

An Efficient Algorithm for Batch Stability Testing

John Dabney
School of Computing
Clemson University
jdabney@cs.clemson.edu

Brian C. Dean
School of Computing
Clemson University
bcdean@cs.clemson.edu

April 25, 2009

Abstract

Given a stable marriage problem instance represented by a bipartite graph having $2n$ vertices and m edges, we describe an algorithm that can verify the stability of k different matchings in a batch fashion in $O((m + kn) \log^2 n)$ time. This affirmatively answers a longstanding open question of Gusfield and Irving as to whether stability can be verified in a batch setting (after sufficient preprocessing) in time sub-quadratic in n .

1 Introduction

For many computational problems, it seems easier from an algorithmic standpoint to verify the correctness or optimality of a candidate solution than to solve the problem outright. Even if this is not possible, it is sometimes possible to verify a *batch* of candidate solutions more efficiently than by checking each one individually, often by preprocessing an instance of the problem to build a data structure that supports fast queries to verify candidate solutions.

We show in this paper how to perform efficient batch testing for stability of a set of matchings in the well-studied stable marriage problem. Given a stable marriage problem instance represented by a bipartite graph having $2n$ vertices and m edges, we describe an algorithm that can verify the stability of k different matchings in a batch fashion in $O((m + kn) \log^2 n)$ time, effectively verifying each single matching in $O(n \log^2 n)$ amortized time after $O(m \log^2 n)$ preprocessing time. This affirmatively answers a longstanding open question from the book of Gusfield and Irving on stable marriage problems [4] as to whether stability can be verified in a batch setting (after sufficient preprocessing) in time sub-quadratic in n ; this question is motivated by existence of an $\Omega(n^2)$ lower bound on the worst-case running time for verifying stability of a single matching [9].

Our results are built upon a new characterization of stable matchings in terms the of connectivity a certain graph related to the underlying set of “rotations” induced by the stable matching instance. Using this characterization, we then apply efficient data structures for fully dynamic connectivity in graphs [5, 10] to obtain an efficient test for stability.

In the next section, we describe the stable marriage problem in detail and summarize its relevant mathematical properties. Following this, we introduce our new connectivity-based characterization

of stability, upon which we build our data structure for verifying the stability of a series of candidate matchings.

2 Preliminaries

The classical stable marriage problem was introduced in 1962 by Gale and Shapley [3]. Its input consists of a set M of men and a set W of women, where $|M| = |W|$. Each man $i \in M$ submits a ranked preference list over all the women, and each woman $j \in W$ submits a ranked preference list over all the men (we use variables i and j instead of m and w to avoid any potential confusion with the m edges in our instance). Our goal is to find a matching \mathcal{M} between M and W that is *stable*, meaning that it admits no *blocking pair* — a pair $(i, j) \notin \mathcal{M}$ for which i and j would both prefer being matched with each other over the way they are assigned in \mathcal{M} (where an individual always prefers being matched over being single). Note that every stable matching must be perfect (pairing up every man and woman), since otherwise we would have an unmatched man i and woman j who together would form a blocking pair. Gale and Shapley proved constructively that a stable matching always exists by giving a simple $O(n^2)$ time algorithm for finding such a matching, where $n = |M| = |W|$; observe that this running time is linear in the size of the input.

In this paper, we focus on a well-studied generalization of the stable marriage problem allowing for incomplete preference lists. Here, our input is described by a bipartite graph $G = (M \cup W, E)$, with $2n$ vertices and $m = |E|$ edges. Each man $i \in M$ submits a ranked preference list over the set of women $\{j \mid (i, j) \in E\}$ to which he is adjacent, and likewise for the women. The definition of stability remains the same, except we only consider couples $(i, j) \in E$ as potential blocking pairs. In this setting, we may not be able to find a stable matching $\mathcal{M} \subseteq E$ for which all men and women are assigned. However, since our results already include an allowance for preprocessing, we assume that we have “cleaned up” our instance by eliminating any men and women who are not assigned in any stable matching. Furthermore, we assume that we have removed any edges from E that are not included in any stable matching. Both steps are easily accomplished in linear time ($O(m)$ time). The reader is referred to [4] for further details. Any matching we later attempt to verify that involves one of the vertices or edges removed during this preprocessing step can be immediately classified as unstable.

Rotations. As with many algorithmic results for the stable matching problems (e.g., [6, 7, 8]), our methods make use of an alternative characterization of the set of all stable matchings in terms of *rotations*, cyclic augmenting structures that allow us to move from one stable matching to another. For our purposes, we define a rotation $\pi = (\pi_M, \pi_W)$ as a set of men $\pi_M \subseteq M$ and a set of women $\pi_W \subseteq W$. We say that a rotation π is *exposed* in a stable matching \mathcal{M} if we obtain another stable matching by advancing the assignments in π_M and retreating those in π_W , where by “advancing” we mean assigning each man $i \in \pi_M$ to the entry in his preference list immediately following his partner in \mathcal{M} , and by “retreating” we mean assigning each woman $j \in \pi_W$ to the entry in her preference list immediately preceding her partner in \mathcal{M} . The process of moving from \mathcal{M} to another stable matching by advancing π_M and retreating π_W is known as *eliminating* π from \mathcal{M} . Note that our definition of a rotation is somewhat simpler than the standard definition of a rotation in the literature, owing to the fact that we have already pruned away useless entries from our preference lists. For a more detailed introduction to rotations and all of their properties we introduce in this section, see [6, 7, 8].

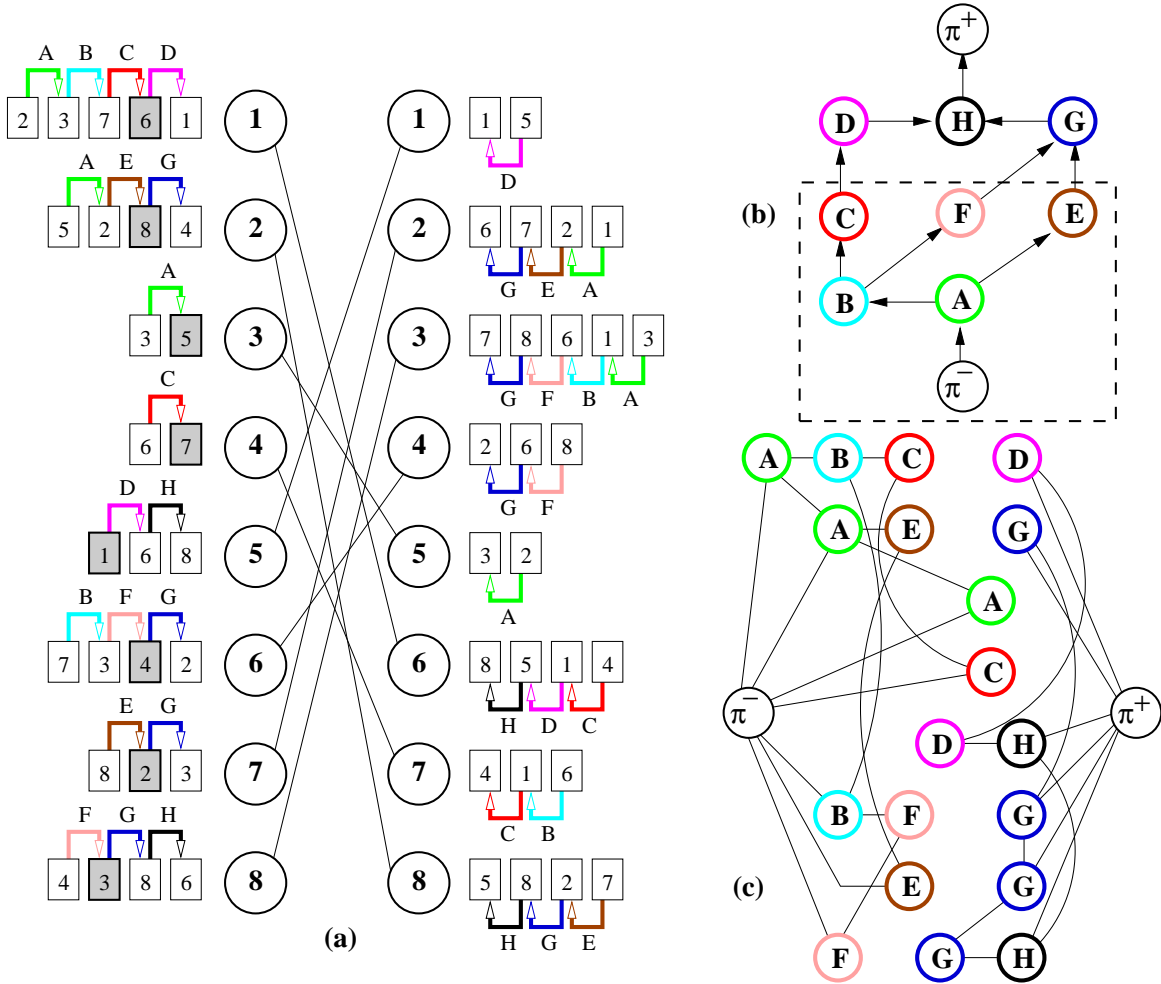


Figure 1: An example instance of the stable matching problem. In (a), the men and their preference lists are shown on the left of the bipartite matching graph G , and the women on the right. The eight rotations (labeled $A \dots H$) are shown overlaid on the preference lists, and the reduced rotation DAG is shown in (b) to the right of the matching instance. A stable matching \mathcal{M} is indicated with shaded entries in the men’s preference lists, and its corresponding closure is outlined in the rotation DAG. The expanded rotation graph H' is shown in (c). Note that the removal of the edges $P(i, j)$ corresponding to $(i, j) \in \mathcal{M}$ leaves H' with two connected components, one containing π^- and the other containing π^+ .

Every stable matching instance without a single unique solution admits two special stable solutions, a “man-optimal, woman-pessimal” matching \mathcal{M}^m where every man is assigned his first choice and every woman her last choice, and a “woman-optimal, man-pessimal” matching \mathcal{M}^w where every woman is assigned her first choice and every man his last choice. Suppose we start with \mathcal{M}^m and repeatedly move to a new stable matching by eliminating an arbitrary exposed rotation, stopping when we reach \mathcal{M}^w (note that every stable matching except \mathcal{M}^w exposes some rotation). The sequence of rotations we apply during this process is called a rotation elimination ordering. Remarkably, it turns out that every rotation elimination ordering involves the exact same set Π of rotations. Moreover, for each man i and each woman j in i ’s preference list (i.e., for each edge $(i, j) \in E$), there is a unique rotation $\pi^-(i, j) \in \Pi$ whose elimination advances i ’s assignment to

j (unless j is the first entry on i 's preference list), and there is a unique rotation $\pi^+(i, j) \in \Pi$ whose elimination advances i 's assignment past j (unless j is the last entry on i 's preference list). For example, Figure 1(a) shows an example stable matching instance with 8 rotations labeled $A \dots H$, where $\pi^-(1, 6) = C$ and $\pi^+(1, 6) = D$. In order to ensure that $\pi^-(i, j)$ and $\pi^+(i, j)$ are clearly defined for every edge $(i, j) \in E$, we include two dummy rotations π^- and π^+ in Π , with $\pi^-(i, j) = \pi^-$ if j is the first woman on i 's preference list, and $\pi^+(i, j) = \pi^+$ if j is the last woman on i 's preference list.

The set Π of all rotations adheres to a natural partial ordering: for any two rotations $\pi, \pi' \in \Pi$, we say $\pi \prec \pi'$ if π' cannot be exposed until π is eliminated (i.e., if π precedes π' in every possible elimination ordering). Observe from Figure 1(a) that two rotations π and π' share a precedence relationship if and only if $\pi_M \cap \pi'_M \neq \emptyset$, in which case their relative ordering amongst the common preference lists of π_M and π'_M determines whether $\pi \prec \pi'$ or $\pi' \prec \pi$. Also note that $\pi^- \preceq \pi$ and $\pi^+ \succeq \pi$ for all $\pi \in \Pi$. We commonly represent the set of all rotations using a directed acyclic graph (DAG), D , where $V(D) = \Pi$ and $(\pi, \pi') \in E(D)$ if and only if $\pi \prec \pi'$. From an algorithmic standpoint, it typically suffices to omit a large number of edges implied by transitivity and consider a reduced DAG D' with $V(D') = \Pi$ and $(\pi, \pi') \in E(D')$ if and only if $\pi = \pi^-(i, j)$ and $\pi' = \pi^+(i, j)$ for some edge $(i, j) \in E$. An example of such a reduced rotation DAG is shown in Figure 1(b). Letting $n = |M| = |W|$ and $m = |E|$ describe the size of our (pruned) bipartite input graph G , it is easy to see that $|V(D')| = |\Pi| \leq m$ and $|E(D')| = m$. One can show that D is the transitive closure of D' , and one can also compute in $O(m)$ time the values $\pi^-(i, j)$ and $\pi^+(i, j)$ for all $(i, j) \in E$, the structure of all rotations in Π , and the reduced rotation DAG D' [4].

For any subset $S \subseteq \Pi$, let S^* be the set of all rotations π such that $\pi \preceq \pi'$ for some $\pi' \in S$ (i.e., the set of all rotations π from which there exists a directed path in D (equivalently, D') to some rotation $\pi' \in S$). We say S is a *closure* if $S = S^*$. We now arrive at a key structural connection between the rotation DAG D' and stable matchings, a proof of which appears in [7].

Lemma 1. *There is a 1-to-1 correspondence between stable matchings in G and closures S in D' such that $\pi^- \in S$ and $\pi^+ \notin S$.*

The closure $S \subseteq \Pi$ corresponding to a stable matching \mathcal{M} contains precisely the set of rotations that must be applied starting from \mathcal{M}^m , according to some valid elimination ordering (a topological ordering of S), in order to reach \mathcal{M} . The condition $\pi^- \in S$ and $\pi^+ \notin S$ has been added to ensure that \mathcal{M}^m and \mathcal{M}^w both correspond to a distinct closure; this condition holds automatically for the closure corresponding to any other stable matching \mathcal{M} .

3 A Connectivity-Based Characterization

Given an instance of the stable marriage problem, we build its *expanded rotation graph* H (an undirected graph) by including in $V(H)$ a vertex representing π^- , a vertex representing π^+ , and a vertex for every pair (π, i) where $\pi \in \Pi - \{\pi^-, \pi^+\}$ and $i \in \pi_M$. The edge set $E(H)$ is given by $P \cup R$, where $P = \{P(i, j) : (i, j) \in E\}$ is a set of *precedence-related* edges, with edge $P(i, j)$ connecting vertex $(\pi^-(i, j), i)$ to vertex $(\pi^+(i, j), i)$; that is, each edge $P(i, j)$ connects the two rotation instances surrounding the edge (i, j) in i 's preference list, reflecting the precedence relationship between these two rotations. The set R contains *rotation-related* edges, and consists of an arbitrarily-chosen forest in which we have a spanning tree on the vertices originating from

each distinct source rotation $\pi \in \Pi - \{\pi^-, \pi^+\}$ (i.e., the vertices (π, i) with $i \in \pi_M$). Note that $P \cap R = \emptyset$, since each rotation $\pi \in \Pi$ appears at most once in each preference list. One can show that $|V(H)| = m - n + 2 = O(m)$ and $|E(H)| = 2m - n + 2 - |\Pi| = O(m)$, and one can build H in $O(m)$ time by first computing the structure of all rotations in Π .

We now arrive at a new characterization of stability in terms of the connectivity of H :

Lemma 2. *A matching \mathcal{M} is stable if and only if π^- and π^+ lie in different connected components in H , after the edge set $P(\mathcal{M}) = \{P(i, j) : (i, j) \in \mathcal{M}\}$ is deleted.*

Proof. Let H' denote H with edge set $P(\mathcal{M})$ deleted, as shown in Figure 1(c). Since we never delete edges from the forest R , note that each rotation $\pi \in \Pi - \{\pi^-, \pi^+\}$ remains “intact”, with all its corresponding vertices (π, \cdot) in the same connected component of H' . Also note that there is a path of precedence-related edges from π^- to the vertex $(\pi^-(i, j), i)$, and from the vertex $(\pi^+(i, j), i)$ to π^+ , for every $(i, j) \in \mathcal{M}$, which implies that H' either consists of one large connected component, or two connected components, with one containing π^- and the other π^+ .

Suppose \mathcal{M} is stable but that H' consists of just one single component. Let $S \subseteq \Pi$ be the closure corresponding to \mathcal{M} according to Lemma 1. Since $\pi^- \in S$ and $\pi^+ \notin S$, if we follow any path p from π^- to π^+ in H' , we must at some point traverse a precedence-related edge $P(i, j)$ where $\pi^-(i, j) \in S$ and $\pi^+(i, j) \notin S$. But then $(i, j) \in \mathcal{M}$, so $P(i, j)$ would have been removed when constructing H' , a contradiction. Suppose now that H' consists of two components. Here, we obtain a valid closure S by taking all rotations lying in the π^- component, so the corresponding matching \mathcal{M} must be stable. \square

4 An Efficient Batch Algorithm for Verifying Stability

Given the characterization from Lemma 2, let us now build the dynamic connectivity data structure of [5] on H . This structure supports the insertion and deletion of edges in an r -vertex graph in $O(\log^2 r)$ amortized time, and it can determine whether or not an arbitrary pair of vertices lie in the same connected component in $O(\log r / \log \log r)$ time. In our case, since $|V(H)| = O(m)$, we obtain amortized running times of $O(\log^2 m) = O(\log^2 n)$ and $O(\log m) = O(\log n)$ respectively. We can build the initial structure in a batch fashion (without inserting edges one by one) in only $O(m)$ time, although amortization requires that we charge $O(m \log^2 n)$ for this step, giving us a credit for future work to be redeemed later at query time.

To verify the stability of a candidate matching \mathcal{M} , we now simply delete the n edges $P(i, j)$ for all $(i, j) \in \mathcal{M}$ from H in $O(n \log^2 n)$ amortized time and issue a connectivity query to test whether π^- and π^+ lie in the same component. Our matching \mathcal{M} is stable if and only if the answer to this query is negative. We then restore the edges $P(i, j)$ to H in anticipation of future queries. Our time for verifying k queries is therefore $O((m + kn) \log^2 n)$, which is effectively $O(n \log^2 n)$ amortized time per stability query after spending $O(m \log^2 n)$ preprocessing time.

5 Concluding Remarks

Note that we cannot claim an $O(n \log^2 n)$ *worst-case* bound (after preprocessing) for verifying a single matching, since due to amortization, some of the $O(m \log^2 n)$ “preprocessing time” is actually spent during queries instead of during preprocessing. It is an interesting open question to see if one can obtain a reasonable worst-case bound for queries. Presently, the best-known dynamic graph connectivity data structures with worst-case bounds [2, 1] seem insufficient to yield a stability test with worst-case running time sub-quadratic in n .

The running time of our algorithm can be easily improved if one obtains a stronger dynamic connectivity data structure. For example, Thorup [10] uses randomization to obtain an expected amortized update time of $O(\log n (\log \log n)^3)$, which gives an $O((m + kn) \log n (\log \log n)^3)$ total expected running time (whereas our original running time is deterministic). Perhaps a faster time can possibly be obtained by exploiting the special structure of H combined with the fact that we really only need to accommodate edge deletions, rather than deletions plus insertions.

References

- [1] D. Eppstein, Z. Galil, G. Italiano, and A. Nissenzweig. Sparsification – a technique for speeding up dynamic graph algorithms. *Journal of the ACM*, 44(5):669–696, 1997.
- [2] G.N. Frederickson. Data structures for on-line updating of minimum spanning trees, with applications. *SIAM Journal on Computing*, 14(4):781–798, 1985.
- [3] D. Gale and L.S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69(1):9–14, 1962.
- [4] D. Gusfield and R. Irving. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, 1989.
- [5] J. Holm, K. de Lichtenberg, and M. Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *Journal of the ACM*, 48(4):723–760, 2001.
- [6] R. Irving. An efficient algorithm for the “stable room-mates” problem. *Journal of Algorithms*, 6:577–595, 1985.
- [7] R. Irving and P. Leather. The complexity of counting stable marriages. *SIAM Journal on Computing*, 15:655–667, 1986.
- [8] R.W. Irving, P. Leather, and D. Gusfield. An efficient algorithm for the “optimal” stable marriage. *Journal of the ACM*, 34(3):532–543, 1987.
- [9] C. Ng and D.S. Hirschberg. Lower bounds for the stable marriage problem and its variants. *SIAM Journal on Computing*, 19:71–77, 1990.
- [10] M. Thorup. Near-optimal fully-dynamic graph connectivity. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 343–350, 2000.