þÿAlmost Stable Matchings by Truncating the Gale

Floréen, Patrik

2010

Almost Stable Matchings by Truncating the Gale–Shapley Algorithm

Patrik Floréen, Petteri Kaski,
Valentin Polishchuk, and Jukka Suomela*

Abstract

We show that the ratio of matched individuals to blocking pairs grows linearly with the number of propose–accept rounds executed by the Gale–Shapley algorithm for the stable marriage problem. Consequently, the participants can arrive at an almost stable matching even without full information about the problem instance; for each participant, knowing only its local neighbourhood is enough. In distributed-systems parlance, this means that if each person has only a constant number of acceptable partners, an almost stable matching emerges after a constant number of synchronous communication rounds.

We apply our results to give a distributed $(2 + \epsilon)$-approximation algorithm for maximum-weight matching in bicoloured graphs and a centralised randomised constant-time approximation scheme for estimating the size of a stable matching.

1 Introduction

The social networking boom brings up computational challenges unforeseen in the past. In a modern large-scale network, gathering full information about the whole network in one place is practically impossible. This motivates design and analysis of *local algorithms* [22, 26, 31] in which the output of a node depends only on the input within a constant number of edges (hops) from the node.

In this paper we study a local version of the classical Gale–Shapley algorithm [9] for the *stable marriage problem*. We show that early termination of the algorithm leads to a matching with relatively few unstable edges in the case when the preference list of each participant has bounded length.

*Helsinki Institute for Information Technology HIIT, University of Helsinki. *Address:* P.O. Box 68, FI-00014 University of Helsinki, Finland. *Email addresses:* patrik.floreen@cs.helsinki.fi, petteri.kaski@cs.helsinki.fi, valentin.polishchuk@cs.helsinki.fi, jukka.suomela@cs.helsinki.fi.

1.1 Matchings

We follow the convention (see, e.g., Fleiner [8]) that an instance of the stable marriage problem is specified by a simple undirected bipartite graph $\mathcal{G} = (R \cup B, E)$; the graph may be disconnected but there are no isolated nodes. Call the nodes in R *red*, the nodes in B *blue*, and the graph $\mathcal{G}$ a *bicoloured* graph. Each node $v \in R \cup B$ has a linear order on the adjacent nodes. The linear order constitutes the *matching preference* for v; if a node u is not adjacent to v, then neither u is an acceptable partner for v nor v is acceptable for u.

Denote by Δ the maximum degree of a node of $\mathcal{G}$. That is, Δ is the maximum number of acceptable partners for one participant. In a social network, the bound on the degree of a node v may not necessarily be due to v being particularly picky; the bound may just reflect the fact that the number of participants, information about whom is available (or comprehensible, or relevant) to v, is limited. Throughout this work we assume that Δ is a constant.

A set of edges $M \subseteq E$ is a *matching* if every node is incident to at most one edge in M. Two nodes joined by an edge in M are *matched* in M; a node not incident to an edge in M is *unmatched* in M. An edge $\{u, v\} \in E \setminus M$ is *unstable* relative to M if both (i) u is unmatched or prefers v over its current match in M; and (ii) v is unmatched or prefers u over its current match in M. The matching M is *stable* if there are no unstable edges.

Let $\epsilon > 0$ be some constant. This work is centred on the following notion of “almost stability”; see Section 6 for background and motivation.

Definition 1. A matching M is *ϵ-stable* if the number of unstable edges is at most $\epsilon|M|$.

A matching (stable or not) is *maximal* if it is not a subset of a larger matching; it is *maximum* if it has the maximum size among all the matchings. In an edge-weighted graph, a *maximum-weight* matching is the one that has the maximum weight among all the matchings. A *greedy* matching is a maximal matching obtained by adding the edges one by one, in order of decreasing weight. It is well-known that the greedy matching is a 2-approximation to the maximum-weight one [3].

1.2 Model of Distributed Computing

We view $\mathcal{G}$ as the communication graph of a distributed system: if $\{u, v\} \in E$, then the node u and the node v can exchange messages.

Let T be a constant that determines the trade-off between the stability and the running time. Initially, each node $v \in R \cup B$ knows the following information: its colour (i.e., whether $v \in R$ or $v \in B$), its degree $d(v)$, and the constant T. The node v has $d(v)$ ports through which it can communicate with its neighbours; the ports are numbered according to v's preferences.

We assume synchronous communication. On each time step, every node can (i) receive messages from its neighbours, (ii) perform deterministic local computations, and (iii) send a message to each of its neighbours. The algorithm runs for T synchronous time steps, after which each node needs to produce the output: which of the neighbours, if any, is its partner in the matching. Naturally we require that the output is consistent: if u announces that v is its partner, then v must announce that u is its partner.

To be consistent with the model traditionally used in the study of local algorithms [26], we allow arbitrarily large messages and unbounded local computation. However, our algorithms do not need to exploit either of these assumptions: the messages are very small (few bits per edge on each synchronous time step) and local computations are simple.

1.3 Statement of Results

Our main contribution lies in an analysis of the "transient phase" of the Gale–Shapley algorithm for the stable marriage problem. We truncate the algorithm to a specified number of proposal rounds and investigate the resulting matching for stability. We find that a constant number of rounds suffice to obtain a constant ratio of unstable edges to matching edges. We obtain the following result.

Theorem 1. *There exists a deterministic distributed algorithm that finds an ϵ-stable matching in a bicoloured graph in time $T \leq 4 + 2\Delta^2/\epsilon$.*

We also consider the case where $\mathcal{G}$ is an edge-weighted graph, and the preferences are determined by the edge weights: each node prefers its incident edges in order of their weight, with the heaviest edge being the most preferred. If all weights are distinct, the unique *stable* matching in $\mathcal{G}$ is the greedy one, and its weight is at least 1/2 of that of the maximum-weight matching. In general, an *almost* stable matching can be a poor approximation to the maximum-weight one; say, a super-heavy edge may be missing from the almost stable matching. Fortunately, the particular almost stable matching from Theorem 1 is much better in this sense: our algorithm returns an almost stable matching that is almost as large and heavy as a stable matching.

Theorem 2. *There exists a deterministic distributed algorithm that finds a $(2 + \epsilon)$-approximation for maximum-weight matching in bicoloured graphs in time $T \leq 4 + 2\Delta/\epsilon$.*

It is well-known that every stable matching in $\mathcal{G}$ has the same size [10, Section 1.4.2]; denote the size by $|M|$. Given access to $\mathcal{G}$ via a *preference oracle* (queried with a node, the oracle replies with the colour of the node and the adjacent nodes listed in order of preference), the algorithm in Theorem 1 enables estimation of the size of a stable matching using a constant number of queries to the oracle. We analyse the non-degenerate case $\Delta \geq 3$.

Theorem 3. *For any $0 < \delta \leq 1/2$, $0 < \epsilon \leq 1$, and $\Delta \geq 3$, there exists a randomised algorithm that, given access to a preference oracle for $\mathcal{G}$, makes at most $25000\epsilon^{-2}(\Delta - 1)^{3+4\Delta/\epsilon} \ln \delta^{-1}$ queries to the oracle and outputs with probability at least $1 - \delta$ an estimate $\hat{m}$ such that $\big|\hat{m} - |M|\big| \leq \epsilon|M|$.*

Here it should be stressed that the algorithm in Theorem 3 estimates the size of a stable matching, not just an ϵ-stable matching.

1.4 Overview

The rest of the paper is structured as follows. Section 2 reviews related work. Sections 3 and 4 present and analyse the distributed, truncated version of the Gale–Shapley algorithm. Section 5 applies this algorithm to prove Theorems 1–3. In Section 6 we present some concluding remarks that in particular relate stable and almost stable matchings in a distributed setting.

2 Related Work

The stable marriage problem remains a subject of active research. The book by Gusfield and Irving [10] is a comprehensive survey on the problem; see the MATCH-UP workshop [11] for the latest developments.

2.1 Almost Stable Matching

Abraham et al. [1] study almost stable matchings in the stable roommates problem. The recent work by Biró et al. [4] is particularly close to ours: they, too, consider the stable marriage problem with incomplete preference lists, and aim at finding a matching with few unstable edges. However, in terms of computational complexity, their work goes in the opposite direction. Their task is to find a *maximum* matching that minimises the number of unstable edges. It turns out that this makes the problem computationally much more *difficult*: the problem is NP-hard, unlike the classical stable marriage problem. In contrast, we do not require that the matching is a maximum matching, which makes the problem computationally *easier*: the problem admits a constant-time distributed algorithm, unlike the classical stable marriage problem.

2.2 Distributed Stable Matching

Everyone witnesses evolution of matchings in real-world and virtual-world social networks; in its simplest form, the matchings attempt to attain stability by switching along unstable edges. In a seminal work, Knuth [19] showed that such a switching may go in cycles, never resulting in a stable matching. Roth and Vande Vate [30] showed that switching *randomly* almost surely leads to stability.

Note that switching partners along unstable edges can be done in a distributed manner. The Gale–Shapley algorithm is also parallel by its nature: the proposals/rejects can be undertaken by all men/women simultaneously during synchronised *rounds* (albeit it can happen that only one man is free at a round [10, Section A.3], [33]). Lower bounds on the running time of the algorithm [10, Section 1.5] show that a linear number of rounds is required to attain stability. But can a *nearly* stable matching be obtained with fewer rounds?

Several works have addressed the last question with experiments. Quinn [29] observes experimentally that a matching with only a fraction of unstable edges emerges long before the Gale–Shapley algorithm converges. Lu and Zheng [23] propose a parallel algorithm that outperforms the Gale–Shapley algorithm in practice. Theorem 1 gives theoretical support to the findings in Quinn [29]. Theorem 1 also addresses the concern expressed in the conclusions of Lu and Zheng [23] where it is claimed that "Most of existing parallel stable matching algorithms cannot guarantee a matching with a small number of unstable pairs within a given time interval." Theorem 1 suggests that if the number of acceptable partners for each participant is bounded, the Gale–Shapley algorithm guarantees a small *relative* number of unstable edges.

From a theory perspective, apparently only a few papers address decentralised implementations of the Gale–Shapley algorithm and/or stability after early termination of a stable matching algorithm. In a recent paper [6] it is claimed that "little theory exists concerning instability." Khuller et al. [17] give bounds on the performance of a simple online algorithm. Feder et al. [7] show that a stable matching can be found on a polynomial number of processors in sublinear time; their algorithm is not local. Other work on parallel stable matching include Tseng and Lee [33], Tseng [32], and Hull [15]. Eriksson and Häggström [6] prove that a simple heuristic works well for *random* inputs. Our Theorem 1 shows that the Gale–Shapley algorithm works well for an arbitrary input.

Kipnis and Patt-Shamir [18] present lower bounds on the distributed complexity of computing a stable matching, and they also consider the case of almost stable matchings (they use the terminology "ρ-approximation for distributed stable matchings" to refer to an almost stable matching). However, their work uses a model of distributed computing in which the size of a message is bounded.

2.3 Local Algorithms and Matchings

The running time T of the algorithms presented in this work only depends on the degree bound Δ and the desired stability or approximation guarantee ϵ. For a constant Δ and ϵ, these are constant-time algorithms; the running time is independent of the number of nodes in the network, the diameter of the network, or other global properties. Such constant-time distributed algorithms are known as local algorithms [22, 26, 31].

There is a range of negative results related to local algorithms for maximal matching [22] and approximate maximum matching [5, 20, 21, 25]. Even if each node is assigned a unique identifier and the network topology is an n-cycle, it is not possible to break the symmetry in the network and find a constant-factor approximation for maximum matching. Without any auxiliary information beyond unique node identifiers, positive results are known only in rare special cases, most notably for graphs where each node has an odd degree [24, 26].

Hence some auxiliary information is needed. For example, in problems related to stable marriage, it is natural to assume that every participant knows his/her gender. This assumption has not been exploited much in prior work on distributed, deterministic constant-time algorithms. We are only aware of Hańćkowiak et al. [12], which shows that there is a constant-time algorithm for maximal matching in bicoloured bounded-degree graphs. Similarly to our algorithm, their algorithm does not need unique node identifiers; *port numbering* [2] is sufficient.

Other work on constant-time distributed algorithms for matching usually assumes either randomness [14,20,21,27,34] or geometric information [13,35]. We refer to the survey [31] for further information on local algorithms.

2.4 Centralised Constant-Time Algorithms and Matchings

Our centralised constant-time approximation algorithm in Theorem 3 is based on the ideas of Parnas and Ron [28] and Nguyen and Onak [27]. Their work presents constant-time approximation algorithms for estimating the size of a maximal matching, maximum-cardinality matching, and maximum-weight matching. Our work complements this line of research by presenting an algorithm for estimating the size of a stable matching.

3 Algorithm

We work with the following distributed variant of the Gale–Shapley algorithm. Each blue node $b \in B$ maintains the variable $p(b)$ which is the match of b in the current matching or $\perp$ (b is unmatched). Each red node $r \in R$ maintains the following variables: $C(r)$ is the list of candidates for matching; $c(r)$ is the candidate from whom r is waiting for a response or $\perp$ (r has no current candidate); and $p(r)$ is the match of r in the current matching or $\perp$ (r is unmatched).

The algorithm executes in *rounds*. Each round consists of two *turns*, a *blue turn* followed by a *red turn*. Each node $v \in R \cup B$ is active during turns of its colour.

Blue turn. Each blue node $b \in B$ completes the following read–compute–write cycle. Initially, $p(b) = \perp$.

1. Receive all incoming messages; let P be the subset of neighbours that sent the message 'propose'. If P is empty, do nothing during this turn.
2. If $p(b) \neq \perp$, then set $Q \leftarrow P \cup \{p(b)\}$; otherwise set $Q \leftarrow P$.
3. Let q be the node in Q most preferred by b.
4. If $q \neq p(b)$:
    (a) If $p(b) \neq \perp$ then send the message 'break' to $p(b)$.
    (b) Send the message 'accept' to q.
    (c) Set $p(b) \leftarrow q$.
5. For each $r \in P \setminus \{q\}$:
    (a) Send the message 'reject' to r.

Red turn. Each red node $r \in R$ completes the following read–compute–write cycle. Initially, $C(r)$ consists of all adjacent blue nodes, in order of decreasing preference, most preferred first; $c(r) = \perp$; $p(r) = \perp$.

1. If $c(r) \neq \perp$:
    (a) Receive a message m from $c(r)$.
    (b) If $m =$ 'accept' then $p(r) \leftarrow c(r)$.
    (c) If $m =$ 'reject' then remove $c(r)$ from $C(r)$.
    (d) $c(r) \leftarrow \perp$.
2. If $p(r) \neq \perp$:
    (a) Receive a message m from $p(r)$, if any.
    (b) If $m =$ 'break' then remove $p(r)$ from $C(r)$, and set $p(r) \leftarrow \perp$.
3. If $p(r) = \perp$ and $C(r)$ is not empty:
    (a) Let $c(r)$ be the first element of $C(r)$.
    (b) Send the message 'propose' to $c(r)$.

4 Analysis

We adopt the convention of using a subscript $i = 1, 2, \dots$ to denote the state of the algorithm at the end of round i. For example, $p_i(r)$ denotes the value of the variable $p(r)$ in the local state of r at the end of round i.

At the end of round $i = 1, 2, \dots$, the local state variables $p_i(\cdot)$ define a matching

$$\begin{aligned} M_i &= \{\{r, p_i(r)\} : r \in R,\ p_i(r) \neq \bot\} \\ &= \{\{b, p_i(b)\} : b \in B,\ p_i(b) \neq \bot\} \subseteq E. \end{aligned}$$

4.1 Lost Edges and Convergence to Stability

We start from a restatement, in our terms, of the fact that as the Gale–Shapley algorithm progresses, women only improve their match and men only get worse.

Lemma 1. *If an edge $\{r, b\} \in E \setminus M_i$ with $r \in R$ and $b \in B$ is unstable in M_i then b is present in $C_i(r)$ and r is unmatched at the end of round i.*

Proof. If b is not in $C_i(r)$, then either b rejected r or broke up with r earlier, which means that b prefers its current match to r. If b is in $C_i(r)$ and r is matched, then r has not proposed to b yet, which means that r prefers its current match to b. $\square$

During each round, each red node $r \in R$ removes at most one blue node $b \in B$ from $C(r)$. We say that such an edge $\{r, b\}$ is a *lost edge* because it can no longer occur in the matching. We write $L_i \subseteq E$ for the set of all edges lost by the end of round i. Note that $L_1 = \emptyset$ because only 'propose' messages are sent during the first round.

Because E is a finite set and $L_{i-1} \subseteq L_i \subseteq E$, there exists a $z \in \{2, 3, \dots\}$ such that $L_z = L_{z-1}$. Since received 'reject' and 'break' messages increase the number of lost edges, it follows that no such messages were received during round z. In particular, any unmatched $r \in R$ at the end of round z must have $C_{z-1}(r)$ empty (and hence $C_z(r)$ empty), because otherwise it would have received a 'reject' or 'break' during round z. It follows by Lemma 1 that M_z is a stable matching. Let us denote this stable matching by M_∞ in what follows.

4.2 Weight and Potential

Associate with each edge $e \in E$ a positive integer weight $w(e)$ such that the weights respect the preferences of each node; that is, whenever a node v prefers x over y, it holds that $w(\{v, x\}) \geq w(\{v, y\})$. This is always possible, e.g., by choosing $w \equiv 1$. In fact, to prove Theorem 1, taking $w \equiv 1$ is sufficient, while the proof (and actually the statement) of Theorem 2 deals with an arbitrary w.

Associate with each blue node $b \in B$ the *weight* $w_i(b) = w(\{b, p_i(b)\})$ if b is matched; otherwise $w_i(b) = 0$. The *total blue weight*, $w_i(B) = \sum_{b \in B} w_i(b)$, is equal to the total weight of the matching M_i.

Associate with each red node $r \in R$ a *potential* $f_i(r)$ as follows. If r is matched or $C_i(r)$ is empty, set $f_i(r) = 0$. Otherwise, set $f_i(r) = w(\{r, b\})$ where b is the first element of $C_i(r)$.

Intuitively, the potential $f_i(r)$ is an upper bound for the extra weight that r could have if we ran the algorithm further than i rounds. In a stable matching, $f_i(r) = 0$ for all $r \in R$, but we do not necessarily achieve this by running the algorithm for a constant number of rounds. However, we can derive an upper bound for the *total potential* $f_i(R) = \sum_{r \in R} f_i(r)$ by observing that lost edges are heavier than those along which proposals (will) go.

Lemma 2. *For all $i = 2, 3, \ldots$ it holds that $f_i(R) \leq w(L_i) - w(L_{i-1})$.*

Proof. Each $r \in R$ that has a positive potential at the end of round $i \geq 2$ has either received a 'reject' or 'break' during the round i; in both cases r has removed a unique blue node b from $C(r)$. The weight of the lost edge $\{r, b\}$ is at least the potential $f_i(r)$ because $C(r)$ is ordered by decreasing preference. The claim follows by taking the sum over all red nodes and edges lost during round i. $\square$

Lemma 3. *For all $i = 2, 3, \ldots$ it holds that $f_i(R) \leq f_{i-1}(R)$.*

Proof. Consider the red turn at round i. The potential of a red node $r \in R$ changes only if it receives a message. First, if r receives a 'reject' message during round i, the potential of r at end of round i may change but it does not increase, that is, $f_i(r) \leq f_{i-1}(r)$. Second, if r receives a 'break' message from a blue node b, there is a unique red node q that receives an 'accept' message from b. In this case the potential of q decreases from $f_{i-1}(q) = w(\{q, b\})$ to $f_i(q) = 0$ while the potential of r increases from $f_{i-1}(r) = 0$ to $f_i(r) \leq w(\{r, b\}) \leq w(\{q, b\}) = f_{i-1}(q)$. Third, if r receives an isolated 'accept' message (without an associated 'break' message), $f_i(r) = 0 \leq f_{i-1}(r)$. The claim follows by taking the sum over all 'reject' messages, all 'break'–'accept' matches, and all isolated 'accept' messages during round i. $\square$

Lemma 4. *For all $i = 2, 3, \ldots$ it holds that $w(L_i) \geq (i-1) f_i(R)$.*

Proof. By Lemma 2 and Lemma 3, we have

$$\begin{aligned} w(L_i) = w(L_i) - w(L_1) &= \sum_{j=2}^{i} \big(w(L_j) - w(L_{j-1})\big) \\ &\geq \sum_{j=2}^{i} f_j(R) \geq (i-1) f_i(R). \end{aligned}$$ □

Lemma 5. *For all $i = 2, 3, \dots$ it holds that $w(L_i) \leq (\Delta - 1) w_i(B)$*

Proof. A blue node $b \in B$ can lose an incident edge $\{r, b\}$ during round $j = 2, 3, \dots, i$ only if b is matched at the end of round j with a node that b prefers to r. Put otherwise, if $\{r, b\} \in L_j \setminus L_{j-1}$, then $w(\{r, b\}) < w_j(b) \leq w_i(b)$. The last inequality follows because each $b \in B$, once matched, only changes to a more preferred match. Furthermore, a blue node can lose at most $\Delta - 1$ incident edges. □

Lemma 6. *For all $\gamma > 0$ and $i \geq 1 + (\Delta - 1)/\gamma$ it holds that $f_i(R) \leq \gamma w_i(B)$.*

Proof. By Lemma 4 and Lemma 5. □

5 Proofs of Theorems 1, 2 and 3

Proof of Theorem 1. For the purpose of analysis, set $w(e) = 1$ for all $e \in E$. Then $|M_i| = w_i(B)$, so Lemma 6 and $i \geq 1 + (\Delta - 1)/\gamma$ imply $f_i(R) \leq \gamma |M_i|$. Denote by u_i the number of unstable edges in $E \setminus M_i$. Because $f_i(R)$ counts the number of unmatched red nodes with a nonempty $C_i(r)$, it follows from Lemma 1 that $u_i \leq \Delta f_i(R)$. Thus, $\gamma = \epsilon/\Delta$ gives $u_i \leq \epsilon |M_i|$ for $i \geq 1 + \Delta(\Delta - 1)/\epsilon$.

We can choose an integer $i < 2 + \Delta^2/\epsilon$. We can compute M_i in i rounds or $T = 2i < 4 + 2\Delta^2/\epsilon$ synchronous communication steps. □

Proof of Theorem 2. Let a positive integer weight $w(e)$ for each edge $e \in E$ be given as input. Re-assign the preferences of each node v so that whenever $w(\{v, x\}) > w(\{v, y\})$ for two edges $\{v, x\}, \{v, y\} \in E$, the node v prefers x to y. Execute the algorithm in Section 3 for i rounds to obtain the matching M_i. Let $M^* \subseteq E$ be a maximum-weight matching.

For each red node $r \in R$, let $g(r) \subseteq B$ consist of the matches of r in M_i and in M^*, if any. In particular, $|g(r)| \leq 2$ for all $r \in R$, and $|g^{-1}(b)| \leq 2$ for all $b \in B$.

Let $\{r, b\} \in M^*$ with $r \in R$ and $b \in B$. Exactly one of the following holds at the end of round i:

1. The node r has not received a response from b. If r is matched with $b_2 \neq b$, r prefers b_2 over b and $w(\{r, b\}) \leq w(\{r, b_2\}) = w_i(b_2)$. Otherwise r is unmatched, b is in $C_i(r)$, and $w(\{r, b\}) \leq f_i(r)$.

2. The node r has received a response from b. Then b is matched with r or with some other red node that b prefers over r. Thus $w(\{r,b\}) \le w_i(b)$.

For every $\{r,b\} \in M^*$ thus

$$w(\{r,b\}) \le f_i(r) + \sum_{c \in g(r)} w_i(c)$$

and hence

$$\begin{aligned} w(M^*) &= \sum_{\{r,b\} \in M^*} w(\{r,b\}) \\ &\le \sum_{r \in R} f_i(r) + \sum_{r \in R} \sum_{c \in g(r)} w_i(c) \\ &\le \sum_{r \in R} f_i(r) + \sum_{b \in B} 2w_i(b) \\ &= f_i(R) + 2w_i(B). \end{aligned}$$

From Lemma 6 we conclude that

$$w(M^*) \le (2+\epsilon) w_i(B) = (2+\epsilon) w(M_i)$$

whenever $i \ge 1 + (\Delta - 1)/\epsilon$.

We can choose an integer $i < 2 + \Delta/\epsilon$. We can compute M_i in i rounds or $T = 2i < 4 + 2\Delta/\epsilon$ synchronous communication steps. □

Proof of Theorem 3. Let us first recall a convenient Chernoff-type upper bound for the tail probability of a binomial random variable $Z \sim \mathrm{Bin}(k,p)$, where k is a positive integer and $0 \le p \le 1$. For $0 \le \beta \le 1$, we have

$$\Pr\bigl(|Z - pk| \ge \beta pk\bigr) \le 2 \exp\left(-\frac{\beta^2 pk}{3}\right). \tag{1}$$

(See Janson et al. [16, Corollary 2.3] for a proof.)

Let $0 < \delta \le 1/2$, $0 < \epsilon \le 1$, and $\Delta \ge 3$ be given. For the purpose of analysis, set $w(e) = 1$ for all $e \in E$. We estimate the size of the stable matching M_∞ in $\mathcal{G}$ using a randomised algorithm that queries the preference oracle for $\mathcal{G}$. We assume that the number of nodes, n, is given as input to the algorithm.

Let us first relate $|M_j|$ to $|M_\infty|$. We have $|M_j| \le |M_\infty|$ because every blue node, once matched, remains matched. Furthermore, $|M_\infty| \le |M_j| + f_j(R)$ because every red node $r \in R$ with empty $C_j(r)$ will be unmatched in M_∞. Let

$$\gamma = \frac{\epsilon}{8\Delta}, \quad j \ge 1 + \frac{2\Delta - 2}{\epsilon} = 1 + \frac{\Delta - 1}{4\Delta\gamma}. \tag{2}$$

Note that $\gamma < 1$. By $w_j(B) = |M_j|$ and Lemma 6 we have

$$|M_j| \le |M_\infty| \le (1 + 4\Delta\gamma)|M_j| \le |M_j| + \frac{\epsilon}{2} \cdot |M_\infty|. \tag{3}$$

It thus suffices to have an estimate for $|M_j|$.

Because $\mathcal{G}$ has no isolated nodes, $|R| \le \Delta|B|$ and $|B| \le \Delta|R|$. Because a stable matching is maximal and there are no isolated nodes,

$$\Delta|M_\infty| \ge |R|, \tag{4}$$
$$\Delta|M_\infty| \ge |B|. \tag{5}$$

By (3) we have $|M_\infty| \le 2|M_j|$ and thus

$$\frac{|R|}{2\Delta} \le |M_j| \le |R|. \tag{6}$$

Since $n = |R| + |B|$, we have

$$\frac{n}{\Delta + 1} \le |R| \le n. \tag{7}$$

Let

$$N_0 = \frac{6(\Delta+1)}{\gamma^2} \ln \frac{6}{\delta}, \quad N \ge N_0. \tag{8}$$

We use the following procedure to estimate $|M_j|$. First, select N nodes uniformly at random and query the oracle for their colour. Denote by X the number of red nodes among the N nodes. Then, select N red nodes uniformly at random; that is, select $2(\Delta+1)N$ nodes uniformly at random and select the first N red nodes among those (if not enough red nodes occur, output 'failure' and stop); denote by Z the number of red nodes obtained (that is, failure occurs if and only if $Z < N$). For each red node, we determine whether it is matched in M_j; denote by Y the number of selected red nodes that are matched in M_j. Output

$$\hat{m} = n \cdot \frac{X}{N} \cdot \frac{Y}{N}$$

and stop.

To analyse the procedure, let us first derive upper bounds for the probabilities of certain "bad" events.

Let us first derive an upper bound for the probability of failure. Let

$$p = \frac{|R|}{n}, \quad k = 2(\Delta+1)N, \quad \text{and} \quad \beta = \frac{1}{2p(\Delta+1)}.$$

By (7) we have $\beta \leq 1/2$. Furthermore, by (7) we have that $Z < N$ implies $|Z - pk| \geq N = \beta pk$. Thus, from (1), (7), and (8) it follows that

$$\begin{aligned}\Pr(Z < N) &\leq 2\exp\Big(-\frac{\beta^2 pk}{3}\Big)\\ &= 2\exp\Big(-\frac{Nn}{6(\Delta+1)|R|}\Big)\\ &\leq 2\exp\Big(-\frac{N}{6(\Delta+1)}\Big) \leq \frac{\delta}{3}.\end{aligned}$$

Let us now derive an upper bound for the probability that the random variable X/N significantly deviates from its expectation $|R|/n$. Let

$$p = \frac{|R|}{n}, \quad k = N, \quad \text{and} \quad \beta = \gamma.$$

Observe that $\beta \leq 1$ by (2). By (1), (7), and (8), we have

$$\begin{aligned}\Pr\Big(\Big|\frac{X}{N} - p\Big| \geq \beta p\Big) &= \Pr\big(|X - pk| \geq \beta pk\big)\\ &\leq 2\exp\Big(-\frac{\beta^2 pk}{3}\Big)\\ &= 2\exp\Big(-\frac{\beta^2 |R| N}{3n}\Big)\\ &\leq 2\exp\Big(-\frac{\beta^2 N}{3(\Delta+1)}\Big) \leq \frac{\delta}{3}.\end{aligned}$$

Finally, let us now derive a similar bound for the probability that the random variable Y/N significantly deviates from its expectation $|M_j|/|R|$. Let

$$p = \frac{|M_j|}{|R|}, \quad k = N, \quad \text{and} \quad \beta = \gamma.$$

Observe that $\beta \leq 1$ by (2). By (1), (6), and (8), we have

$$\begin{aligned}\Pr\Big(\Big|\frac{Y}{N} - p\Big| \geq \beta p\Big) &= \Pr\big(|Y - pk| \geq \beta pk\big)\\ &\leq 2\exp\Big(-\frac{\beta^2 pk}{3}\Big)\\ &= 2\exp\Big(-\frac{\beta^2 |M_j| N}{3|R|}\Big)\\ &\leq 2\exp\Big(-\frac{\beta^2 N}{6(\Delta+1)}\Big) \leq \frac{\delta}{3}.\end{aligned}$$

Next we show that, with probability at least $1 - \delta$, the estimate $\hat{m}$ approximates $|M_\infty|$ with the claimed accuracy. Observe first that $|ac - bd| \leq$

$a|c-d|+d|a-b|$ for $a,b,c,d \geq 0$. Thus, with probability at least $1-\delta$, both no failure occurs and

$$\begin{aligned}\big|\hat{m} - |M_j|\big| &= n\left|\frac{X}{N}\cdot\frac{Y}{N} - \frac{|R|}{n}\cdot\frac{|M_j|}{|R|}\right| \\ &\leq n\left(\frac{X}{N}\cdot\left|\frac{Y}{N} - \frac{|M_j|}{|R|}\right| + \frac{|M_j|}{|R|}\cdot\left|\frac{X}{N} - \frac{|R|}{n}\right|\right) \\ &\leq n\left(\frac{X}{N}\cdot\gamma\cdot\frac{|M_j|}{|R|} + \frac{|M_j|}{|R|}\cdot\gamma\cdot\frac{|R|}{n}\right) \\ &\leq 2\gamma n = 2\gamma\big(|R|+|B|\big) \leq 4\Delta\gamma|M_\infty| = \frac{\epsilon}{2}\cdot|M_\infty|\end{aligned} \tag{9}$$

where the last inequality follows by (4) and (5). Thus, with probability at least $1-\delta$, we have

$$\begin{aligned}\big|\hat{m} - |M_\infty|\big| &\leq \big|\hat{m} - |M_j|\big| + \big||M_j| - |M_\infty|\big| \\ &\leq \frac{\epsilon}{2}\cdot|M_\infty| + \frac{\epsilon}{2}\cdot|M_\infty| = \epsilon|M_\infty|\end{aligned}$$

where the last inequality follows by (3) and (9).

It remains to derive an upper bound for the number of oracle queries we make to compute the estimate. First, we make N queries to determine X. Second, we make $2(\Delta+1)N$ queries to obtain N red nodes (or declare failure). Third, for each of the N red nodes, we execute the algorithm in Section 3 for j rounds to determine whether the node is matched in M_j. Each round of the algorithm propagates information (messages) for at most two hops in $\mathcal{G}$, which implies that we can decide whether any given $r \in R$ is matched in M_j if we know the preferences of all nodes within distance $2j$ from r in $\mathcal{G}$. There are at most

$$1 + \Delta\sum_{i=0}^{2j-1}(\Delta-1)^i = 1 + \frac{\Delta}{\Delta-2}\big((\Delta-1)^{2j}-1\big) < 3(\Delta-1)^{2j}$$

such nodes. Because $\Delta \geq 3$ and $\delta \leq 1/2$, we have

$$\begin{aligned}N_0 &= 384\Delta^2(\Delta+1)\epsilon^{-2}\big(\ln 6 + \ln\delta^{-1}\big) \\ &\leq 384\cdot\left(\frac{3}{2}\right)^2\cdot\frac{4}{2}\cdot(\Delta-1)^3\epsilon^{-2}\left(\frac{\ln 6}{\ln 2}+1\right)\ln\delta^{-1} \\ &< 6195(\Delta-1)^3\epsilon^{-2}\ln\delta^{-1}.\end{aligned}$$

Therefore we can select integers j and N such that $j \leq 2\Delta\epsilon^{-1}$ and $N \leq 6250(\Delta-1)^3\epsilon^{-2}\ln\delta^{-1}$. In total we make at most

$$\begin{aligned}N + 2(\Delta+1)N + 3(\Delta-1)^{2j}N &\leq 4(\Delta-1)^{2j}N \\ &< 25000\epsilon^{-2}(\Delta-1)^{3+4\Delta/\epsilon}\ln\delta^{-1}\end{aligned}$$

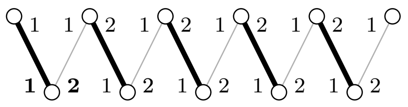
queries. This completes the proof. □

6 Concluding Remarks

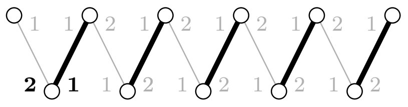
Robustness and Parallelism. We mention three further interpretations of Theorem 1, each of which is an immediate consequences of the fact that in T synchronous communication steps, information can only be propagated from distance T in the network.

- *Robustness and dynamic graphs.* The ϵ-stable matching M from Theorem 1 is robust in a dynamic network. A change in the network only affects M in the radius-T neighbourhood around the point of change.
- *Available information.* Information within the radius-T neighbourhood of an edge e is sufficient to determine whether e is part of the globally consistent ϵ-stable matching M.
- *Parallelism and circuit complexity.* For any graph, we can construct a bounded-fan-in Boolean circuit that maps the matching preferences to an ϵ-stable matching. The depth of the circuit only depends on the constants Δ and ϵ, not on the size of the graph.

Almost Stable vs. Stable. While an almost stable matching is robust to change in the preferences, this is not the case for a stable matching. For example, consider the following graph where the numbered edge ends indicate preference rankings (the most preferred match has rank 1).



Now transpose the preferences shown in boldface to obtain a graph whose unique stable matching is edge-disjoint from the unique stable matching in the original graph.



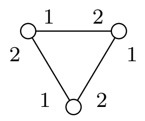
Thus, a single transposition in preferences can force every match to break up.

This example shows that every node must have essentially complete information about the network to arrive at a stable matching. In particular, if each node initially knows only its neighbours and its own preferences, the number of communication steps required between neighbours is linear in the diameter of the network.

Definition of Almost Stable. Our results are somewhat oblivious to the exact definition of an "almost stable" matching. The number of unstable edges (blocking pairs) appears to be generally accepted as a basic measure of instability [1, 4, 6, 17], but one could equally well consider other measures. For example, one can consider the number of nodes that are endpoints of unstable edges.

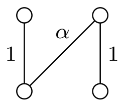
Naturally we must measure stability in relation to some other quantity; an absolute guarantee of stability cannot be achieved in constant time. We have chosen to measure the number of unstable edges relative to $|M|$, as this gives us the strongest results when compared with other quantities such as $|R|$, $|B|$, $|R \cup B|$, $|E|$, and $|R||B|$, each of which is at least as large as $|M|$. Eriksson and Häggström [6] provide arguments in favour of comparing unstable edges to $|E|$.

Marriages vs. Roommates. An immediate question is whether our results apply to the stable roommates problem – the non-bipartite version of the stable marriage problem. Unfortunately, a non-bipartite graph need not have an ϵ-stable matching for $\epsilon < 1$: consider a triangle with "rotational-symmetric" preferences – there is one unstable edge to every matching edge.



Bicoloured vs. Bipartite. A graph is 2-colourable if and only if it is bipartite. Given a global view of a bipartite graph, it is trivial to 2-colour the graph. In the context of distributed algorithms this relationship is more subtle, however. Linial [22] shows that there is no constant-time algorithm for finding a maximal matching in a cycle with $2n$ with vertices. Czygrinow et al. [5] show that finding a constant-factor approximation to maximum-cardinality matching with a deterministic constant-time algorithm is not possible either. However, these negative results heavily rely on the fact that the nodes in the bipartite graph do not know their colours in a 2-colouring. Theorem 2 shows that knowledge of the colour is not only necessary but also sufficient to find a constant-factor approximation of maximum-weight matching with a deterministic distributed constant-time algorithm.

A Lower Bound on Approximating Maximum-Weight Matching. The approximation factor of $2 + \epsilon$ in Theorem 2 is almost the best possible for *any* algorithm (distributed or centralised) that has access only to the relative order of the weights, and not their numerical values. To see that no algorithm can have the approximation factor $2 - \epsilon$ for any $\epsilon > 0$, consider the following graph with the weight $\alpha > 1$.



If $\alpha \gg 1$, an algorithm must include the edge with weight α into the output (otherwise the algorithm has no approximation guarantee at all). But then the algorithm, having access only to the relative order of the weights, must include this edge also when $1 < \alpha < 2/(2-\epsilon)$, which contradicts with the approximation guarantee $2-\epsilon$.

Acknowledgements

This work was supported in part by the Academy of Finland, Grants 116547, 117499, and 118653 (ALGODAN), by Helsinki Graduate School in Computer Science and Engineering (Hecse), and by the Foundation of Nokia Corporation.

We acknowledge email exchange with the members of the Algorithms and Complexity Research at Glasgow group, and thank the organisers of ALGO 2008 for the creative atmosphere at the conference. We thank the anonymous reviewers for their useful comments.

References

[1] David J. Abraham, Péter Biró, and David F. Manlove. "Almost stable" matchings in the roommates problem. In *Proc. 3rd Workshop on Approximation and Online Algorithms (WAOA, Palma de Mallorca, Spain, October 2005)*, volume 3879 of *Lecture Notes in Computer Science*, pages 1–14. Springer, Berlin, Germany, 2006.

[2] Dana Angluin. Local and global properties in networks of processors. In *Proc. 12th Annual ACM Symposium on Theory of Computing (STOC, Los Angeles, CA, USA, April 1980)*, pages 82–93. ACM Press, New York, NY, USA, 1980.

[3] David Avis. A survey of heuristics for the weighted matching problem. *Networks*, 13(4):475–493, 1983.

[4] Péter Biró, David F. Manlove, and Shubham Mittal. Size versus stability in the marriage problem. In *Proc. 6th Workshop on Approximation and Online Algorithms (WAOA, Karlsruhe, Germany, September 2008)*, volume 5426 of *Lecture Notes in Computer Science*, pages 15–28. Springer, Berlin, Germany, 2009.

[5] Andrzej Czygrinow, Michał Hańćkowiak, and Wojciech Wawrzyniak. Fast distributed approximations in planar graphs. In *Proc. 22nd International Symposium on Distributed Computing (DISC, Arcachon, France, September 2008)*, volume 5218 of *Lecture Notes in Computer Science*, pages 78–92. Springer, Berlin, Germany, 2008.

[6] Kimmo Eriksson and Olle Häggström. Instability of matchings in decentralized markets with various preference structures. *International Journal of Game Theory*, 36(3–4):409–420, 2008.

[7] Tomás Feder, Nimrod Megiddo, and Serge A. Plotkin. A sublinear parallel algorithm for stable matching. *Theoretical Computer Science*, 233(1–2):297–308, 2000.

[8] Tamás Fleiner. A fixed-point approach to stable matchings and some applications. *Mathematics of Operations Research*, 28(1):103–126, 2003.

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

[10] Dan Gusfield and Robert W. Irving. *The Stable Marriage Problem: Structure and Algorithms.* The MIT Press, Cambridge, MA, USA, 1989.

[11] Magnús M. Halldórsson, Robert W. Irving, Kazuo Iwama, and David F. Manlove, editors. *MATCH-UP: Matching Under Preferences – Algorithms and Complexity. Satellite workshop of ICALP 2008*, July 2008.

[12] Michał Hańćkowiak, Michał Karoński, and Alessandro Panconesi. On the distributed complexity of computing maximal matchings. In *Proc. 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA, San Francisco, CA, USA, January 1998)*, pages 219–225. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1998.

[13] Marja Hassinen, Joel Kaasinen, Evangelos Kranakis, Valentin Polishchuk, Jukka Suomela, and Andreas Wiese. Analysing local algorithms in location-aware quasi unit-disk graphs, 2009. Manuscript submitted for publication.

[14] Jaap-Henk Hoepman, Shay Kutten, and Zvi Lotker. Efficient distributed weighted matchings on trees. In *Proc. 13th International Colloquium on Structural Information and Communication Complexity (SIROCCO, Chester, UK, July 2006)*, volume 4056 of *Lecture Notes in Computer Science*, pages 115–129. Springer, Berlin, Germany, 2006.

[15] M. Elizabeth C. Hull. A parallel view of stable marriages. *Information Processing Letters*, 18(2):63–66, 1984.

[16] Svante Janson, Tomasz Łuczak, and Andrzej Rucinski. *Random Garphs.* John Wiley & Sons, New York, NY, USA, 2000.

[17] Samir Khuller, Stephen G. Mitchell, and Vijay V. Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theoretical Computer Science*, 127(2):255–267, 1994.

[18] Alex Kipnis and Boaz Patt-Shamir. A note on distributed stable matching. In *Proc. 29th IEEE International Conference on Distributed Computing Systems (ICDCS, Montreal, QC, Canada, June 2009)*, pages 466–473. IEEE, Piscataway, NJ, USA, 2009.

[19] Donald E. Knuth. *Mariages Stables.* Les Presses de l'Université de Montréal, 1976.

[20] Fabian Kuhn. *The Price of Locality: Exploring the Complexity of Distributed Coordination Primitives.* PhD thesis, ETH Zürich, 2005.

[21] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. The price of being near-sighted. In *Proc. 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA, Miami, FL, USA, January 2006)*, pages 980–989. ACM Press, New York, NY, USA, 2006.

[22] Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

[23] Enyue Lu and S.Q. Zheng. A parallel iterative improvement stable matching algorithm. In *Proc. 10th International Conference on High Performance Computing (HiPC, Hyderabad, India, December 2003)*, volume 2913 of *Lecture Notes in Computer Science*, pages 55–65. Springer, Berlin, Germany, 2003.

[24] Alain Mayer, Moni Naor, and Larry Stockmeyer. Local computations on static and dynamic graphs. In *Proc. 3rd Israel Symposium on the Theory of Computing and Systems (ISTCS, Tel Aviv, Israel, January 1995)*, pages 268–278. IEEE, Piscataway, NJ, USA, 1995.

[25] Thomas Moscibroda. *Locality, Scheduling, and Selfishness: Algorithmic Foundations of Highly Decentralized Networks.* PhD thesis, ETH Zürich, 2006.

[26] Moni Naor and Larry Stockmeyer. What can be computed locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995.

[27] Huy N. Nguyen and Krzysztof Onak. Constant-time approximation algorithms via local improvements. In *Proc. 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS, Philadelphia, PA, USA, October 2008)*, pages 327–336. IEEE Computer Society Press, Los Alamitos, CA, USA, 2008.

[28] Michal Parnas and Dana Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theoretical Computer Science*, 381(1–3):183–196, 2007.

[29] Michael J. Quinn. A note on two parallel algorithms to solve the stable marriage problem. *BIT Numerical Mathematics*, 25(3):473–476, 1985.

[30] Alvin E. Roth and John H. Vande Vate. Random paths to stability in two-sided matching. *Econometrica*, 58(6):1475–1480, 1990.

[31] Jukka Suomela. Survey of local algorithms. `http://www.iki.fi/jukka.suomela/local-survey`, 2009. Manuscript submitted for publication.

[32] S. S. Tseng. The average performance of a parallel stable marriage algorithm. *BIT Numerical Mathematics*, 29(3):448–456, 1989.

[33] S. S. Tseng and R. C. T. Lee. A parallel algorithm to solve the stable marriage problem. *BIT Numerical Mathematics*, 24(3):308–316, 1984.

[34] Mirjam Wattenhofer and Roger Wattenhofer. Distributed weighted matching. In *Proc. 18th International Symposium on Distributed Computing (DISC, Amsterdam, Netherlands, October 2004)*, volume 3274 of *Lecture Notes in Computer Science*, pages 335–348. Springer, Berlin, Germany, 2004.

[35] Andreas Wiese and Evangelos Kranakis. Local maximal matching and local 2-approximation for vertex cover in UDGs. In *Proc. 7th International Conference on Ad-Hoc Networks & Wireless (AdHoc-NOW, Sophia Antipolis, France, September 2008)*, volume 5198 of *Lecture Notes in Computer Science*, pages 1–14. Springer, Berlin, Germany, 2008.