Algorithmic Game Theory- Stable Marriage Problem

Michael Levet

January 9, 2016

1 Introduction

In mathematics, economics, and computer science, matching problems deal with taking a set of elements and pairing them off such that any two pairs are disjoint. Note that elements may be unpaired. A marriage is a prime example of a matching, as no person can have multiple spouses. Some people may prefer to remain single as well. These are all acceptable options in a stable marriage.

This blog entry will explore the Stable Marriage Problem, including the Gale-Shapley algorithm and results about the stable matchings it yields.

2 Stable Marriage Problem

In the last section, the notion of a matching was described intuitively as a pairing of a subset of elements, leaving potentially unpaired elements. This is formalized by defining a matching as an involution. Of course, a definition of an involution is first needed. Consider some set $X$and a function $f : X \to X$. Then the function f is an involution if f is its own inverse. That is, $f \circ f(x) = x$, for every $x \in X$.

Let's consider an example of an involution. Let $X = \{0, 1, 2, 3, 4, 5\}$, and let $f : X \to X$be given by $f(5) = 5$ and $f(x) = 4 - x$ for every $x \neq 5$. Observe that for $x = 5$, $f \circ f(5) = 5$, which satisfies the involution property. Now suppose $x = 3$. So $f(x) = 1$, and $f(1) = 3$. So $f \circ f(3) = 3$. More generally, if $x \neq 5$, then $f \circ f(x) = 4 - f(x) = 4 - (4 - x) = 4 - 4 + x = x$. So the function provided is an involution. Observe as well that this involution pairs the elements $0 - 4$and $1 - 3$, leaving 2 and 5 unpaired.

Another example of a matching is a marriage. We have John marrying Jane, Bob marrying Betsy, and Carl and Carol remaining single. We define this formally with the set $X = \{$John, Jane, Bob, Betsy, Carl, Carol $\}$and $f : X \to X$ defined by:

- f(John) = Jane, and f(Jane) = John
- f(Bob) = Betsy, and f(Betsy) = Bob
- f(Carl) = Carl, f(Carol) = Carol

So now that a matching has been defined, the next step is to introduce the Stable Marriage Problem. We start with two types of players, given by sets X and Y with $|X| = |Y|$. For example, we could have sets of Firms and Workers, or Men and Women. Each member of a set can only be matched with a member of the other set or itself. That is, no two distinct elements in the set X can be matched together, nor can two distinct elements in the set Y be matched together. However, a player can be unmatched, which is equivalent to matching it with itself. Each player can only be matched with at most one other player.

Finally, each player i has a strict and transitive preference relation $\succ^i$. In particular, we only care about the players with whom i would prefer to be matched with over staying single. If player i prefers to stay single than be matched with player j, we need not consider player $j$when constructing a stable marriage.

Note: $j \succ^i k$ reads that player i strictly prefers player j to player k.

So we know that a stable marriage is a matching. So what exactly does it mean for a marriage to be stable? Intuitively, if there is a player of type X and a player of type Y who would prefer each other to their current mates, then the marriage is not stable. So consider the example marriage above. Suppose John and Jane are married, and Bob and Betsy are married. If Bob would prefer Jane to Betsy as his spouse, and Jane would prefer Bob to John as her spouse, then the marriage is not stable. The reason for this is that Bob and Jane would leave their respective spouses for each other, which maximizes their respective preferences.

So formally, a marriage $\mu : X \cup Y \to X \cup Y$ is stable if and only if for every pair $(i, j) \in X \times Y$, we have $j \not\succ^i \mu(i)$ or $i \not\succ^j \mu(j)$ (or both). That is, there is no couple that would prefer to be with each other instead of their current spouses (which includes being single).

Let's consider an example. Let $X = \{x_1, x_2, x_3\}$and $Y = \{y_1, y_2, y_3\}$. Below are each player's preferences in descending order. That is, the first element in the list is the player's most desirable spouse. The last element on the given list is the last alternative preferable to being single. So for example, x_1 as shown below prefers y_2 the most, then y_1. However, x_1 would rather stay single than be matched with y_3.

- x_1: y_2, y_1
- x_2: y_1, y_2, y_3
- x_3: y_1, y_2
- y_1: x_1, x_3, x_2
- y_2: x_2, x_1, x_3
- y_3: x_1, x_3, x_2

Consider the matching of $x_1 - y_1$, $x_2 - y_2$, $x_3 - y_3$. Observe that y_1 and y_2 have their most preferred spouses. Even though x_1 and x_2 would prefer to swap spouses, their respective partners would not prefer to swap. So thus far, the definition of a stable marriage seems to hold. Now consider the pairing of $x_3 - y_3$. Observe that x_3 prefers to be single rather than matched with y_3. Recall that remaining single is equivocal to being matched with oneself. So x_3 would prefer to be matched with a spouse (namely, x_3) that would prefer mutually prefer to be matched with x_3. Thus, this marriage fails to be stable. Instead, the marriage $x_1 - y_1$, $x_2 - y_2$, x_3, and y_3 is stable.

In a later section, the notion of stability will be generalized into the solution concept known as the core. The notion of the core provides a more comprehensive framework to discuss cooperative games, such as the stable marriage problem.

For now, let's introduce the Gale-Shapley algorithm to solve the Stable Marriage Problem.

3 Gale-Shapley Algorithm

In solving the Stable Marriage Problem, we seek to design a mechanism that takes individual preferences and returns a stable matching. The mechanism design by Gale and Shapley is an algorithm which follows an intuitive proposal model in the marriage market, allowing input from both types of players. We start by exploring the algorithm, its correctness, and its time complexity. From there, we will examine results about the matchings produced by this mechanism.

The Gale-Shapley algorithm is very straight-forward. It starts with the two types of players, X and Y. One of these types serves as the proposing, and the other type as the recipient type. We begin at each iteration by selecting an unmatched element $x_i \in X$. The element x_i then proposes to its potential mates in preference order until its proposal is held or it exhausts all potential mates. Note that x_i will not propose to a player $y_j \in Y$ if it prefers being single to being matched with y_j.

The players of type Y cannot make proposals. They can only accept or reject proposals as they receive them. If a player $y_j \in Y$ is unmatched, it will hold a proposal from player x_i if and only if y_j prefers to be matched

with x_i. If y_j is presently holding a proposal by another player $x_k \in X$, then y_j will accept x_i's proposal if and only if $x_i \succ^j x_k$. If this happens, then x_k is unmatched and placed back into the pool of X elements seeking a mate.

Let's work through an example. Recall the sets $X = \{x_1, x_2, x_3\}$and $Y = \{y_1, y_2, y_3\}$with preferences:

- x_1: y_2, y_1
- x_2: y_1, y_2, y_3
- x_3: y_1, y_2
- y_1: x_1, x_3, x_2
- y_2: x_2, x_1, x_3
- y_3: x_1, x_3, x_2

Suppose the elements in set $X$are the proposers. Consider the steps of this algorithm.

- We start with element x_1, who proposes to y_2. As y_2 prefers x_1 to being single, $y_2$holds the proposal from x_1.
- Next, x_2 proposes to y_1, who holds x_2's proposal.
- Now, x_3 proposes to y_1, who rejects his proposal. Next x_3 proposes to y_2, who also rejects his proposal. Thus, x_3 will remain unmatched.

As there are no more unmatched Xelements, the algorithm terminates and the Y elements are matched as specified by the proposals being held. So the marriage here is $x_1 - y_2, x_2 - y_1, x_3, y_3$. Observe that this is a different stable marriage than was deduced in the previous section. The stable marriage of $x_1 - y_1, x_2 - y_2, x_3, y_3$ will be returned if the Yelements proposed instead of the X elements.

Let's contrast the cases in which the Xelements proposed versus when the Y elements proposed. Observe that the stable matching returned by the Gale-Shapley algorithm favors the proposers. That is, when the X elements proposed, the X elements obtained their preferred mate in comparison to when the Y elements proposed. Similarly, when the Y elements proposed, they received their preferred mates in comparison to the stable marriage when the X elements proposed. In a later section, it will be proven that the Gale-Shapley algorithm favors the proposers.

I now provide a Java implementation of the Gale-Shapley algorithm.

Consider first the Actor class. This class is the backbone of the implementation, modeling a single player in the Stable Marriage Problem. For this implementation, we classify players as either Firms or Workers. That is, we have a set of Firms and a set of Players, with one of of these sets being the proposing set. Functionality for both making proposals and accepting proposals is included in the Actor class. Note that a different component is responsible for orchestrating how the Actors interact.

See the tutorial page for the code listing Actor.java

The component that handles Actor interaction and the logic-flow of the Gale-Shapley algorithm is the Match-Maker class.

See the tutorial page for the code listing MatchMaker.java

Finally, consider the driver class for this implementation. This demo uses the following actors with their preferences shown below. It demonstrates both firms as the proposers, and workers as the proposers.

Firms				Workers			
F_1	F_2	F_3	F_4	W_1	W_2	W_3	W_4
W_3	W_1	W_1	W_3	F_1	F_1	F_3	F_4
W_1	W_3	W_3	W_1	F_3	F_4	F_1	F_1
	W_4		W_4		F_2	F_4	F_3
	W_2		W_2		F_3		

See the tutorial page for the code listing StableMarriage.java

Now that the algorithm is clear, the next goal is to prove and understand its correctness. In order for the Gale-Shapley algorithm to be a valid mechanism, it must terminate and produce a stable marriage for any valid set of preference profiles.

Theorem 1: The Gale-Shapley algorithm terminates and produces a stable matching.

Claim 1, Claim 2, and Claim 3 below together imply Theorem 1.

Claim 1: The Gale-Shapley Algorithm correctly terminates.

Proof: This proof is by extremality. Each proposer can prefer at most all acceptors to being single. Suppose this is the case, and a single proposer receives his worst mate. Then the remaining $n-1$ proposers must make at most $n-1$ proposals each. By the algorithm, each proposer will solicit a potential mate at most once. Therefore, at most $n+(n-1)^2 = n^2-n+1$ proposals will be executed. If no unmatched proposer is eligible to solicit a mate, then the algorithm will terminate. And so, the algorithm must terminate after at most n^2-n+1 proposals. QED.

Corollary: The Gale-Shapley algorithm executes in $O(n^2)$ time.

Proof: By Claim 1, we have at most $O(n^2)$ proposals necessary to construct the matching. Each proposal takes $O(1)$ time, as does selecting the next available proposer to match. It follows that the algorithm takes $O(n^2)$ time. QED.

Claim 2: The Gale-Shapley algorithm produces a matching.

Proof: It suffices to show that no player is matched with more than one mate. Observe that the proposers solicit each potential mate in preference order. By the algorithm, a proposer stops soliciting when a potential mate holds his proposal. It follows that no proposer will be matched with multiple mates.

Similarly, each acceptor can either accept or reject a proposal. If the acceptor is presently single, then the definition of a matching will remain satisfied either by accepting or rejecting the proposal. If the acceptor is already holding a proposal, then the acceptor can hold the new proposal only by unmatching herself from the current suitor. It follows that no acceptor will be matched with more than one suitor. It follows that the Gale-Shapley algorithm produces a matching. QED.

Claim 3: The matching produced by the Gale-Shapley algorithm is stable.

Proof: The proof is by contradiction. Without loss of generality, let Xbe the set of proposers. Let $\mu : X \cup Y \to X \cup Y$be the matching returned by the Gale-Shapley algorithm. Suppose $\mu$is not stable.

Then there exists a pair $(i, j) \in X \times Y$such that $j \succ^i \mu(i)$and $i \succ^j \mu(j)$. By the algorithm, player $i$would have proposed to player $j$before player $\mu(i)$since $j \succ^i \mu(i)$. If player $j$was single, then player $j$would have held player i's proposal. If instead player $j$was not single and (without loss of generality) holding $\mu(j)$, then player $j$would have instead opted to hold player i's proposal. However, this contradicts the fact that the algorithm matched player $i$with player $\mu(i)$. It follows that μmust be stable. QED.

4 Stable Matchings and the Core

Recall from the end of Section 2 that the Core was briefly mentioned. It is the standard solution concept in cooperative games. Many important results regarding the Gale-Shapley algorithm and the Stable Marriage Problem are phrased in terms of the Core. It is also important to understand how the Stable Marriage Problem fits in with the existing and more general theory of cooperative games. For this reason, it is important to introduce briefly the notion of the Core.

Recall from basic notions of game theory that the Nash equilibrium is the standard solution concept. A set of strategies constitutes a Nash equilibrium if no player can unilaterally deviate and improve his or her outcome. The Core generalizes this notion of the Nash equilibrium to coalitions. Intuitively, an allocation is in the Core if no subset of players can deviate, possibly by cooperation, and improve their outcomes.

Let's start with the framework. In a cooperative game, each player $i \in \{1, ..., n\}$ has an initial endowment ω^i from the set of commodities. Note that a given player may have an initial endowment of nothing. Each player may belong to zero or more potential coalitions, where a coalition is a set $S \subset \{1, ..., n\}$. The coalition where $S = \{1, ..., n\}$ is referred to as the grand coalition. Every coalition has a set of available actions, and each player i has a preference relation $\succeq^i$ over all the coalitions and actions available to said player.

In the stable marriage problem, each player i's initial endowment is $\omega^i = i$, as each player is initially single or matched with itself. The coalitions come into play when we ask the question- when does an allocation not make sense? The answer is pretty clear- when a coalition of players can collaborate and improve their outcomes. If a coalition can improve upon the given allocation, the coalition is said to block the allocation. This notion of a blocking coalition gets to the heart of the core. An allocation is said to be in the core if no blocking coalition exists.

Let's formalize these notions some. Let $x = (x_1, x_2, ..., x_n)$ be an allocation. The allocation x is said to be dominated by some other feasible allocation $y = (y_1, y_2, ..., y_n)$ if there exists a coalition S such that for every $s \in S$, $y_s \succeq^s x_s$; and for some $j \in S$, $y_j \succ^j x_j$. That is, all members in the coalition S are at least indifferent to the allocation y in comparison to the allocation x; and at least one member of S strictly prefers y to x. The coalition S is said to then block x.

The above definition of the core is actually pretty formal. The core contains the set of allocations that are not dominated; or equivocally, the set of allocations for which no blocking coalition exists.

So how does the notion of the core relate to the Stable Marriage Problem? As it turns out, the core is exactly the set of stable matchings. This fact will be proven later, but let's first start with an example to develop some intuition.

Recall the example from Section 2 with the sets $X = \{x_1, x_2, x_3\}$and $Y = \{y_1, y_2, y_3\}$, and the preferences given below.

- x_1: y_2, y_1
- x_2: y_1, y_2, y_3
- x_3: y_1, y_2
- y_1: x_1, x_3, x_2
- y_2: x_2, x_1, x_3

- y_3: x_1, x_3, x_2

In that example, we first examined the matching μ given by $x_1 - y_1, x_2 - y_2, x_3 - y_3$. It was shown above that μ was not stable, as x_3 would prefer to be matched with itself. It was unwieldy to apply the definition of a stable matching, though, in treating x_3 as its own spouse. Instead, we show that μ is not in the core of this game. The argument is conceptually the same, but using a coalition makes things much cleaner.

In order to show that μ is not in the core of this game, it suffices to construct a blocking coalition. Consider the coalition $\{x_3\}$. The only other feasible allocation for x_3 is to retain its initial endowment, which means that x_3 will instead opt to remain single. Observe that x_3 prefers to remain single than to be paired with y_3. Thus, μ is not in the core, as a blocking coalition exists. It follows that μ is not stable.

Now consider the matching τ given by $x_1 - y_1, x_2 - y_2, x_3, y_3$. Recall from the example in section II that this matching is stable. Let's now prove that this matching is in the core. Observe that x_1, x_2, y_1 and y_2 all prefer their mates to being single, so these players will not form a one-person coalition. As x_3 and y_3 are already single, they will not form a blocking coalition by themselves. Now consider two-person coalitions. As y_1 and y_2 have their top choices, they will not participate in a blocking coalition. So x_1, x_2 and x_3 can only participate in a blocking coalition with y_3. In all instances, there is no incentive for x_1, x_2 or x_3 to do so. It follows that there are no blocking coalitions, so τ is in the core.

This brings us to proving our result:

Theorem: Let X, Ybe the sets of players satisfying the assumptions of the stable marriage problem. A matching $\mu$is in the core if and only if each matched player prefers his or her mate to being single, and there does not exist a pair $(i, j) \in X \times Y$such that $j \succ^i \mu(i)$and $i \succ^j \mu(j)$.

Proof: Let μbe a matching. Suppose first that $\mu$is in the core. By definition of μbelonging to the core, there is no blocking coalition for μ. Suppose to the contrary that $\mu$is not stable. Then either a matched player would prefer to be single, or there exists an $(i, j) \in X \times Y$such that $j \succ^i \mu(i)$and $i \succ^j \mu(j)$. If a matched player would prefer to be single, then said player could form a blocking coalition by himself, contradicting the assumption that $\mu$is in the core. By similar argument, if there was an $(i, j) \in X \times Y$such that $j \succ^i \mu(i)$and $i \succ^j \mu(j)$, then $\{i, j\}$would constitute a blocking coalition, again contradicting the assumption that $\mu$is in the core. It follows that μmust be a stable matching, and so each element of the core is a stable matching.

Conversely, suppose that $\mu$is stable. It will be shown that $\mu$is also in the core. As $\mu$is stable, no player with a mate prefers to be single. Thus, no one-player blocking coalitions exist. Similarly, since $\mu$is stable, there does not exist a pair $(i, j) \in X \times Y$such that $j \succ^i \mu(i)$and $i \succ^j \mu(j)$. It follows that there can be no blocking coalition with more than one player. Thus, no blocking coalitions exist and $\mu$is in the core. Thus, the set of stable matchings is a subset of the core.

It follows that the core is exactly the set of stable matchings. QED.

5 Results on Gale-Shapley Algorithm

The purpose of this section is to analyze the Gale-Shapley algorithm to analyze how well it performs as a mechanism. In other words, this section addresses the economic issues associated with the Gale-Shapley algorithm instead of the computer science issues. Two primary questions will be addressed. The first question asks how well the players perform under the Gale-Shapley algorithm. The second question asks if there is room to falsely report one's preferences. In other words, does the Gale-Shapley algorithm incentivize honesty? If reporting one's preferences honestly is a weakly dominant strategy for all players, then the mechanism is called strategy proof. We will examine whether or not the Gale-Shapley algorithm is strategy proof.

Recall the example from Section 3, where an example of the Gale-Shapley algorithm was provided. We had the sets $X = \{x_1, x_2, x_3\}$ and $Y = \{y_1, y_2, y_3\}$ with preferences:

- x_1: y_2, y_1

- x_2: y_1, y_2, y_3
- x_3: y_1, y_2
- y_1: x_1, x_3, x_2
- y_2: x_2, x_1, x_3
- y_3: x_1, x_3, x_2

When the X elements were the proposers, we had the matching $x_1 - y_2, x_2 - y_1, x_3, y_3$. Observe that the X elements each obtain their best matches under the Gale-Shapley algorithm, while the Yelements are not so fortunate.

What happens instead if the Y elements were the proposers? Under the Gale-Shapley algorithm, the stable matching would be $x_1 - y_1, x_2 - y_2, x_3, y_3$. Observe that this favors the Y elements, but not the X elements.

In fact, for any preference profile, the Gale-Shapley algorithm provides the proposers with their best core allocation and the acceptors with their worst core allocation. These statements will be formally proven.

Theorem 2: For any preference profile and any ordering of the proposers, the Gale-Shapley algorithm returns the same stable matching μ. In this stable matching, each proposer has his best possible mate amongst all stable matchings.

Proof: This theorem will be proven by contradiction. Let $\sigma \in S_X$, where S_X is the Symmetry group or group of permutations of the proposing set X. Let μ be the matching returned by the Gale-Shapley algorithm when evaluating proposers in the order $x_{\sigma(1)}, x_{\sigma(2)}, ..., x_{\sigma(n)}$. Now let μ' be a stable matching, and let $x_i \in X$ such that $\mu'(x_i) \succ^i \mu(x_i)$. By the Gale-Shapley algorithm, x_i proposed to $\mu'(x_i)$, who in turn rejected x_i. Suppose that, without loss of generality, this is the first instance of an acceptor rejecting a stable partner. As μ' is a stable matching, $\mu'(x_i)$ prefers to be matched with x_i over being single. So $\mu'(x_i)$ must have been already matched with another partner, $x_j \in X$. Since no stable partner has been previously rejected by any acceptor, it follows that x_j can have no stable partner better than $\mu'(x_i)$. So $\mu'(x_i) \succ^j \mu'(x_j)$. Thus, $\{x_j, \mu'(x_i)\}$ form a coalition blocking μ', contradicting the stability of μ'.

As the ordering of the proposers was arbitrary, it follows that for every $\sigma \in S_X$, the same stable matching μ is produced by the Gale-Shapley algorithm, and that $\mu$is the best stable matching for the proposers. QED.

Theorem 2 yields an important corollary, that the proposers have no incentive to manipulate the outcome of the Gale-Shapley algorithm.

Corollary: It is a weakly dominant strategy for proposers to honestly reveal their preferences during the execution of the Gale-Shapley algorithm.

Proof: From Theorem 2, the stable matching μ returned by the Gale-Shapley algorithm is the best possible core allocation for the proposers. Therefore, no proposer can misrepresent his preferences and improve his outcome. QED.

Next, it will be shown that the acceptors receive their worst core allocation under the Gale-Shapley algorithm. To prove this, a more general result will be shown.

Theorem 3: Let μ be a proposer-optimal stable matching. Then μ grants each acceptor her worst possible mate amongst all core allocations.

Proof: This theorem will be proven by contradiction. Suppose there there exists an acceptor that does not have her worst choice in μ. Suppose there exists a second stable matching μ' in which an acceptor prefers her mate in $latex\mu$ to her mate in μ'. Let $y_i \in Y$ such that $\mu(y_i) \succ^i \mu'(y_i)$. As μ is proposer-optimal, $\{y_i, \mu(y_i)\}$

blocks μ', contradicting the stability of μ'. QED.

So we have some powerful results. The Gale-Shapley algorithm correctly produces a stable marriage for any preference profile; and in fact, it produces the same stable marriage for any initial ordering of the proposers. This mechanism also favors the proposers to the point where it is in their best interests to honestly reveal their preferences, while the acceptors don't fare as well.

The last important result deals with whether or not the Gale-Shapley algorithm is a strategy-proof mechanism. That is, can some players misrepresent their preferences to improve their outcomes? The answer is yes. In fact, there is a stronger result- no incentive compatible, strategy-proof mechanism exists for the stable marriage problem. That is, no mechanism exists where every player benefits from honestly revealing their preferences. The proof is actually quite simple.

Theorem 4: No incentive compatible, strategy-proof mechanism exists for the stable marriage problem.

Proof: A counter-example suffices as proof. Consider sets $X = \{x_1, x_2\}$and $Y = \{y_1, y_2\}$ with preferences:

- x_1: y_1, y_2
- x_2: y_2, y_1
- y_1: x_2, x_1
- y_2: x_1, x_2

The only two stable matchings are μgiven by $x_1 - y_1, x_2 - y_2$, and τ given by $x_1 - y_2, x_2 - y_1$. Suppose that players of type X are proposers. Then player y_1 can misrepresent her preferences by indicating she prefers to be single rather than mated with player x_2. This will induce the matching τ with μ believed not to be stable. Similarly, if the players of type Y are proposing, player x_1 can misrepresent his preferences, indicating he will only pair with y_1. This will induce the matching μ, the only stable marriage believed to be stable. Thus, no incentive compatible, strategy-proof mechanism exists for this instance of the stable marriage problem. It follows that no incentive compatible, strategy proof mechanism exists for the general problem. QED.