Subramanian's stable matching algorithm

Yuval Filmus

November 2011

Abstract

Gale and Shapley introduced the well-known stable matching problem in 1962, giving an algorithm for the problem. Subramanian later showed that the problem can be solved by comparator circuits. Lê, Cook and Ye simplified his construction.

We present an algorithm interpolating (in some sense) between the two. Our algorithm is a symmetrized "lazy" version of Gale-Shapley, while Subramanian's algorithm in turn can be viewed as a lazy version of our algorithm. Subramanian's algorithm employs three-valued logic, a device which arises more naturally in the setting of our algorithm.

1 Stable matching problem

An instance of the stable matching problem consists of a set M of n men and a set W of n women. In addition, each person $p \in M \cup W$ has an order of preference over persons of the opposite sex:

$$p\colon \pi_1(p) \succ \pi_2(p) \succ \cdots \succ \pi_n(p).$$

Person p prefers $\pi_1(p)$ the most, and $\pi_n(p)$ the least. If $i > j$ then p prefers $\pi_i(p)$ over $\pi_j(p)$, in symbols $\pi_i(p) \succ_p \pi_j(p)$. The person most preferred by p among a list P is denoted $\max_p P$. The one least preferred is $\min_p P$. Beware that in terms of indices $\pi_i(p)$, $\max_p P$ is the person with lowest index!

A *matching* S is a list of pairs $(m, w) \in M \times W$ such that each man is matched with exactly one woman. We will denote the match of a person p by $S(p)$.

A pair $(m, w) \notin S$ is *unstable* if $w \succ_m S(m)$ and $m \succ_w S(w)$. In words, an unmatched pair (m, w) is unstable if both parties prefer the other person over their current partner. A matching is *stable* if there is no unstable pair.

Gale and Shapley [1] have shown that a stable matching always exists, by giving an algorithm computing it. Their algorithm constructs a stable matching S with the additional property that for each man m, $S(m) \succeq_m T(m)$ for *any* stable matching T. In words, in S every man is matched with the best woman he could ever get in a stable matching. This matching is called the *man-optimal* stable matching. The same algorithm also shows the existence of the *woman-optimal* stable matching.

2 Gale-Shapley algorithm

The Gale-Shapley algorithm proceeds in rounds. In the first round, each man proposes to his top woman, and each woman selects her most preferred suitor. In each subsequent round, each rejected man proposes to his next choice, and each woman selects her most preferred suitor (including her choice from the previous round). The situation eventually stabilizes, and the woman's choices are finalized. They form the man-optimal stable matching.

We can keep track of the workings of the algorithm using a bipartite graph, which lists for each person their potential spouses. Algorithm 1 follows this viewpoint.

Algorithm 1 Gale-Shapley

$G \leftarrow M \times W$
repeat
  $\text{top}(m) \leftarrow \max_m\{w : (m, w) \in G\}$ for all $m \in M$
  $\text{best}(w) \leftarrow \max_w\{m : \text{top}(m) = w\}$ for all $w \in W$ $(\max_w \emptyset = \bot)$
  $G \leftarrow \{(m, w) : m \succeq_w \text{best}(w)\}$ $(m \succ_w \bot$ for all $m \in M)$
until $\text{best}(w)$ is defined for all $w \in W$
return $\{(\text{best}(w), w) : w \in W\}$

It is straightforward to prove by induction that $\text{top}(m)$ can only decrease, $\text{best}(w)$ can only increase, and the graph G keeps shedding edges. Moreover, induction shows that each stable matching is a subgraph of G: indeed, if $m \prec_w \text{best}(w)$ then, in any matching where m is matched to w, $(\text{best}(w), w)$ is an unstable pair. At the end, each woman w is matched to a man m such that $\text{top}(m) = w$. It follows that the matching is the man-optimal stable matching.

Why does the algorithm terminate? Say that a man m and a woman w are *engaged* if $m = \text{best}(w)$ (and so $w = \text{top}(m)$). Suppose that at some iteration, the women in W_f are unengaged. Engaged couples come in pairs, so some man m is unengaged. Either $\text{top}(m) \in W_f$, in which case $\text{top}(m)$ will become engaged in the next round, or $\text{top}(m)$ is engaged to some m'. In the latter case, the two men will compete in the next round, resulting in an update of the graph. So G keeps reducing until all women get engaged.

At the end of the algorithm, it is clear that if $m = \text{best}(w)$ then m is the only man with that property, and so $m = \min_w\{m : (m, w) \in G\}$. So not only is the resulting stable matching man-optimal, but also woman-pessimal.

3 Interval algorithm

The Gale-Shapley algorithm has one disadvantage: it finds only the man-optimal stable matching. Algorithm 2 finds both the man-optimal and the woman-optimal stable matchings at the same time.

Considering Algorithm 2, if we change the quantification of p, q to $p \in M$ and $q \in Q$, then we get an algorithm very similar to Gale-Shapley. The undirected

Algorithm 2 Interval algorithm

$I_0(m) \leftarrow W$ for all $m \in M$
$I_0(w) \leftarrow M$ for all $w \in W$
$t \leftarrow 0$
repeat
  $\mathrm{top}_t(p) = \max_p I_t(p)$ for all $p \in M \cup W$
  $\mathrm{best}_t(q) = \max_q \{p : q = \mathrm{top}_t(p)\}$ for all $q \in M \cup W$
  $I_{t+1}(q) \leftarrow \{p \in I_t(q) : p \succeq_q \mathrm{best}_t(q)\}$ for all $q \in M \cup W$
  $I_{t+1}(p) \leftarrow I_{t+1}(p) \setminus \mathrm{top}_t(p)$ for all $p \in M \cup W$ such that $p \notin I_t(\mathrm{top}_t(p))$
  $t \leftarrow t + 1$
until $I_t = I_{t-1}$
return $\{(m, \max_m I_t(m)) : m \in M\}$, $\{(\max_w I_t(w), w) : w \in W\}$

graph G is replaced by a directed graph represented by sets of neighbors $I(p)$. When a woman removes a man from her neighborhood, he is not immediately informed of the fact; he will eventually find out, when she becomes his top choice. The stopping condition is also different, but the proof shows that when the situation stabilizes, all women are engaged.

In contrast to Gale-Shapley, Algorithm 2 is symmetric in both parties. The analysis in the previous paragraph should convince the reader that it returns both the man-optimal and the woman-optimal stable matchings. The rules updating $I_t(p)$ ensure that these always remain intervals within p's preference order, hence the name *Interval algorithm.*

4 Three-valued logic

There is a curious connection between Algorithm 2 and three-valued logic. In three-valued logic, apart from the usual definite boolean values $0, 1$ (which we think of as singletons $\{0\}$,$\{1\}$), we have a further *indefinite* value $* = \{0, 1\}$. The boolean operators $\neg, \wedge, \vee$ are generalized in the natural way:

$$\neg x = \{\neg b : b \in x\}, \quad x \wedge y = \{b \wedge c : b \in x, c \in y\}, \quad x \vee y = \{b \vee c : b \in x, c \in y\}.$$

So we have $\neg * = *$, $0 \vee * = *$, $1 \vee * = 1$, $* \vee * = *$, and so on.

Our aim is to reformulate Algorithm 2 using three-valued logic. The sets $I_t(p)$ will be replaced by two matrices $\mathfrak{M}_t(m, w)$, $\mathfrak{W}_t(m, w)$, defined according to the following rules ($m \in M$, $w \in W$):

$$\mathfrak{M}_t(m, w) = \begin{cases} 1 & w \succeq_m \max_m I_t(m) \\ * & \max_m I_t(m) \succ_m w \succeq_m \min_m I_t(m) \\ 0 & \min_m I_t(m) \succ_m w \end{cases}$$

$$\mathfrak{W}_t(w, m) = \begin{cases} 0 & m \succeq_w \max_w I_t(w) \\ * & \max_w I_t(w) \succ_w m \succeq_w \min_w I_t(w) \\ 1 & \min_w I_t(w) \succ_w m \end{cases}$$

Since the intervals $I_t(p)$ always shrink during the course of Algorithm 2, once a value in $\mathfrak{M}$ or $\mathfrak{W}$ becomes definite, it never changes. When does a value become definite? Suppose that $\mathfrak{M}_t(m, \pi_i(m)) = *$. The value can change to 1 only if $w \triangleq \pi_{i-1}(m) = \max_m I_t(m)$ is chopped. That, in turn, happens if $m \notin I_t(w)$. In fact, we must have $m \prec_w \min_w I_t(w)$, for otherwise (m, w) would be an unstable pair in any stable matching produced by the algorithm. So $\mathfrak{M}_t(m, w) = \mathfrak{W}_t(w, m) = 1$. Our reasoning makes it clear that this condition is not only necessary but also sufficient.

Consider the other case: given that $\mathfrak{M}_t(m, \pi_i(m)) = *$, when does it change to 0? That happens when $w \triangleq \text{best}_t(m) \succ_m \pi_i(m)$. Since $m = \text{top}_t(w)$, we see that $\mathfrak{W}_t(w, m) = 0$. So if $\mathfrak{M}_{t+1}(m, \pi_i(m)) = 0$, we must have $\mathfrak{W}_t(w, m) = 0$ for some $w \succ_m \pi_i(m)$. Conversely, suppose that $\mathfrak{W}_t(w, m) = 0$ for some $w \succ_m \pi_i(m)$. If $m \succ_w \text{top}_t(w)$ then $m = \text{top}_s(w)$ at some earlier point in time $s < t$. At that point, $\text{best}_s(m) \succeq_m w \succ_m \pi_i(m)$, and so $\pi_i(m) \notin I_{s+1}(m)$, contradicting our assumption $\mathfrak{M}_t(m, \pi_i(m)) = *$. Hence $m = \text{top}_t(w)$ after all, and so $\mathfrak{M}_{t+1}(m, \pi_i(m)) = 0$. We have completed the proof of the following lemma.

Lemma 1. *Suppose* $\mathfrak{M}_t(m, \pi_{i+1}(m)) = *$. *Then* $\mathfrak{M}_{t+1}(m, \pi_{i+1}(m)) = 1$ *if and only if*

$$\mathfrak{M}_t(m, \pi_i(m)) = \mathfrak{W}_t(\pi_i(m), m) = 1,$$

and $\mathfrak{M}_{t+1}(m, \pi_{i+1}(m)) = 0$ *if and only if*

$$\mathfrak{W}_t(\pi_j(m), m) = 0 \text{ for some } j \le i.$$

Combining the two conditions together, we obtain a recurrence relation for the array $\mathfrak{M}$.

Lemma 2. *For all* t, $\mathfrak{M}_t(m, \pi_1(m)) = 1$. *At the beginning of the algorithm,* $\mathfrak{M}_0(m, w) = *$ *for* $w \neq \pi_1(m)$, *and in general*

$$\mathfrak{M}_{t+1}(m, \pi_{i+1}(m)) = \mathfrak{M}_t(m, \pi_i(m)) \wedge \bigwedge_{j \le i} \mathfrak{W}_t(\pi_j(m), m).$$

Proof. The first step is to relate Lemma 1 to the recurrence equation. We do this by (implicit) induction on t. Suppose that $\mathfrak{M}_t(m, \pi_{i+1}(m)) = *$, and let

$$R = \mathfrak{M}_t(m, \pi_i(m)) \wedge \bigwedge_{j \le i} \mathfrak{W}_t(\pi_j(m), m).$$

If $R = 1$, then the lemma implies that $\mathfrak{M}_{t+1}(m, \pi_{i+1}(m)) = 1$. Conversely, if $\mathfrak{M}_{t+1}(m, \pi_{i+1}(m)) = 1$ then the lemma implies that $\mathfrak{M}_t(m, \pi_i(m)) = \mathfrak{W}_t(\pi_i(m), m) = 1$. If $i > 1$, then the recurrence equation for $\mathfrak{M}_t(m, \pi_i(m))$ shows that $\mathfrak{W}_{t-1}(\pi_j(m), m) = 1$ for $j \le i - 1$. As mentioned above, once a value of $\mathfrak{W}$ becomes definite it remains constant, and so $R = 1$.

If $\mathfrak{M}_{t+1}(m, \pi_{i+1}(m)) = 0$, then the lemma shows that $R = 0$. Conversely, if $R = 0$, then either $\mathfrak{M}_t(m, \pi_i(m)) = 0$, or $\mathfrak{W}_t(\pi_j(m), m) = 0$ for some $j \le i$. In

the first case, the definition of $\mathfrak{M}$ directly implies that $\mathfrak{M}_{t+1}(m, \pi_{i+1}(m)) = 0$. In the latter case, we reach the same conclusion by way of the lemma.

Summarizing, the recurrence equation holds in case $\mathfrak{M}_t(m, \pi_{i+1}(m)) = *$. If $\mathfrak{M}_t(m, \pi_{i+1}(m))$ is boolean, then either all values in R are equal to 1, or at least one of them is equal to 0. Since boolean values in the arrays remain constant, the correctness of the recurrence equation for $t + 1$ follows directly from its correctness for t. □

The lemma allows us to reformulate Algorithm 2 as a three-valued logic algorithm, Algorithm 3.

Algorithm 3 Interval algorithm, three-valued logic formulation

$$\mathfrak{M}_0(m, w) = \begin{cases} 1 & w = \pi_1(m) \\ * & \text{otherwise} \end{cases}$$

$$\mathfrak{W}_0(w, m) = \begin{cases} 0 & m = \pi_1(w) \\ * & \text{otherwise} \end{cases}$$

$t \leftarrow 0$

repeat

$$\mathfrak{M}_{t+1}(m, \pi_i(m)) = \begin{cases} 1 & i = 1 \\ \mathfrak{M}_t(m, \pi_{i-1}(m)) \wedge \bigwedge_{j \le i-1} \mathfrak{W}_t(\pi_j(m), m) & \text{otherwise} \end{cases}$$

$$\mathfrak{W}_{t+1}(w, \pi_i(w)) = \begin{cases} 0 & i = 1 \\ \mathfrak{W}_t(w, \pi_{i-1}(w)) \vee \bigvee_{j \le i-1} \mathfrak{M}_t(\pi_j(w), w) & \text{otherwise} \end{cases}$$

$t \leftarrow t + 1$

until $\mathfrak{M}_t = \mathfrak{M}_{t-1}$ and $\mathfrak{W}_t = \mathfrak{W}_{t-1}$

$S_M \leftarrow \{(m, w) : \mathfrak{M}_t(m, w) = 1 \text{ and } \mathfrak{W}_t(w, m) \in \{0, *\}\}$

$S_W \leftarrow \{(m, w) : \mathfrak{W}_t(w, m) = 0 \text{ and } \mathfrak{M}_t(m, w) \in \{1, *\}\}$

return S_M, S_W

The lemma shows that at the end of Algorithm 3, $\mathfrak{M}_t$ and $\mathfrak{W}_t$ correspond to I_t at the end of Algorithm 2. The latter algorithm returns the man-optimal stable matching

$$T_M = \{(m, \max_m I_t(m)) : m \in M\} = \{(\min_w I_t(w), w) : w \in W\}.$$

The definitions of $\mathfrak{M}_t, \mathfrak{W}_t$ makes it plain that $S_M \subseteq T_M$. Conversely, suppose that $\mathfrak{M}_t(m, w) = 1$ and $\mathfrak{W}_t(w, m) \in \{0, *\}$. In terms of intervals, $w \succeq_m \max_m I_t(m)$ and $m \succeq_w \min_w I_t(w)$. So $w = \max_m I_t(m)$ and $m = \min_w I_t(w)$, for otherwise (m, w) would be an unstable pair in T_M. We conclude that $S_M = T_M$ is the man-optimal stable matching.

5 Subramanian's algorithm

Subramanian's paper [3] is concerned with comparator circuits. These can be thought of as programs in a language with boolean variables and a single in-

struction, known as *comparison*:

$$(x, y) \leftarrow (x \wedge y, x \vee y).$$

For the sake of computation, we are additionally allowed to initialize some of the variables by constants.

Lê, Cook and Ye [2] show that if we enlarge the language by allowing three-valued variables and by adding a negation instruction $x \leftarrow \neg x$, then the resulting program can be simulated by a comparator circuit; this includes a final step in which each $*$ is replaced by a predetermined value. For more information on comparator circuits, consult their paper.

Comparator circuits simulate boolean formulas, but they are conjectured not to simulate general circuits. The problem is with gates whose fan-out is larger than 1: once a variable is compared against another variable, its original value is irretrievably lost. Subramanian's goal was to come up with a comparator circuit for the stable matching problem. Algorithm 3 cannot be used for this purpose as it stands, since a value $\mathfrak{W}_t(\pi_j(m), m)$ is compared against $\mathfrak{M}_t(m, \pi_i(m))$ for all $i \geq j$.

The main result of Subramanian's paper is a comparator circuit algorithm for stable matching. He presents his algorithm in terms of X-networks; our presentation follows the one by Lê et al.

Looking at Lemma 1, we see that we only need to consider several values of $\mathfrak{W}_t(\cdot, m)$ when checking whether an entry $\mathfrak{M}_{t+1}(m, \pi_i(m))$ changes to zero. What if we only looked at $\mathfrak{W}_t(\pi_i(m), m)$? A zero in $\mathfrak{W}_t(\pi_j(m), m)$ for $j < i$ will change $\mathfrak{M}_{t+1}(m, \pi_{j+1}(m))$ to zero, and the zero will propagate in $i - j$ steps to $\mathfrak{M}(m, \pi_{i+1}(m))$. This prompts Algorithm 4, which is realizable as a comparator circuit.

In order to convince ourselves that Algorithm 4 is valid, we show that, in some sense, Algorithms 3 and 4 simulate each other. Since we will be considering both algorithms at the same time, it will be convenient to use $\mathfrak{M}^I, \mathfrak{W}^I$ for Algorithm 3 and $\mathfrak{M}^S, \mathfrak{W}^S$ for Algorithm 4.

Lemma 3. *If $\mathfrak{M}^I(m, w) = b \in \{0, 1\}$ at the end of Algorithm 3, then $\mathfrak{M}^S(m, w) = b$ at the end of Algorithm 4. The same is true for $\mathfrak{W}$.*

Proof. The proof is by induction on the time t in which an entry becomes definite in Algorithm 3. We only consider $\mathfrak{M}$, the proof for $\mathfrak{W}$ being dual. The lemma is trivial for entries which become definite during the initialization phase. Suppose an entry $\mathfrak{M}^I_{t+1}(m, \pi_{i+1}(m)) = b$ becomes definite at time $t + 1$.

If $b = 1$ then $\mathfrak{M}^I_t(m, \pi_i(m)) = \mathfrak{W}^I_t(\pi_i(m), m) = 1$, and so the induction hypothesis implies that at some time s, $\mathfrak{M}^S_s(m, \pi_i(m)) = \mathfrak{W}^S_s(\pi_i(m), m) = 1$. At the following step, $\mathfrak{M}^S_{s+1}(m, \pi_{i+1}(m)) = 1$.

If $b = 0$, then either $\mathfrak{M}^I_t(m, \pi_i(m)) = 0$, or $\mathfrak{W}^I_t(\pi_j(m), m) = 0$ for some $j \leq i$. In the first case, argue as before. In the second, the induction hypothesis implies that at some time s, $\mathfrak{W}^S_s(\pi_j(m), m) = 0$. At the following step, $\mathfrak{M}^S_{s+1}(m, \pi_{j+1}(m)) = 0$, and $i - j$ steps later, $\mathfrak{M}^S_{s+1+i-j}(m, \pi_{i+1}(m)) = 0$. □

Algorithm 4 Subramanian's algorithm

$$\mathfrak{M}_0(m,w) = \begin{cases} 1 & w = \pi_1(m) \\ * & \text{otherwise} \end{cases}$$

$$\mathfrak{W}_0(w,m) = \begin{cases} 0 & m = \pi_1(w) \\ * & \text{otherwise} \end{cases}$$

$t \leftarrow 0$

repeat

$$\mathfrak{M}_{t+1}(m,\pi_i(m)) = \begin{cases} 1 & i = 1 \\ \mathfrak{M}_t(m,\pi_{i-1}(m)) \wedge \mathfrak{W}_t(\pi_{i-1}(m),m) & \text{otherwise} \end{cases}$$

$$\mathfrak{W}_{t+1}(w,\pi_i(w)) = \begin{cases} 0 & i = 1 \\ \mathfrak{W}_t(w,\pi_{i-1}(w)) \vee \mathfrak{M}_t(\pi_{i-1}(w),w) & \text{otherwise} \end{cases}$$

$t \leftarrow t+1$

until $\mathfrak{M}_t = \mathfrak{M}_{t-1}$ and $\mathfrak{W}_t = \mathfrak{W}_{t-1}$

$S_M \leftarrow \{(m,w) : \mathfrak{M}_t(m,w) = 1 \text{ and } \mathfrak{W}_t(w,m) \in \{0,*\}\}$

$S_W \leftarrow \{(m,w) : \mathfrak{W}_t(w,m) = 0 \text{ and } \mathfrak{M}_t(m,w) \in \{1,*\}\}$

return S_M, S_W

Lemma 4. *If $\mathfrak{M}^S(m,w) = b \in \{0,1\}$ at the end of Algorithm 4, then $\mathfrak{M}^I(m,w) = b$ at the end of Algorithm 3. The same is true for $\mathfrak{W}$.*

Proof. The proof is by induction on the time t in which an entry becomes definite in Algorithm 4. We only consider $\mathfrak{M}$, the proof for $\mathfrak{W}$ being dual. The lemma is trivial for entries which become definite during the initialization phase. Suppose an entry $\mathfrak{M}^S_{t+1}(m,\pi_{i+1}(m)) = b$ becomes definite at time $t+1$.

If $b = 0$ then either $\mathfrak{M}^S_t(m,\pi_i(m)) = 0$ or $\mathfrak{W}^S_t(\pi_i(m),m) = 0$. In both cases, the induction hypothesis implies that the same is true at some time s for $\mathfrak{M}^I_s(m,\pi_i(m))$ or $\mathfrak{W}^I_s(\pi_i(m),m)$, and so $\mathfrak{M}^I_{s+1}(m,\pi_{i+1}(m)) = 0$.

If $b = 1$, then $\mathfrak{M}^S_t(m,\pi_i(m)) = \mathfrak{W}^S_t(\pi_i(m),m) = 1$. The induction hypothesis implies that at some time s, $\mathfrak{M}^I_s(m,\pi_i(m)) = \mathfrak{W}^I_s(\pi_i(m),m) = 1$. The recurrence equation for $\mathfrak{M}^I_s(m,\pi_i(m))$ in Algorithm 3 implies that $\mathfrak{W}^I_s(\pi_j(m),m) = \mathfrak{W}^I_{s-1}(\pi_j(m),m) = 1$ for all $j < i$, and so $\mathfrak{M}^I_{s+1}(m,\pi_{i+1}(m)) = 1$. □

The two lemmas together imply that at the end of both algorithms, $\mathfrak{M}^I = \mathfrak{M}^S$ and $\mathfrak{W}^I = \mathfrak{W}^S$. So Subramanian's algorithm correctly computes the man-optimal and woman-optimal stable matchings. Each value of $\mathfrak{M}_t, \mathfrak{W}_t$ is used twice in the computation of $\mathfrak{M}_{t+1}, \mathfrak{W}_{t+1}$, once as an input to an $\wedge$ gate and once as an input to an $\vee$ gate. In both cases, it is compared against the same value: indeed $\mathfrak{M}_t(m,w)$ is always compared against $\mathfrak{W}_t(w,m)$. This shows that the algorithm can be implemented as a comparator circuit, if we replace the repeat-until loop with $2n^2$ fixed iterations.

References

[1] David Gale and Lloyd S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[2] Dai Tri Man Lê, Stephen Cook, and Yuli Ye. A formal theory for the complexity class associated with the stable marriage problem. In *Computer Science Logic (CSL'11)*, 2011.

[3] Ashok Subramanian. A new approach to stable matching problems. *SIAM Journal of Computing*, 23:671–701, August 1994.