Fair Procedures for Fair Stable Marriage Outcomes

Nikolaos Tziavelis,[1] Ioannis Giannakopoulos,[2] Rune Quist Johansen,[3]
Katerina Doka,[2] Nectarios Koziris,[2] Panagiotis Karras[4]
[1]Northeastern University, [2]National Technical University of Athens, [3]Aalborg University, [4]Aarhus University

Abstract

Given a two-sided market where each agent ranks those on the other side by preference, the stable marriage problem calls for finding a perfect matching such that no pair of agents prefer each other to their matches. Recent studies show that the number of stable solutions can be large in practice. Yet the classical solution to the problem, the Gale-Shapley (GS) algorithm, assigns an *optimal* match to each agent on one side, and a *pessimal* one to each on the other side; such a solution may fare well in terms of equity *only* in highly asymmetric markets. Finding a stable matching that *minimizes* the *sex equality cost*, an equity measure expressing the discrepancy of mean happiness among the two sides, is strongly NP-hard. Extant heuristics either *(a)* oblige some agents to involuntarily abandon their matches, or *(b)* bias the outcome in favor of some agents, or *(c)* need high-polynomial or unbounded time.

We provide the first procedurally fair algorithms that output equitable stable marriages and are guaranteed to terminate in at most cubic time; the key to this breakthrough is the monitoring of a monotonic state function and the use of a selective criterion for accepting proposals. Our experiments with diverse simulated markets show that: *(a)* extant heuristics fail to yield high equity; *(b)* the best solution found by the GS algorithm can be very far from optimal equity; and *(c)* our procedures stand out in both efficiency and equity, even when compared to a non-procedurally fair approximation scheme.

1 Introduction

The *stable marriage* (or matching) problem (SMP) calls for each agent on two sides to find a match on the other side. Eventually, there should be no pair of agents that would rather be matched to each other than to their allocated matches. Each agent holds a *preference list* for members of the opposite side. Gale and Shapley (1962) showed that a stable solution can be found in quadratic time. The problem finds application in several two-sided markets, including those among doctors and hospitals (Roth 1984; 2008), students and schools (Teo, Sethuraman, and Tan 2001), or sailors and vessels (Liebowitz and Simien 2005). Roth and Shapley shared the 2012 Nobel Memorial Prize in Economic Sciences for that work among others.

The set of stable matchings in a real-world market is large (Hassidim, Romm, and Shorrer 2017). Still, the Gale-Shapley algorithm yields one that is *most preferred* by the one side and *least preferred* by the other (McVitie and Wilson 1971). Arguably, many societal applications require a stable matching that compromises among the two sides (Gusfield and Irving 1989; Roth and Sotomayor 1990; Romero-Medina 2005). For example, in a health care market, each surgeon may have preferences for which anesthetist to work with, and vice versa; ensuring a sense of fairness among the two sides may lead to better performance and societal outcomes (Romero-Medina 2005). Thus, the problem of finding a stable matching making a good compromise among the two sides is worth studying (Roth 2018).

A popular way to measure the goodness of compromise, or equity, among the two sides is the *sex equality cost* (Gusfield and Irving 1989); yet the problem of optimizing that measure is NP-hard (Kato 1993); no known algorithm offers a theoretical approximation guarantee with respect to optimality. An approximation algorithm (Iwama, Miyazaki, and Yanagisawa 2010) provides a guarantee *only* with respect to Gale-Shapley algorithm's solutions. Besides, by this algorithm, some agents have to *involuntarily* forgo their match in an already stable matching for a less preferable one, so that the whole community moves to another stable matching. A local-search heuristic (Viet et al. 2016a) shares this *involuntary* character. Proposed voluntary procedures either bias their result in favor of some agents (Ma 1996; Aldershof, Carducci, and Lorenc 1999), or bear no theoretical guarantee to terminate in finite time (Everaere, Morge, and Picard 2013; Gelain et al. 2013; Giannakopoulos et al. 2015), or need high-polynomial time (Dworczak 2016). To date, no *voluntary and unbiased* procedure is guaranteed to reach a stable matching in less than *quartic* time.

We develop the first, to our knowledge, voluntary and unbiased procedures that are theoretically guaranteed to reach a stable marriage in *cubic* time. These procedures monitor a monotonic state function to enforce progression and apply a selective criterion for accepting match proposals when facing difficulty. Our thorough experimental study shows that our procedures outperform others in terms of equity and efficiency over diverse preference distributions.

2 Background and Related Work

In an instance I of the *stable marriage problem* (SMP), we are given n men and n women. Each person has a *preference list* ranking the members of the other side by worsening order of preference. Let ℓ_{q_i} be the preference list of agent q_i; $\ell_{q_i}[k] = p_j$ means that p_j is the k-th preference of q_i; we also write $pr_{q_i}(p_j) = k$. If a woman w prefers m_1 to m_2, i.e., $pr_w(m_1) < pr_w(m_2)$, we denote that as $m_1 \succ_w m_2$; likewise for men's preferences. A *matching* M on I is a set of disjoint pairs. When a woman w and a man m are matched in M, we write $M(w) = m$ and $M(m) = w$. A woman w and a man m form a *blocking pair* for M when: (i) $M(m) \neq w$; (ii) $w \succ_m M(m)$; and (iii) $m \succ_w M(w)$. A matching M is *stable* if no blocking pair exists for M. The SMP calls for finding a stable *perfect* matching.

2.1 The Gale-Shapley Algorithm

In an iteration of the Gale-Shapley (GS) algorithm (1962), each single man proposes to the woman of his next preference κ, starting with $\kappa = 0$. A woman w accepts a proposal from a man m if she is single or prefers m to her current fiancé, $M(w)$. Thus, w *accepts* a proposal from m under a condition expressed by the Boolean predicate:

$$\texttt{accept}(w, m) = \texttt{single}(w) \ \vee \ m \succ_w M(w) \quad (1)$$

where M is the matching created so far. In case the proposal is rejected, m moves to preference $\kappa+1$ in the next iteration.

The GS algorithm arrives at a stable matching in $O(n^2)$ steps (Gusfield and Irving 1989), thanks to the monotonic property that the number of pairs is non-decreasing: an eloping woman breaks her old pair, yet forms a new pair with a single man; there are no circumstances in which a matched woman elopes for a matched man, breaking two old pairs and forming only one new, since only unmatched men issue proposals. Notably, a woman's preference for her match can only improve, while that of a man can only worsen.

2.2 Striving for Equity

Unfortunately, the GS algorithm is biased: it returns, out of a set of stable solutions that is large in real-world markets (Hassidim, Romm, and Shorrer 2017) and exponentially growing in the worst case (Irving and Leather 1986), one that is most preferable to each *proposing* agent and least preferable to each *receiving* agent (McVitie and Wilson 1971). For example, in case men's first preferences do not conflict, each man may obtain his first choice, regardless of how satisfactory that solution is to women. This bias calls for solutions of higher *equity*. A popular equity measure, the *sex equality cost*, measures the gap between the two sides' average obtained preference:

$$d(M) = \left| \sum_{(m,w)\in M} pr_m(w) - \sum_{(m,w)\in M} pr_w(m) \right| \quad (2)$$

where $pr_m(w)$ (respectively, $pr_w(m)$) denotes the position of woman w in man m's preference list (respectively, of m in w's list). Other measures of the quality, though not the equity, of a matching are the *egalitarian cost*, $c(M) = \sum_{(m,w)\in M} pr_m(w) + \sum_{(m,w)\in M} pr_w(m)$, and the *regret cost* $r(M) = \max_{(m,w)\in M} \max\{pr_m(w), pr_w(m)\}$.

There are $O(n^2)$ (Gusfield 1987) and $O(n^4)$ (Romero-Medina 2005) algorithms for minimizing regret cost. Egalitarian cost is minimized in $O(n^3)$ (Irving, Leather, and Gusfield 1987; Feder 1992). Such solutions explore a lattice that contains all stable matchings (Irving 2008). Yet minimizing the equity-oriented measure of sex equality cost is strongly NP-hard (Kato 1993). An $O\left(n^{3+\frac{1}{\epsilon}}\right)$ approximation algorithm (Iwama, Miyazaki, and Yanagisawa 2010) provides bounds with respect to the outputs of the GS algorithm, but *not* versus the optimal cost; an exponential-time algorithm solves the problem exactly (McDermid and Irving 2014).

Besides, all herein discussed algorithms require that some agents unwillingly abandon a match, so as to move from one stable matching to another; local-search algorithms (Viet et al. 2016b; 2016a) share this *coercive* character. We are interested in procedures that exhibit *procedural fairness*, hence can be voluntarily applied by agents in a real-world market.

2.3 Procedural Fairness

Two works suggest procedures in which both sides act voluntarily. However, these procedures have an inherent bias in favor of some agents, like the bias of the GS algorithm in favor of one side: ROM (Ma 1996) eliminates blocking pairs among an iteratively growing subset of agents; thus, the *later* an agent is included in this growing subset, the more the outcome is biased in its favor (Blum, Roth, and Rothblum 1997). LOTTO (Aldershof, Carducci, and Lorenc 1999) iteratively removes unattainable pairings from preference lists and assigns a random agent to its *best* remaining preference. Then the *earlier* an agent is chosen, the more the outcome is irrevocably biased in its favor. Some recent works suggest procedures that are both *voluntary* (i.e., do not oblige any agent to act unwillingly) and *unbiased* (i.e., do not favor any one agent or one side by design). We call such procedures *fair*. Out of those, the following may enter cyclical loops and have no termination guarantee:

- SML2 (Gelain et al. 2013) iteratively eliminates selected blocking pairs to transform an initial random matching to a stable one. However, its core process may endlessly cycle (Tamura 1993).
- SWING (Everaere, Morge, and Picard 2013) iteratively lets each agent reissue proposals, until it reaches either its current match or a new position in its preference list. However, SWING is not guaranteed to terminate in finite iterations.
- ESMA (Giannakopoulos et al. 2015), iteratively lets each *unmatched* agent propose to its following preference κ; while faster than SWING, neither ESMA has a theoretical termination guarantee: any agent may cycle, first accepting a proposal bettering its κ index, then being abandoned and worsening κ again.

DACC (Dworczak 2016) is the only known fair procedure that provably terminates, albeit in $O(n^4)$, with extra measures required to ensure termination (Dworczak 2019), fol-

lowing an arbitrary sequence of proposals. Thus, no known fair procedure has less than quartic time.

3 Monotonic Properties

By the fair procedures we have discussed, an agent is in one of three states:

- *single* (S): single, hence issuing proposals or having reached the end of its preference list;
- *discontent* (D): in a pair, yet still issuing proposals to unexamined preferences more preferable than its match;
- *content* (C): matched and having already issued all proposals up to its match in its preference list, hence not issuing proposals.

We aim to develop *fair* procedures that reach an *equitable* stable marriage in *cubic* time. To do so, we first revisit the monotonic properties of GS and examine whether they are retained by the unbiased procedures of Section 2.3. As we saw in Section 2.1, in GS the preference rank of a recipient (woman) for its assignee is not worsening, while that of a proposer (man) is not improving. These properties are lost when both sides act as both proposers and recipients: the preference index κ_p to which an agent p issues its next proposal may fluctuate both up and down in its preference list. Then, the number of pairings in GS is non-decreasing; this property is lost too: when a proposing agent abandons one pairing to form another with an accepting recipient, two pairings may be broken and only one created.

Thus, the monotonicity that guarantees the termination of GS is lost. However, there may be some other monotonicity to discover. We look at this matter in more detail. As discussed, an agent can be in a single, content, or discontent state, S, C, D respectively. When a new couple (p, q) is formed, the proposing agent p has already examined all options it prefers to q, hence will necessarily become C; the recipient agent q will be either C or D, depending on whether its preference rank for the proposer, $pr_q[p]$, is above or below κ_q. We denote the changes occurring when an agent X proposes to another agent Y, who accepts, using the syntax:

⋆X Y⋆ → ⋆ XY ⋆

where two closely adjacent symbols (like ⋆X and XY) denote a couple; spaces separate one couple or agent from another; ⋆ is a placeholder for possible assignees to the proposing agent X and the recipient agent Y prior to the acceptance of X's proposal by Y and the creation of couple XY. The left side of the arrow indicates the state of affairs before the proposal of X is accepted by Y, while the right side indicates the state of affairs after that acceptance. All of the symbols X, Y, and ⋆ can assume the values S, C, and D, representing the states of the respective agents before and after the proposal's acceptance. It follows that CC denotes a couple of two content agents, while CD and DC denote pairings of one content and one discontent agent. By this notation, we distinguish two possible repercussions of a proposal's acceptance. First, such an acceptance may lead to the creation of an additional CC couple. That may happen in any of the following ways:

```
 S  CD  →  CC  S
 S  DC  →  CC  S
 S  S   →  CC
CD  CD  →  S   CC  S
CD  DC  →  S   CC  S
CD  S   →  S   CC
    DC  →  CC
```

We obtain this list by outlining all ways in which an S or D agent may propose to a C, D, or S agent, and lead to an additional CC couple; the represented agents appear in the *same order* on the left and right side of the syntax, while their state may change. For example, the fifth line above indicates the case where the D agent in a CD couple proposes to the D agent in another DC couple, who prefers the proposer to its κ^{th} preference; the latter accepts the proposal, creating one new CC couple and turning the two former assignees of the newly paired agents into singles. The seventh line shows that case where a D agent proposes to its own partner. A proposal's acceptance may *decrease* the number of couples, but cannot reduce the number of CC couples. It may fail to create an additional CC couple, yet their number remains stable, in any of the following ways:

```
 S  CC  →  CC  S
 S  DC  →  CD  S
 S  S   →  CD
CD  CC  →  S   CC  S
CD  DC  →  S   CD  S
CD  S   →  S   CD
```

Then the following theorem follows.

Theorem 3.1. The number of CC couples is non-decreasing in *any* procedure where both sides propose.

Proof. The number of CC couples could only decrease if an accepted proposal were to break such a couple without creating a new one. A proposer p is single or discontent, hence not party to a CC couple. On the other hand, a recipient q may be party to a CC couple and accept the proposal from p in case $p \succ_q M(q)$. Then q remains content, hence one CC couple is broken and another is created. Thus, the number of CC couples is non-decreasing. □

4 Revising the Acceptance Criterion

From our preceding analysis it follows that there are two kinds of content agents: those in CC couples and those in CD couples; the content state of the latter is precarious, as their assignee is discontent and thus more likely to elope. We reason that such content agents in CD couples may not be rendered content in the first place. After all, their confinement to a CD couple prevents them from proposing. We may ban the *discontent* state as such. That state arises out of the acceptance criterion in Equation (1): a single agent accepts a proposal in all cases, and may thus become discontent. We can substitute this criterion by a *selective* one:

$$\texttt{accept}(q, p) = p \succ_q \ell_q[\kappa_q] \tag{3}$$

By this criterion, q accepts a proposal from p *if and only if* p is preferable to q over its next proposal target. Such an acceptance renders q Content, and the proposer p is always Content after a proposal is accepted; hence, by this criterion,

discontent agents do not arise. We call the use of this criterion *discontent suspension*. We have determined that DACC can also employ discontent suspension instead of more convoluted processes to ensure termination (Dworczak 2019).

5 Terminating Procedures

Here, we use the building blocks of Sections 3 and 4 to build fair procedures that reach a stable matching in cubic time. By Theorem 3.1, there exists a non-decreasing state function: the number of CC couples. If we could enforce that function's increase at each iteration, the procedure would terminate in $O(n)$ iterations.

Thus, we try to enforce the increase of the number of CC couples (in short, CC *increase*). We observe that, when only one side issues proposals, there can be no cyclical behavior, i.e., no return to a previous state. The agents on the chosen side descend their preference lists monotonically. Eventually, each such agent becomes *idle*, i.e., content or single at the end of its preference list. Even so, we cannot be sure that a CC increase will occur before that happens. Yet, if we then allow the other side to take turn, we can prove that it will.

Theorem 5.1. Assume each agent on the one side, P is *idle* (i.e., content or single at the end of its preference list) and there is at least one non-idle single on the other side, Q. Then, proposals from Q incur a CC increase before all singles in Q become idle.

Proof. As all agents on side P are idle, the only possible repercussions of a proposal by a single agent on side Q are:

S S → CC
S CD → CC S
S CC → CC S

Out of these actions, the first two incur CC increase. The third does not incur CC increase and spawns a new *single* on side Q. Assume that all singles on side Q become idle, and a CC increase does not occur; then all proposals issued are of the third type. The overall process is monotonic, as single agents descend their preference lists. Eventually, there is at least one single agent q left on side Q, hence there should be at least one single agent p on side P too, who is also idle. Then, p and q must have proposed to, and rejected, each other. Without loss of generality, assume p was the last one to issue such a proposal, which q rejected. Since q is now an idle single, it must have later returned the proposal to p, which leads to a contradiction. □

Theorem 5.1 shows how we can enforce a CC increase. Besides, when *all* agents become idle there can be no idle singles, hence we reach stability (Gale and Shapley 1962). We now utilize this result to build terminating procedures.

5.1 Late Discontent Suspension (LDS)

We start with an unbiased procedure that lets both sides issue proposals in turns starting from an arbitrarily selected side (Algorithm 1); at each iteration, it enforces a CC increase before it moves to the next iteration. This algorithm treats all agents on the same side equally, without discrimination: they all get a chance to propose in each round, and the outcome of their proposals is order-independent. *Proposers*, first chosen at random, propose once each. If these actions incur a CC increase, we alternate sides and move on; otherwise, we invoke a procedure that enforces CC increase; this process suspends the D state to impose that increase as a last resort, hence the name *Late Discontent Suspension* (LDS).

LDS enforces CC increase at each iteration in up to five stages. At *Stage 1*, it lets any *active* (i.e., non-idle) agent on the proposing side P issue proposals until it achieves a CC increase. If P exhausts all possible proposals without creating an additional CC couple, then all proposers have become *idle*. Then LDS enters *Stage 2*, in which it lets active *single recipients*, if such exist, act as proposers. By Theorem 5.1, this operation will incur a CC increase. Yet if single recipients do not exist, we let *discontent recipients* act, moving to *Stage 3*; with all agents on side P idle and no singles on either side, only the three kinds of action are possible:

CD → CC
CD CD → S CC S
CD CC → S CC S

The former two actions incur a CC increase. The latter does not, while it creates one new single on each side. If that happens, we enter *Stage 4*: we *suspend discontents*, adopting the acceptance criterion of Equation (3), hence create no *new* D agents or CD pairs, and let the unique single proposer issue proposals. Each such proposal either incurs CC increase, or spawns a new single proposer: S CC → CC S. Whoever is the unique single proposer, descends its preference list. Eventually, either a CC increase occurs, or the unique single proposer becomes idle, hence all proposers are idle. Thereafter, the unique single recipient acts, in *Stage 5*. With all proposers idle, by Theorem 5.1, a CC increase occurs.

5.2 Early Discontent Suspension (EDS)

We devise a simpler variant of LDS, *Early Discontent Suspension* (EDS), that suspends the discontent state whenever it forces CC increase. Algorithm 2 illustrates EDS: at *Stage 1* proposers insist on proposing; we treat all discontent receivers as *single*, employing the selective acceptance criterion of Equation (3). If Stage 1 renders all proposers idle, the baton passes to recipients in *Stage 2*. By Theorem 5.1, Stage 2 brings forth a CC increase; should we reach that stage, it will be the final stage of the overall procedure, since there is no action to be taken by the other idle side.

5.3 Permanent Discontent Ban (PDB)

EDS allows agents to be *discontent* in its regular operation. We can ban the discontent state altogether, always using the selective acceptance criterion. The resulting *Permanent Discontent Ban* (PDB) variant works as in Algorithm 2, albeit with the acceptance criterion always set to *selective*.

5.4 PowerBalance

As the following theorem shows, the hitherto proposed algorithms have *cubic* worst-case complexity.

Theorem 5.2. LDS, EDS, and PDB terminate in $O(n^3)$.

Proof. All variants incur $O(n)$ rounds of CC increase, each of which may need $O(n^2)$ proposals; $O(n^3)$ in total. □

Algorithm 1 Late Discontent Suspension

while not (everyone is Content) **do**
  Acceptance_Criterion = Non-selective (Equation (1))
  for all $p \in P$ **do** ▷ First Try
    propose(p) (propose to next preference at κ_p)
  if (CC did not increase) **then** FORCE_INCREASE(P)
  swap P and Q (swap Proposing side)

procedure FORCE_INCREASE(P)
  while $\exists$ active $p \in P$ **do** ▷ Stage 1
    propose(p)
    if (CC increased) **then return**
  while $\exists$ active Single $q \in Q$ **do** (Q is the other side) ▷ Stage 2
    propose(q)
    if (CC increased) **then return**
  while $\exists$ Discontent $q \in Q$ and $\nexists$ active Single $p \in P$ **do** ▷ Stage 3
    propose(q)
    if (CC increased) **then return**
  Acceptance_Criterion = Selective (Equation (3))
  while $\exists$ active Single $p \in P$ **do** ▷ Stage 4
    propose(p)
    if (CC increased) **then return**
  while $\exists$ active Single $q \in Q$ **do** ▷ Stage 5
    propose(q)

Algorithm 2 Early Discontent Suspension (Force Increase)

procedure FORCE_INCREASE(P)
  Acceptance_Criterion = Selective (Equation (3))
  while $\exists$ active $p \in P$ **do** ▷ Stage 1
    propose(p)
    if (CC increased) **then return**
  while $\exists$ active $q \in Q$ **do** ▷ Stage 2
    propose(q)

For the sake of completeness, we also develop a biased quadratic-time procedure. We revisit the subroutine enforcing CC increase in EDS and PDB (Algorithm 2). In this subroutine, if we render all agents on side P *idle* and let side Q propose, we reach termination. This procedure has quadratic complexity, as it monotonically descends all preference lists, and is biased, as it treats the two sides differently. On initially unmatched agents it degenerates to the GS algorithm: first one side becomes idle, and then the other side obtains its side-optimal matching.

We propose a variant, *PowerBalance* (PB), that calls this biased termination-enforcing procedure after reaching a *balance* among the two sides. PB operates in two phases: in first phase, it lets the two sides issue proposals for $O(n)$ rounds, without forcing CC increase; to ensure some balance among the two sides, the more advantaged side, in terms of average current preference, proposes in each iteration; that is fair, since the proposing side lowers its preferences, while the receiving side raises them. In case the first phase does not reach termination, we enter the second, biased phase that enforces termination by first rendering one side idle and then letting the other side propose. We ban Discontent agents, i.e., adapt the criterion of Equation (3), throughout PB's execution. We let the side that is initially *advantaged* in terms of average preference propose in the second, biased phase.

Theorem 5.3. PB terminates in $O(n^2)$ proposals.

Proof. The $O(n)$ initial rounds make $O(n)$ proposals each; the termination procedure makes $O(n^2)$ proposals. □

6 Experiments

We tested all algorithms on datasets drawn from distributions with diverse characteristics:

- *Uniform(U)*, with preferences created fully *at random*.
- *Gaussian(G)*, which adds *Gaussian noise* on an initial order; each agent starts with its id as score; we add to each score a random value from a Gaussian $\mathcal{N}(0, pol)$ and resort; we measure *polarity* pol as a percentage of n.
- asymmetric distributions, in which one side follows the Uniform or Gaussian model, while the other side follows a *Discrete(D)* uniform model: a percentage of agents participate in a *hot set*, all of which are more preferable than the rest; for example, the notation *UD* denotes uniform distributions on men and discrete on women.

We set the *hot set* of each *Discrete* distribution to include 40% of the agents, and the polarity of each *Gaussian* to 40% of n. All algorithms were implemented in Java[1] and tested on an Intel Xeon 2GHz CPU with 8GB RAM.

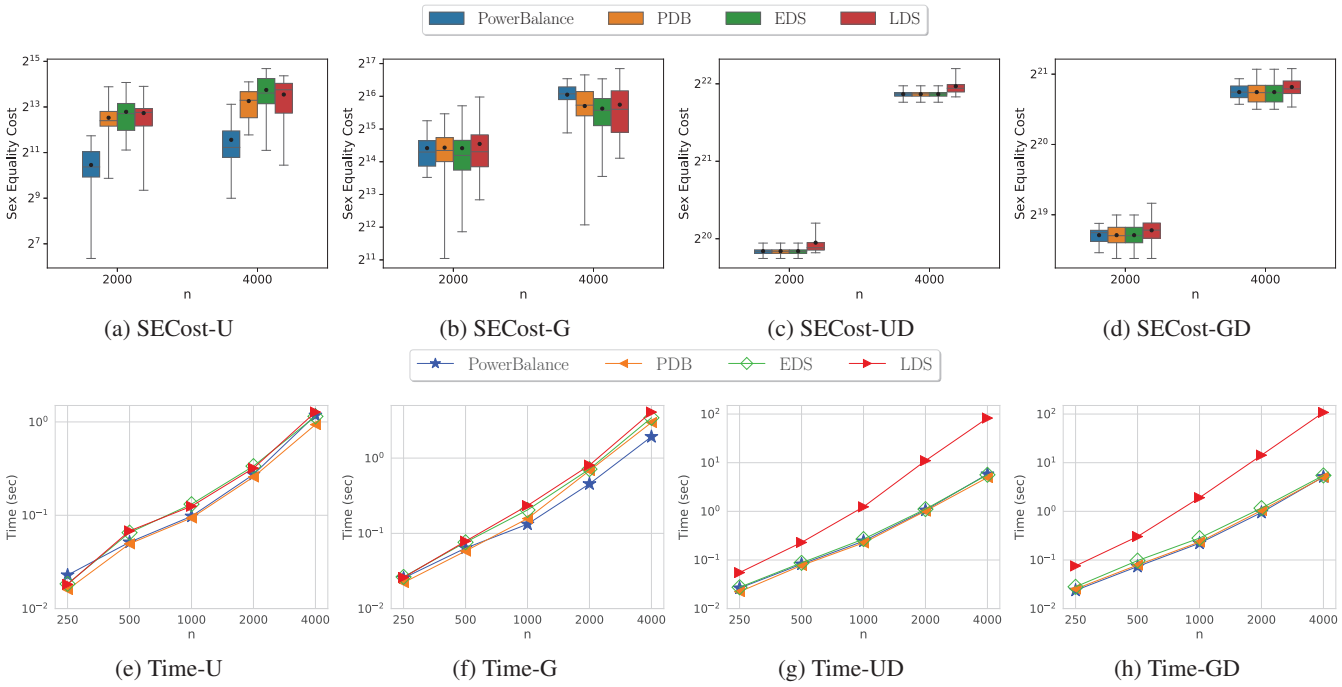
6.1 Comparison among our procedures

We first compare our procedures amongst themselves. We measure equity in terms of the sex equality cost (SECost) of Section 2.2 for 50 instances per distribution, and show box and whisker plots; a black dot denotes the mean; we also measure runtime. PowerBalance runs for $4n$ iterations before enforcing termination. Figure 1 shows our results. We observe that PowerBalance outperforms other approaches in quality on Uniform data, but not on Gaussian; on asymmetric distributions they come even. The complexity advantage of PowerBalance does not always translate to lower runtime.

6.2 Evaluation against Heuristics

We observe that our three fair procedures perform similarly to each other. We construct a single procedurally fair representative, **PF**, that reports the best solution after running all three. We assess the performance of PF and PowerBalance against heuristics with a bias and procedurally fair ones: **GS**, the Gale-Shapley algorithm that outputs the male-optimal (MaleOpt) or the female-optimal (FemOpt) solution; **PolyMin**, which finds the solutions minimizing the regret and egalitarian cost and reports the best

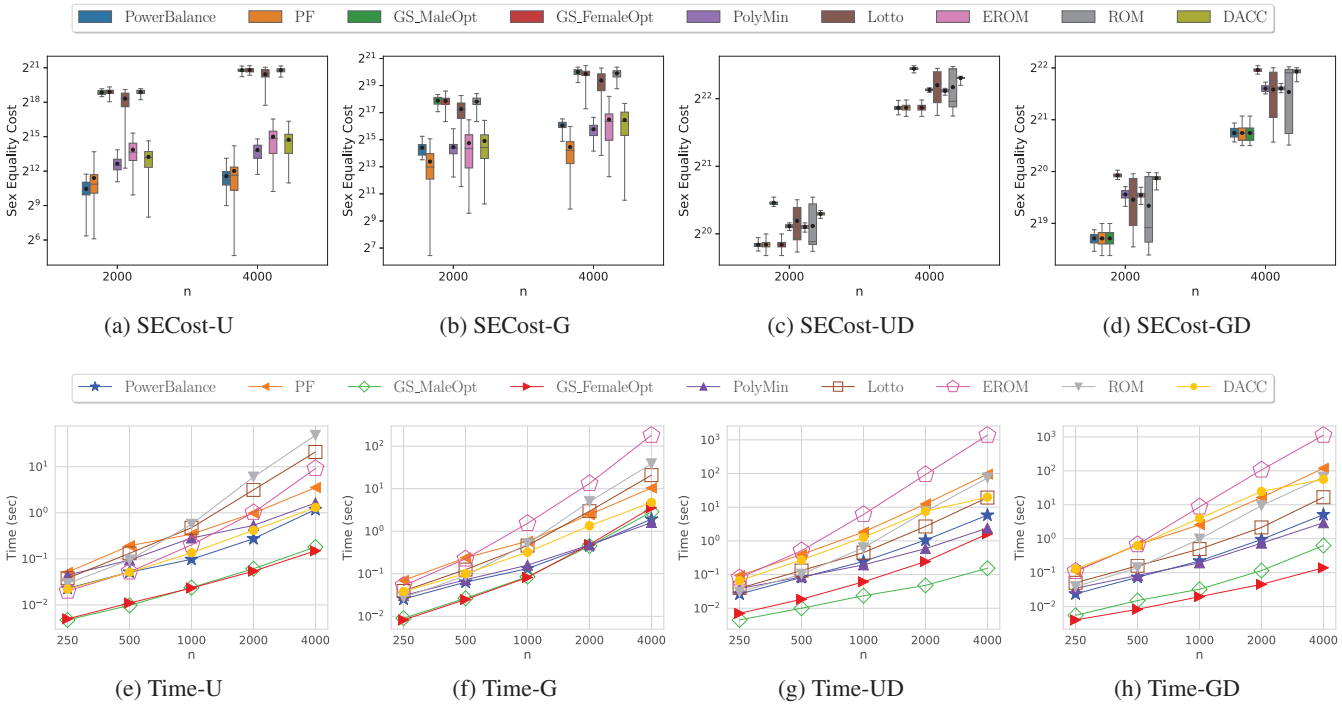
[1] Available at https://github.com/ntzia/stable-marriage.



(a) SECost-U (b) SECost-G (c) SECost-UD (d) SECost-GD

(e) Time-U (f) Time-G (g) Time-UD (h) Time-GD

Figure 1: Comparison among our procedures

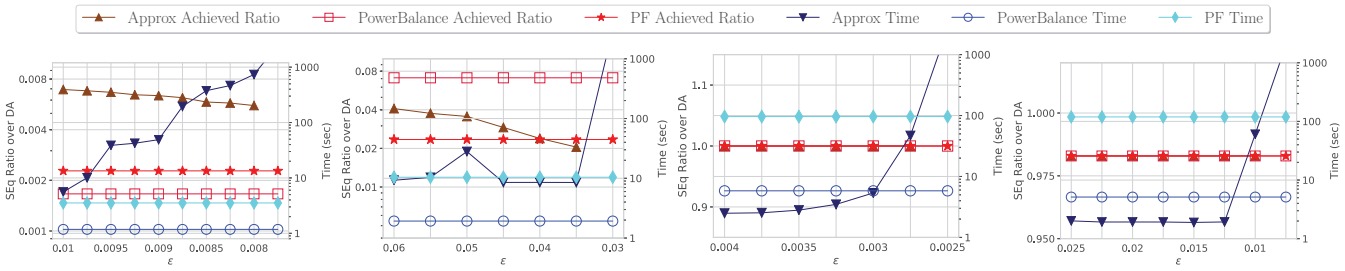


(a) SECost-U (b) SECost-G (c) SECost-UD (d) SECost-GD

(e) Time-U (f) Time-G (g) Time-UD (h) Time-GD

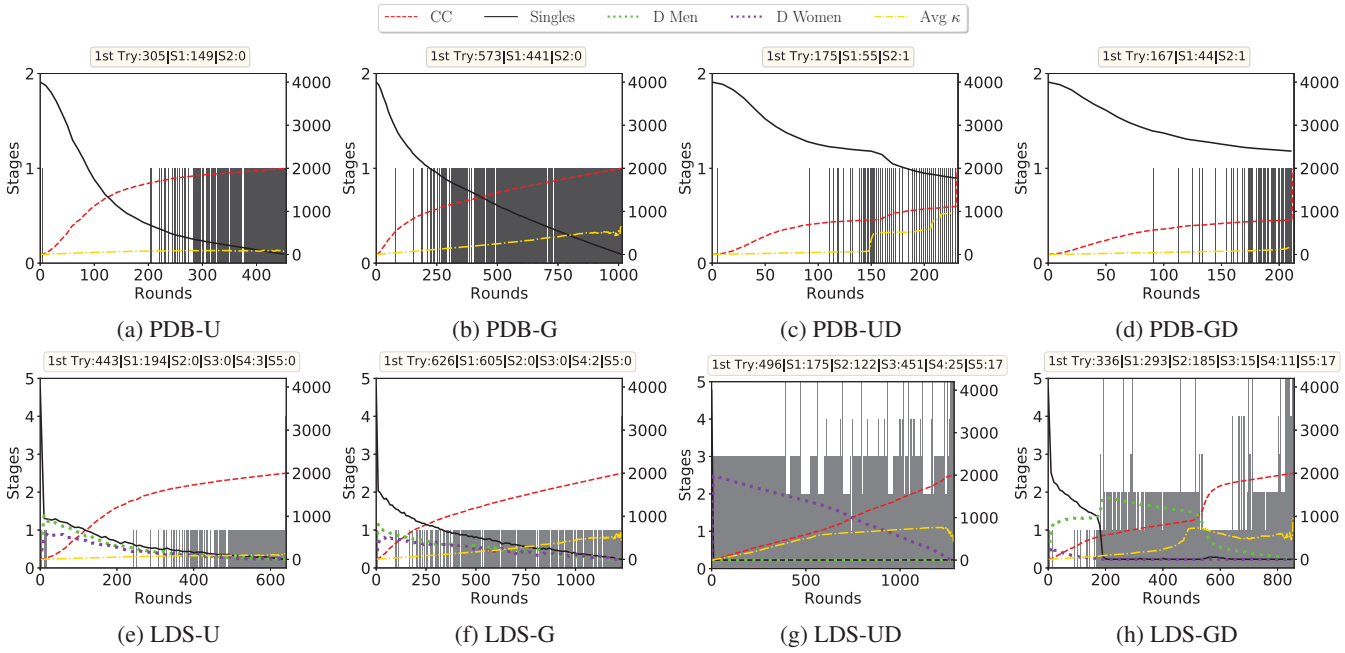
Figure 2: Evaluation against Heuristics

result; **Lotto** (Aldershof, Carducci, and Lorenc 1999), an $O(n^2)$ biased heuristic that randomly picks an agent to assign at its first remaining preference at each step; **ROM** (Ma 1996), an $O(n^3)$ biased heuristic that starts out with an *empty* matching and eliminates blocking pairs among an iteratively increasing subset of agents; **EROM** (Romero-Medina 2005), an $O(n^4)$ regret-minimizing procedure that lets agents propose to each other with progressive receptiveness: in round k, only proposals ranked up to k may be accepted. **DACC** (Dworczak 2016), an $O(n^4)$ fair procedure;



(a) SECost-U (b) SECost-G (c) SECost-UD (d) SECost-GD

Figure 3: Evaluation against Approximation Algorithm



(a) PDB-U (b) PDB-G (c) PDB-UD (d) PDB-GD

(e) LDS-U (f) LDS-G (g) LDS-UD (h) LDS-GD

Figure 4: Representative Instances

we let DACC switch sides as PowerBalance does.

Figure 2 shows SECost and mean runtime results on sizes of up to 4000 agents. Overall, PF and PowerBalance are highly competitive in terms of equity, with predictable runtimes. In case of asymmetry among the two sides, one of the two GS solutions yields a good SECost outcome.

6.3 Evaluation against APPROX

We now compare vs. the scheme of Iwama et al. (2010), APPROX, where ε provides a guarantee with respect to the best SECost achieved by the GS algorithm. For each tested size and distribution, we generate 50 instances and explore the range of ε to find appropriate values. Figure 3 presents our results. The axes on the left denote cost ratios (for APPROX, upper-bounded by ε), while those on the right denote runtime. Remarkably, PF matches or outperforms APPROX in all cases, and PB also does so except on Gaussian. The three contestants come even on asymmetric distributions, yet APPROX needs too high runtime with ill-chosen ε.

6.4 Representative Instances

We now study the runtime behavior of our procedures using data sets of 2000 agents. On each data, we show the instance of median SECost. We focus on PDB and LDS. Figure 4 depicts our runtime monitoring results. Bars show, on the y-axis, the maximum stage reached to achieve CC increase per iteration. Red dotted lines show the non-decreasing CC number. Red solid lines show the number of singles S. Green and pink lines show the number of discontents D per side, if any. At all times, $n = 2\mathtt{CC} + \mathtt{S} + \mathtt{D}$. We also plot the average preference (i.e., the κ index) among active agents. The legend above each plot shows how many times CC increase was achieved at each stage. Figure 4a shows a typical PDB behavior. Initially, CC grows rapidly; later, the algorithm struggles to achieve new CC pairings. Figure 4c shows that PDB ends forcefully by entering Stage 2 after one side is rendered idle. On LDS, the case of Uniform preferences (Figure 4e) exhibits again seamless progress. In Figure 4g, men have uniform preferences for women, hence many CD

pairs are easily created and all men become content. Thereafter, the initiative lies primarily with discontent women. Figure 4h shows three distinct phases; initially the process is smooth, as the existence of both *hot set* women and some highly preferable men allows the creation of CC pairings. Yet these are soon exhausted; by round 180, proposals by less desirable women create many discontent men. These men recuperate after round 520, causing a rapid CC increase.

7 Conclusion

The number of stable two-sided matchings in real-world applications is large, while the Gale-Shapley algorithm usually yields outcomes of suboptimal equity. A *fair procedure* should operate without coercion or bias towards any agent or side. We designed the first fair procedures that terminate in cubic time. Our experimental study demonstrates that these procedures achieve consistently higher equity than competing heuristics and can even outperform a computationally demanding approximation algorithm that lacks procedural fairness. Further, we proposed a quadratic-time algorithm lacking procedural fairness; we build upon it in (Tziavelis et al. 2019). In the future, we will examine the cases of incomplete lists and ties (Irving, Manlove, and O'Malley 2009).

References

Aldershof, B.; Carducci, O. M.; and Lorenc, D. C. 1999. Refined inequalities for stable marriage. *Constraints* 4(3):281–292.

Blum, Y.; Roth, A. E.; and Rothblum, U. G. 1997. Vacancy chains and equilibration in senior-level labor markets. *Journal of Economic Theory* 76(2):362–411.

Dworczak, P. 2016. Deferred acceptance with compensation chains. In *ACM EC*, 65–66.

Dworczak, P. 2019. Private Communication.

Everaere, P.; Morge, M.; and Picard, G. 2013. Minimal concession strategy for reaching fair, optimal and stable marriages. In *AAMAS*, 1319–1320.

Feder, T. 1992. A new fixed point approach for stable networks and stable marriages. *Journal of Computer and System Sciences* 45(2):233–284.

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

Gelain, M.; Pini, M. S.; Rossi, F.; Venable, K. B.; and Walsh, T. 2013. Local search approaches in stable matching problems. *Algorithms* 6(4):591–617.

Giannakopoulos, I.; Karras, P.; Tsoumakos, D.; Doka, K.; and Koziris, N. 2015. An equitable solution to the stable marriage problem. In *ICTAI*, 989–996.

Gusfield, D., and Irving, R. W. 1989. *The Stable marriage problem - structure and algorithms*. Foundations of computing series. MIT Press.

Gusfield, D. 1987. Three fast algorithms for four problems in stable marriage. *SIAM Journal on Computing* 16(1):111–128.

Hassidim, A.; Romm, A.; and Shorrer, R. I. 2017. Need vs. merit: The large core of college admissions markets.

Irving, R. W., and Leather, P. 1986. The complexity of counting stable marriages. *SIAM Journal on Computing* 15(3):655–667.

Irving, R. W.; Leather, P.; and Gusfield, D. 1987. An efficient algorithm for the "optimal" stable marriage. *Journal of the ACM* 34(3):532–543.

Irving, R. W.; Manlove, D.; and O'Malley, G. 2009. Stable marriage with ties and bounded length preference lists. *Journal of Discrete Algorithms* 7(2):213–219.

Irving, R. W. 2008. Stable matching problems with exchange restrictions. *Journal of combinatorial optimization* 16(4):344–360.

Iwama, K.; Miyazaki, S.; and Yanagisawa, H. 2010. Approximation algorithms for the sex-equal stable marriage problem. *ACM Trans. on Algorithms* 7(1):2:1–2:17.

Kato, A. 1993. Complexity of the sex-equal stable marriage problem. *Japan Journal of Industrial and Applied Mathematics* 10(1):1–19.

Liebowitz, J., and Simien, J. 2005. Computational efficiencies for multi-agents: a look at a multi-agent system for sailor assignment. *Electronic Government* 2(4):384–402.

Ma, J. 1996. On randomized matching mechanisms. *Economic Theory* 8(2):377–381.

McDermid, E., and Irving, R. W. 2014. Sex-equal stable matchings: Complexity and exact algorithms. *Algorithmica* 68(3):545–570.

McVitie, D. G., and Wilson, L. B. 1971. The stable marriage problem. *Communications of the ACM* 14(7):486–490.

Romero-Medina, A. 2005. Equitable selection in bilateral matching markets. *Theory and Decision* 58(3):305–324.

Roth, A. E., and Sotomayor, M. A. O. 1990. *Two-sided matching: a study in game-theoretic modeling and analysis*. Econometric Society monographs ; no. 18. Cambridge University Press.

Roth, A. E. 1984. The evolution of the labor market for medical interns and residents: A case study in game theory. *Journal of Political Economy* 92(6):991–1016.

Roth, A. E. 2008. What have we learned from market design? *The Economic Journal* 118(527):285–310.

Roth, A. E. 2018. Private Communication.

Tamura, A. 1993. Transformation from arbitrary matchings to stable matchings. *Journal of Combinatorial Theory, Series A* 62(2):310–323.

Teo, C.-P.; Sethuraman, J.; and Tan, W.-P. 2001. Gale-Shapley stable marriage problem revisited: Strategic issues and applications. *Management Science* 47(9):1252–1267.

Tziavelis, N.; Giannakopoulos, I.; Doka, K.; Koziris, N.; and Karras, P. 2019. Equitable stable matchings in quadratic time. In *NeurIPS*.

Viet, H. H.; Trang, L. H.; Lee, S.; and Chung, T. 2016a. A bidirectional local search for the stable marriage problem. In *ACOMP*, 18–24.

Viet, H. H.; Trang, L. H.; Lee, S.; and Chung, T. 2016b. An empirical local search for the stable marriage problem. In *PRICAI*, 556–564.