for the algorithm to terminate, l must either be paired or have proposed to every follower. Since s/he cannot have proposed to f, s/he must not have been unmatched. This is a contradiction. We conclude that all leads and followers are paired when the algorithm terminates; these pairings constitute a perfect matching.
3. We again proceed by contradiction. Suppose that the algorithm produces an unstable pairing between a lead l and a follower f. In other words, there exists a different follower f' who prefers l over their current partner l', *and* who is also preferred by l over l's current partner f. If this is the case, then l must have proposed to f' first. But followers can only reject a proposal if they then accept a different proposal from someone they prefer. In the case of f', we must conclude that s/he prefers l' to l, as in the final configuration they are paired together. This is a contradiction. We therefore conclude that all matchings produced by the Gale-Shapley algorithm are stable.

2.2 Stability

We have shown that a stable matching exists for any set of preferences on any $K_{n,n}$. However, it is worth examining the different types of stable matchings that exist and what properties they have. We begin with the stable matching SM_l produced by the Gale-Shapley algorithm:

SM (l)**.** *This matching gives the leads their highest preference, subject to the constraint that all matchings are stable. To gain an intuition for why this is the case, note that the only circumstance where a lead l does not get their first preference f is if some other lead l' shares the same preference. In this case, f must prefer l' over l, and therefore pairing l and f would be unstable.*

Note, then, that allowing followers to propose first instead of leads produces a stable matching SM_f where the followers get their highest preference. This follows by symmetry from the argument above. We now have matchings that favor both parties. Are there any intermediate stable matchings?

Example. *Consider the following:*

- A: $\{y, x, z\}$
- B: $\{z, y, x\}$
- C: $\{x, z, y\}$
- X: $\{b, a, c\}$
- Y: $\{c, b, a\}$
- Z: $\{a, c, b\}$

Here, we have three stable matchings: $M_1 = \{AY, BZ, CX\}$, $M_2 = \{AX, BY, CZ\}$ and $M_3 = \{AZ, BX, CY\}$. M_1 provides the leads with their first choices, M_2 does the same for the followers, and M_3 gives everyone their second choice.

It seems, from the perspective of the leads, that we have a sort of hierarchy of stable matchings: those where they get their top preferences, those where they get their intermediate preferences, and those where they get their lowest preferences (M_1, M_3 and M_2, respectively, in the example above). This hierarchical structure exists for all stable matching problems and has a specific name: a *distributive lattice* [4].

Definition 2.1 (Distributive Lattice)**.** *A lattice is a partially ordered set P with two operations meet $\wedge$ and join $\vee$. For elements x, y and z in P, $z = x \wedge y$ if $z \leq x$, $z \leq y$ and for all $w \in P$ such that $w \leq x$ and $w \leq y$, $z \geq w$. Similarly, $z = x \vee y$ if $z \geq x$, $z \geq y$ and for all $w \in P$ such that $w \geq x$ and $w \geq y$, $z \leq w$. A lattice is distributive if for every $x, y, z \in P$, $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$ and $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$.*

We examine how a set of solutions S to the stable matching problem fits this definition. We show that (1) S is a partially ordered set, (2) S forms a lattice, and (3) this lattice is distributive.

1. We define a comparison operator $\leq$ as follows: for stable matchings $M, N \in S$, $M \leq N$ if and only if every lead prefers matching N to matching M. In the above example, $M_1 \geq M_2 \geq M_3$. However, it is possible that leads may disagree over which matching they prefer. If this is the case, M and N are considered incomparable. We see that the three requirements of a poset follow directly from the definition of $\leq$. Let $M, N, L \in S$.
   (a) Reflexivity ($M \leq M$)
   (b) Anti-symmetry ($M \leq N$ and $N \leq M$ are not both true)
   (c) Transitivity (if $M \leq N$ and $N \leq L$, then $M \leq L$)
2. We define the operations $\wedge$ and $\vee$ for stable matchings $M, N \in S$. Let $M \wedge N$ define the matching where each lead is given their *lowest* preference follower of those they are assigned in M and N. Let $M \vee N$ define the matching where each lead is given their *highest* preference follower of those they are assigned in M and N. In the above example, $M_1 \wedge M_2 = M_2$ and $M_1 \vee M_2 = M_1$. It follows that both $M \wedge N$ and $M \vee N$ are (a) matchings and (b) stable.
   (a) Clearly, every lead must be assigned to a follower in both $M \wedge N$ and $M \vee N$. Suppose, then, that two leads l and l' are assigned to one follower f in $M \vee N$. Without loss of generality, suppose l is preferred by f. Then, l and f form an unstable pair in whichever of M or N they are unmatched. The argument for $M \wedge N$ follows by symmetry.

(b) Suppose some lead and follower form an unstable pair in $M \vee N$. Then this same unstable pair must also exist in at least one of M or N. Again, the argument for $M \wedge N$ follows by symmetry.

Finally, we show that the operations we have defined have the properties of meet and join. We show that (a) $M \vee N$ is greater than or equal to both M and N, and (b) $M \vee N$ is less than or equal to all other matchings greater than or equal to M and N. In both cases, $M \wedge N$ follows by symmetry.

(a) $M \vee N$ must be at least as large as M and N because each lead's choice is either improved or unchanged, as defined by the operation $\vee$.

(b) $M \vee N$ must be less than or equal to all other matchings greater than or equal to M and N because any matching strictly better than $M \vee N$ would require a lead-follower pairing not present in M or N.

3. Distributivity follows from the fact that $M \vee N$ and $M \wedge N$ behave similarly to the minimum and maximum operations on total orderings. More information can be found in Knuth [4]).

The fact that a set of solutions S to the stable matching problem forms a distributive lattice has algorithmic consequences, as certain computations can be performed quite efficiently on distributive lattices. Additionally, it provides a framework for growing/enlarging a set of stable matchings by applying the operations join and meet.

3 Additional Questions

We will end this paper with a couple of questions.

- Questions related to the properties of graphs:
  - What characteristics do graphs with a certain number of stable matchings have?
  - How many stable matchings can a graph G have?
  - Can every perfect matching in G be stable?
- Relaxations of stable matchings:
  - What are the properties of stable matchings that allow multiple leads to pair with one follower? Or vice versa?
  - What about stable matchings where each person can be paired with every other person?
  - Can we introduce conditionals into the preference lists?

4 References

[1] Maggs, Bruce; Sitaraman, Ramesh (2015). "Algorithmic nuggets in content delivery". ACM SIGCOMM Computer Communication Review. **45** (3).

[2] Masso, Jordi. (2015). "The theory of stable allocations and the practice of market design. The Nobel Prize in Economics 2012 for Alvin E. Roth and Lloyd S. Shapley". Contributions to Science. **11**: 103-112. doi:10.2436/20.7010.01.218

[3] Gale, D.; Shapley, L. S. (1962). "College Admissions and the Stability of Marriage". American Mathematical Monthly. **69** (1): 9-14. doi:10.2307/2312726. JSTOR 2312726.

[4] Knuth, D. E. (1997). *Stable Marriage and Its Relation to Other Combinatorial Problems: An Introduction to the Mathematical Analysis of Algorithms.* CRM Proceedings and Lecture Notes (10). English translation. American Mathematical Society.