Fairly Stable Two-Sided Matching with Indifferences

Benjamin Cookson and Nisarg Shah
University of Toronto

Abstract

Stability has been a foundational criterion for two-sided matching. When agents on one side have weak preferences involving indifferences, the seminal work of Kesten and Ünver [26] proposes the *Fractional Deferred Acceptance* (FDA) algorithm for computing a fractional matching that satisfies (ex ante) stability along with a fairness criterion that ensures no discrimination among (equally-preferred) agents on one side.

We show that their algorithm can actually fail to terminate, refuting their claim of (polynomial-time) termination. Using substantially new algorithmic ideas, we develop an algorithm, *Doubly-Fractional Deferred Acceptance Via Strongly Connected Components* (DFDA-SCC), which can handle agents on both sides exhibiting indifferences and, in polynomial time, compute a fractional matching satisfying ex ante stability and no ex ante discrimination among agents on both sides.

1 Introduction

Ever since the seminal work of Gale and Shapley [17], the problem of two-sided matching has influenced not only a vast sea of academic research at the intersection of economics and computer science [33, 11], but also a wide range of real-world applications ranging from school admissions and placement of hospital residents to course allocation and centralized kidney markets [6].

The simplest formulation involves two sets of agents, N ("proposers") and M ("acceptors"), with $|N| = |M|$ and each agent $i \in N \cup M$ having preferences $\succcurlyeq_i$ over agents on the other side. The goal is to find a desirable one-to-one matching x between agents on the two sides based on their preferences. Much of the prior work assumes *strict preferences*, where each agent i has a total order $\succ_i$ over agents on the other side, and seeks *integral matchings*, where each agent is matched to a unique agent on the other side, i.e., $x \in \{0,1\}^{N \times M}$ with $\sum_{j' \in M} x_{ij'} = \sum_{i' \in N} x_{i'j} = 1$ for all $i \in N$ and $j \in M$. A celebrated example is the polynomial-time *Deferred Acceptance* (DA) algorithm by Gale and Shapley [17], which satisfies *stability*: no pair of proposer and acceptor who are not matched to each other should prefer each other over the agents they are respectively matched to. Many extensions of DA have been proposed to handle real-world nuances such as agent capacities [32], "couples constraints" [31], and decentralized implementations [34].

One such practical consideration is agents having *weak preferences* that exhibit indifferences (ties). Such indifferences are commonplace in real-world applications. For example, in a school choice program, schools prioritize students based on only a few criteria, such as the walk zone and sibling criteria [1], inducing ties among many students. When indifferences are allowed, one thread of the literature still continues to focus on integral matchings. Erdil and Ergin [14] show that, while stability already implies Pareto optimality under strict preferences, finding a stable and Pareto optimal matching is a much more involved task in the presence of indifferences. Manlove et al. [28] show that maximizing the number of agents matched in a stable matching is NP-hard in the presence of indifferences; this can be approximated up to a factor of $3/2$ [29] ($1 + 1/e$ if indifferences exist only on one side [27]).

However, when one adds *fairness* considerations to the mix, it becomes evident that one must allow a fractional matching, where $x \in [0,1]^{N \times M}$ and x_{ij} denotes the degree to which the pair of agents i and j are matched (For intuition on why exactly fractional matching is needed, see Appendix A).

This observation has inspired a fostering literature on seeking fractional matchings that are both *stable*

and *fair* in the presence of indifferences (For a more detailed look at the state of the art for current fractional matching algorithms, we include an extended discussion in Appendix B). The seminal work of Kesten and Ünver [26] studies a model in which only acceptors can have indifferences (while proposers have strict preferences), and seeks two criteria (see Section 2 for formal definitions):

- *ex ante stability*, a suitable adaptation of stability for fractional matchings demanding that no pair of agents i and j be able to even increase their degree of match by decreasing their degrees of matches to less-preferred agents; and
- *no ex ante discrimination (among proposers)*, a fairness criterion which informally requires that there should be no discrimination between two proposers being matched to an acceptor when the acceptor is indifferent between them.

Kesten and Ünver [26] define *Fractional Deferred Acceptance* (FDA), a generalization of DA that achieves both these guarantees simultaneously. While this work has been able to cleanly answer the question of what happens when there are indifferences on one side, it is much less clear how these definitions can be met with respect to both sides simultaneously when all agents have indifferences. This causes us to raise the following question.

> *When agents on both side exhibit indifferences, does there always exist a matching that is simultaneously stable (in the sense of ex ante stability) and fair (in the sense of no ex ante discrimination) to agents on both sides? If so, can it be computed in polynomial time?*

1.1 Our Contributions

Our main contribution is to answer both questions affirmatively, but our story actually begins with a closer examination of the seminal work of Kesten and Ünver [26]. In more detail, they define Fractional Deferred Acceptance (FDA) as a natural iterative procedure of proposals and rejections to find a matching satisfying both ex ante stability and no ex ante discrimination among proposers; they term this combination of axioms *strong ex ante stability*. While FDA may not terminate, a result of Alkan and Gale [2] can be used to show that it converges to the proposer-optimal strongly ex ante stable matching,[1] although Kesten and Ünver [26] give a direct proof of this. Let us refer to this matching as the FDA matching.

The reason that FDA may not terminate is that it can get stuck in an infinite loop wherein agents in a cycle keep proposing/rejecting to the next agent in the cycle, but the degree of match being proposed/rejected diminishes over time, leading to convergence. Kesten and Ünver [26] design an algorithm that detects such a cycle when it forms and immediately "jumps" to the matching that infinitely many proposals/rejections along the cycle would lead to. They claim that this algorithm, which we refer to as FDA-Cycle, finds the FDA matching in polynomial time.

Our first significant contribution is to show that this is incorrect. We present an example where even the FDA-Cycle ends up in an infinite loop, despite "resolving cycles" immediately as they arise. This makes polynomial-time (or even finite-time) computation of a strongly ex ante stable matching an open question once again. We resolve this positively, while extending the model of Kesten and Ünver [26] to allow indifferences and achieve no ex ante discrimination on *both sides*.

Second, we define Doubly-Fractional Deferred Acceptance (DFDA), a natural iterative procedure similar to FDA, but which incorporates indifferences on both sides. We show that DFDA satisfies ex ante stability, no ex ante discrimination among both proposers and acceptors, and a fourth axiom we term *ex*

[1] This means a strongly ex ante stable matching that is weakly ordinally preferred to every other strongly ex ante stable matching by every proposer simultaneously.

ante indifference neutrality; we term the combination of all four axioms *doubly-strong ex ante stability*. By invoking the framework of Alkan and Gale [2], we show that, while DFDA may not terminate, it converges to a proposer-optimal doubly-strong ex ante stable matching.

Next, we design our polynomial-time algorithm. The insight we obtain from our counterexample to FDA-Cycle is that it is not sufficient to resolve one *cycle* at a time. Instead, our algorithm detects entire *strongly connected components* as they arise (or even before they arise), and resolves them by jumping to their resultant matching. We show that this algorithm, which we term DFDA-SCC, in fact terminates in polynomial time and returns a doubly-strong ex ante stable matching; this is our main contribution with a highly intricate proof. There is one key difference between our work and that of Kesten and Ünver [26]. While FDA-Cycle exactly mimics (a serialization of) FDA and thus converges to the same matching (despite failing to converge), DFDA-SCC does not exactly mimic DFDA. Despite significant effort, we are unable to prove that it returns the same matching that DFDA converges to (or at least a proposer-optimal matching), but conjecture this to be the case. We discuss this issue in Section 6.

2 Preliminaries

For $k \in \mathbb{N}$, define $[k] \triangleq \{1, \dots, k\}$. There are two sets of agents, N and M, with $|N| = |M|$. We use i, i', i'' to refer to agents in N, called *proposers*, and j, j', j'' to refer to agents in M, called *acceptors*. Each proposer $i \in N$ has *weak preferences* over acceptors in M given by a strict weak ordering $\succcurlyeq_i$, which partitions M into equivalence classes $E_i = \{E_{i1}, \dots, E_{ik_i}\}$, for some $k_i \in \mathbb{N}$, such that: (i) for all $t \in [k_i]$, proposer i is *indifferent* between all acceptors $j, j' \in E_{it}$, denoted by $j \sim_i j'$, and (ii) for all $t, t' \in [k_i]$ with $t > t'$, proposer i *strictly prefers* any acceptor $j \in E_{it}$ to every acceptor $j' \in E_{it'}$, denoted by $j \succ_i j'$. Similarly, each acceptor $j \in M$ has a strict weak ordering $\succcurlyeq_j$, which partitions N into equivalence classes $E_j = \{E_{j1}, \dots, E_{jk_j}\}$, for some $k_j \in \mathbb{N}$, such that $i \sim_j i'$ for all $t \in [k_j]$ and $i, i' \in E_{jt}$ and $i \succ_j i'$ for all $t, t' \in [k_j]$ with $t < t'$, $i \in E_{jt}$, and $i' \in E_{jt'}$. An ordinal two-sided matching problem is given by the four-tuple $(N, M, \succcurlyeq_N = (\succcurlyeq_i)_{i \in N}, \succcurlyeq_M = (\succcurlyeq_j)_{j \in M})$.

When $k_i = |M|$ for each $i \in N$ (i.e., there are no indifferences), we say that the proposers have strict preferences; when $k_j = |N|$ for each $j \in M$, we say that the acceptors have strict preferences; and when both are true, we say that both sides have strict preferences.

A *(fractional) matching* $x \in [0,1]^{N \times M}$ is a doubly stochastic matrix satisfying $\sum_{j' \in M} x_{ij'} = \sum_{i' \in N} x_{i'j} = 1$ for all $i \in N$ and $j \in M$. We refer to row $x_i = (x_{ij})_{j \in M}$ as the matching of proposer $i \in N$ and column $x_j = (x_{ij})_{i \in N}$ as the matching of acceptor $j \in M$. We also denote $|x_i| \triangleq \sum_{j \in M} x_{ij}$ and $|x_j| \triangleq \sum_{i \in N} x_{ij}$. When $x_{ij} \in \{0, 1\}$ for all $i \in N$ and $j \in M$, we refer to the matching as *integral*.

2.1 Stability and Fairness Criteria

Our starting point is stability and fairness criteria for fractional matching introduced by Kesten and Ünver [26].

Definition 1 (Ex ante stability). *A matching x is* ex ante stable *if there are no $i, i' \in N$ and $j, j' \in M$ such that $j \succ_i j'$, $i \succ_j i'$, $x_{ij'} > 0$, and $x_{i'j} > 0$. In words, no pair of proposer i and acceptor j should both be positively matched to agents they prefer less than each other.*

For integral matchings, ex ante stability coincides with the popular stability criterion of Gale and Shapley [17]; all stable integral matchings are ex ante stable, but there are often additional fractional matchings that are ex ante stable as well. For fractional matchings under *strict* preferences, ex ante stability coincides with *strong stability* defined and studied by Roth et al. [35]. Ex ante stability is also

same as the fractional stability criterion of Caragiannis et al. [9] for cardinal utilities, but with cardinal comparisons replaced by SD-preference relations defined above.[2]

While ex ante stability focuses on the strict portion of the preferences, the following fairness criterion focuses on indifferences.

Definition 2 (No ex ante discrimination among proposers)**.** *A matching x has* no ex ante discrimination among proposers *if there are no $i, i' \in N$ and $j, j' \in M$ such that $i \sim_j i'$, $j \succ_i j'$, $x_{ij'} > 0$, and $x_{ij} < x_{i'j}$. In words, no proposer i should receive less of acceptor j than another proposer i' while still being positively matched to an acceptor j' she prefers strictly less than j, if j is indifferent between i and i'.*

Kesten and Ünver [26] also give a name to the combination of the above two criteria.

Definition 3 (Strong ex ante stability)**.** *A matching x is strongly ex ante stable if it is ex ante stable and has no ex ante discrimination among proposers.*

Kesten and Ünver [26] assume that only acceptors in M can have indifferences while proposers in N have strict preferences. When both proposers and acceptors have weak preferences, as is the case in our general model, it is natural to symmetrically apply the no-discrimination criterion to acceptors based on indifferences in proposers' preferences. To the best of our knowledge, we are the first to consider this criterion of fairness.

Definition 4 (No ex ante discrimination among acceptors)**.** *A matching x has* no ex ante discrimination among acceptors *if there are no $i, i' \in N$ and $j, j' \in M$ such that $j \sim_i j'$, $i \succ_j i'$, $x_{i'j} > 0$, and $x_{ij} < x_{ij'}$. In words, no acceptor j should receive less of proposer i than another acceptor j' while still being positively matched to a proposer i' she prefers strictly less than i, if i is indifferent between j and j'.*

The two no-discrimination criteria stipulate desired behavior when indifferences on one side interact with strict preferences on the other side. While the no-discrimination criterion addresses conditions where indifferences on one side interact with strict preferences on the other side, the following criterion that we introduce addresses conditions where indifferences on the two sides interact with each other. See Section 6 for additional discussion about this criterion.

Definition 5 (Ex ante indifference neutrality)**.** *A matching x is* ex ante indifference neutral *if there are no $i, i' \in N$ and $j, j' \in M$ such that $j \sim_i j'$, $i \sim_j i'$, $x_{ij} < \min\left\{x_{ij'}, x_{i'j}\right\}$. In words, if proposer i and acceptor j prefer each other as much as they prefer acceptor j' and proposer i', respectively, then they should be matched to a degree at least as much as the degree of match between either i and j' or i' and j. When agents have an innate preference to balance their degrees of matches to equally-preferred agents,[3] this makes sense: in case of the above violation, proposer i and j would "deviate" to increase x_{ij} to at least $\min\left\{x_{ij'}, x_{i'j}\right\}$ as this would leave them both happier by increasing their balance.*

When a matching meets all four criteria above, we call it *doubly-strong ex ante stable.*

Definition 6 (Doubly-strong ex ante stability)**.** *A matching x is* doubly-strong ex ante stable *if it is ex ante stable, has no ex ante discrimination among proposers and among acceptors, and is ex ante indifference neutral.*

2.2 Proposer-Optimal Matchings

Deferred-acceptance style algorithms often find a matching that not only satisfies desirable stability and fairness criteria but is in fact "proposer-optimal" among such matchings. This is formalized using

[2] Based on common nomenclature, this would be called SD-fractional-stability.

[3] This is in fact formalized when we make a connection to the result of Alkan and Gale [2] and impose a preference for balancedness to turn the weak preferences strict.

ordinal dominance. First, we extend agents' preferences over individual agents to preferences over fractional matches using the (first-order) stochastic dominance (SD) relation.

Definition 7 (SD-preferences)**.** *For proposer $i \in N$ and two fractional matches $x_i, y_i \in [0,1]^M$, we say that i weakly SD-prefers x_i to y_i, denoted $x_i \succcurlyeq_i^{SD} y_i$, if, for each $j \in M$, we have that $\sum_{j' \in M : j' \succcurlyeq_i j} x_{ij'} \geqslant \sum_{j' \in M : j' \succcurlyeq_i j} y_{ij'}$. We say that i strictly SD-prefers x_i to y_i, denoted $x_i \succ_i^{SD} y_i$, if $x_i \succcurlyeq_i^{SD} y_i$ holds and at least one of its defining inequalities is strict. SD-preferences of each acceptor $j \in M$ are defined symmetrically.*

Next, we use the SD-preference relation to define ordinal dominance.

Definition 8 (Ordinal dominance for the proposers)**.** *Given two matchings $x, y \in [0,1]^{N \times M}$, we say that x* ordinarily dominates *y for the proposers, denoted $x \succcurlyeq_N^{SD} y$, if $x_i \succcurlyeq_i^{SD} y_i$ for each $i \in N$.*

Ordinal dominance can then be used to define a "best matching" for proposers within a set of matchings.

Definition 9 (Proposer-optimality)**.** *Given a set X of matchings, a matching $x \in X$ is proposer-optimal within X if, for every $y \in X$, we have that $x \succcurlyeq_N^{SD} y$.*

In general, it is possible that there is no proposer-optimal matching within X. Interestingly, though, the sets of strongly ex ante stable matchings and doubly-strong ex ante stable matchings always admit a proposer-optimal matching; Kesten and Ünver [26] establish the former and Theorem 1 establishes the latter.

3 The Fault in Our Stars: Strong Ex Ante Stability in Finite Time?

Our story begins with the seminal work of Kesten and Ünver [26], who study fractional matchings in the presence of indifferences in acceptors' preferences, define strong ex ante stability (the combination of ex ante stability and no ex ante discrimination among proposers), and identify *Fractional Deferred Acceptance* (FDA), a natural adaptation of (integral) Deferred Acceptance (DA) of Gale and Shapley [17], which produces a fractional matching provably satisfying strong ex ante stability. While we will not present all the formal details of their work, we must present enough for the reader to understand our first significant contribution, which is to identify (and, in later sections, fix) a major flaw in the main contribution of Kesten and Ünver [26].

Algorithm FDA. A formal description of the FDA algorithm is presented as Algorithm 2 in Appendix C. Informally, it is an iterative process, which starts with an empty matching and every proposer having a free weight of 1. In each iteration, all proposers simultaneously propose their free weight to their respective most-preferred acceptors who have not yet rejected any of their proposals (even fractionally).[4] Then, each acceptor whose sum of matched weight and total proposed weights exceeds 1 rejects enough proposed weight such that this sum reduces to 1. The rejections happen in a water-filling manner—from the least preferred equivalence class to the most preferred, and within each equivalence class, at an equal rejection pace to all the highest-matched proposers at any given time. At the end, all unrejected proposed weights get added to the current fractional matching and all rejected proposed weights return to those proposers as free weights, which they propose in subsequent iterations.

The procedure is quite natural, but Kesten and Ünver [26] observe that it has a critical flaw: there may be a cycle of agents $i_1 \to j_1 \to i_2 \to \ldots i_k \to j_k \to i_1$ such that in some iteration, i_1 proposes some weight to j_1, who rejects some matched weight with i_2; so in the next iteration, i_2 proposes some weight to j_2, who rejects some matched weight with i_3; at some point, j_k rejects some matched weight

[4] Recall that in their model, proposers have *strict preferences*, so such an acceptor is unique for each proposer.

with i_1, who then proposes to j_1 again; and this can continue indefinitely. Due to such cycles, which they term *rejection cycles*, FDA may never terminate.

Nonetheless, they observe that by viewing FDA as a specific instantiation of a more general two-sided "schedule matching" process studied by Alkan and Gale [2], one can easily conclude that FDA *converges* to a matching—henceforth, the FDA matching—that is strongly ex ante stable, and, in fact, proposer-optimal within the set of such matchings.[5] This still leaves the issue of finite-time computation of a strongly ex ante stable matching, leading to their main contribution.

Algorithm FDA-Cycle. They propose an algorithm, which we refer to as FDA-Cycle (presented formally as Algorithm 4 in Appendix C), which *allegedly* computes the FDA matching in polynomial time. First, they notice that the proposals can be serialized as the resulting matching is still unique and independent of the order of proposals (see their Corollary 1). In this serialized process, the infinitely many proposals and rejections along any rejection cycle can be viewed as consecutive iterations. However, instead of executing these infinitely many iterations, FDA-Cycle detects a rejection cycle as soon as it forms and directly computes the matching that these infinitely many iterations would have converged to in finite time.

Formally, FDA-Cycle keeps track of a *rejection graph*, which is a directed graph with the agents as nodes and edges $i \to j \to i'$ exist (for all $i, i' \in N$ and $j \in M$) whenever $x_{ij} > 0$, $x_{i'j} > 0$, i has never been rejected by j, and i' has been rejected by j.[6] Intuitively, this tells us that whenever i proposes to j, j will reject some fraction of i'.[7] FDA-Cycle monitors this graph, and as soon as a directed cycle forms, it solves a linear program to compute the matching that infinitely many proposals and rejections across the cycle would converge to, "resolving" the cycle (temporarily).

Erroneous claim. Kesten and Ünver [26] claim in their Proposition 3 that FDA-Cycle terminates after a finite number of steps. Our first significant contribution is to show that this is incorrect. The issue lies in the last paragraph of their proof, presented in their Appendix B, which makes the following (rephrased) claim: "after all proposers make proposals, at least one proposer is rejected by one acceptor and has an outstanding fraction, or the algorithm converges, whether or not a [rejection] cycle occurs. Since there are $|N|$ proposers and $|M|$ acceptors, the algorithm converges after at most $|N||M|$ steps". It is not clear what they mean by a proposer being rejected by an acceptor, but the latter conclusion would hold if they mean a proposer is rejected by an acceptor either *for the first time* or *fully* (i.e., making their degree of match 0). It turns out that the former statement does not hold under either interpretation.

Our counterexample. Our first significant contribution is a counterexample in which FDA-Cycle in fact fails to terminate, thus precluding the possibility of an alternative proof of its finite-time convergence. We emphasize that significant effort and careful analysis went into designing this counterexample. Due to the length of the argument, we defer the exact counterexample instance to Appendix D.

In short, at one point during the execution of FDA-Cycle, a rejection cycle forms, which the algorithm resolves. Crucially, after the resolution, the cycle remains in the graph, albeit with no free weights left on the proposers. In the subsequent iterations, new edges get added to the rejection graph, causing another rejection cycle to form. Again, after the algorithm resolves this cycle, it remains in the graph. Later in the algorithm, a proposal occurs that requires re-resolving one of these cycles, which leads to another proposal that requires re-resolving the other cycle. Re-resolving the second cycle directly leads to re-resolving the first cycle again, essentially creating a cycle of cycles. FDA-Cycle then continues for infinitely many steps.

[5] They provide an independent proof for this too.

[6] Actually, in the rejection graph of Kesten and Ünver [26], only the proposers are nodes, and instead of edges $i \to j \to i'$, they add an edge $i \xrightarrow{j} i'$ labeled with the acceptor j; these are equivalent representations.

[7] This is because j having rejected i' previously implies that j must be fully matched at the moment, so accepting any proposed weight requires it to reject some existing weight, and once it rejects agent i', it continues to do so until $x_{i'j} = 0$.

This reopens the question of finite-time computation of a strongly ex ante stable matching. The main contribution of the next two sections is to uncover a novel insight that lets us overcome the limitation of FDA-Cycle and design a novel polynomial-time algorithm, DFDA-SCC, which in fact computes a doubly-strong ex ante stable matching in the presence of indifferences on both sides.

Our counterexample highlights the key issue: when multiple cycles have paths to each other (i.e., they are part of the same strongly connected component), they can keep "reactivating" each other. This suggests that the right approach is to not resolve one cycle at a time, but rather resolve entire strongly connected components in one shot, which is precisely what we do later in Section 5.

4 Doubly-Fractional Deferred Acceptance

Before we present a polynomial-time algorithm, we take a slight detour and extend the model of Kesten and Ünver [26] to allow indifferences on both sides, not only in acceptors' preferences. The first step is to extend their infinite iterative procedure, Fractional Deferred Acceptance (FDA). We term our procedure *Doubly-Fractional Deferred Acceptance* (DFDA), and show that it produces a proposer-optimal doubly-strong ex ante stable matching via a reduction to the framework of Alkan and Gale [2]. Then, in the next section, we design our DFDA-SCC algorithm, which somewhat mimics DFDA, resolves one strongly connected component (SCC) in each iteration, and provably terminates at a doubly-strong ex ante stable matching in polynomial time.

DFDA, (which is formally presented in Appendix C as Algorithm 3), is almost identical to FDA, with a simple and natural change to account for possible indifferences in the proposers' preferences. Recall that in FDA, each proposer proposes all her free weight to the most-preferred acceptor who has not rejected any fraction of her, and this acceptor is unique due to strict preferences. In DFDA, each proposer considers the set of all (equally) most-preferred acceptors who have not rejected any fraction of her—note that they must all be part of the same equivalence class—and proposes to all of them simultaneously, evenly splitting her free weight between them. Thus, each iteration of DFDA witnesses both proposers proposing to multiple acceptors simultaneously and acceptors rejecting multiple proposers simultaneously, which explains the name of the algorithm. DFDA is a strict generalization of FDA, reducing to FDA when proposers have strict preferences.

We establish the desired properties of DFDA by invoking the general framework of Alkan and Gale [2]. They study two-sided fractional matching under a broad class of preferences, given by the so-called (strict) "choice functions". They prove that (1) the set of stable matchings—with a specific stability definition that we refer to as *AG-stability*—form a lattice structure, which admits a unique proposer-optimal matching under the strict choice-functions-based preferences; and (2) a natural iterative procedure converges to this unique proposer-optimal AG-stable matching. We take the weak preferences of proposers and acceptors and impose a secondary preference for "balancedness" to induce strict choice functions under which (1) AG-stability becomes equivalent to doubly-strong ex ante stability, thus establishing the existence of a proposer-optimal doubly-strong ex ante stable matching under the strict choice functions, and (2) the iterative procedure of Alkan and Gale [2] becomes equivalent to DFDA, which finds the aforementioned matching. This yields the following result; a formal proof, along with an introduction to the framework of Alkan and Gale [2], is given in Appendix E.

Theorem 1. *DFDA converges to a proposer-optimal doubly-strong ex ante stable matching.*

5 A Polynomial-Time Algorithm for Doubly-Strong Ex Ante Stable Matching

Because DFDA coincides with FDA when proposers happen to have strict preferences, clearly we cannot resolve rejection cycles one at a time, otherwise we would have the same non-termination issue as FDA-Cycle on our counterexample from Section 3. Based on the insight obtained from our

counterexample, we propose a new algorithm, DFDA-SCC (Algorithm 1), which circumvents this issue by resolving an entire strongly connected component (SCC) in each iteration.

Our contribution lies not only in the design of this algorithm, but also in its analysis. For Kesten and Ünver [26], it is easy to establish equivalence between FDA-Cycle and FDA because FDA-Cycle exactly follows a serialization of FDA, simply skipping-forward intermediate blocks of infinitely many iterations across individual rejection cycles. Unfortunately, this is not the case for DFDA-SCC: it is possible that one of its intermediate matchings may never be produced during any serialization of DFDA. It is still possible that DFDA-SCC is equivalent to DFDA by eventually producing the same matching (which would establish its proposer-optimality), but, sadly, we are unable to prove so and leave this as an open question. This is discussed in Section 6. Nonetheless, we are able to establish doubly-strong ex ante stability of DFDA-SCC, in addition to polynomial-time convergence. This is our main result with an intricate proof.

Theorem 2. *DFDA-SCC (Algorithm 1) terminates in polynomial time and returns a doubly-strong ex ante stable matching.*

Description of DFDA-SCC. Let us describe what DFDA-SCC (Algorithm 1) intuitively does.

The basis of our algorithm is the *proposal graph*, a directed bipartite graph in which there is a node for each proposer and acceptor. Each proposer i has directed edges to her most-preferred acceptors who have not yet rejected her; these are the acceptors she will propose to next. Each acceptor j has directed edges to her least-preferred proposers that she is matched to (and among this set, the ones who currently have the highest degree of match to j); if j wishes to fractionally reject existing matches to accept proposals from more-preferred proposers, these are the proposers she will start rejecting.

In each iteration, the algorithm partitions the proposers into groups based on whether they are part of the same strongly connected component (SCC) of the current proposal graph and sorts these groups according to the topological order of the proposal graph. Then, it "resolves" each group (and its corresponding SCC) via a linear program (LP). For any proposer i, the SCC of the proposal graph that contains i is guaranteed to include all cycles containing i, as well as some "higher-order cycles" that cause infinite loops like in Section 3.

The algorithm terminates when all proposers have no free weight remaining, which we will prove must occur after a polynomial number of iterations.

Description of LP-SCC. The linear program at the heart of each step of the algorithm is shown in Figure 1. At a high level, this LP works by maximizing the amount of total weight proposed for a given connected component, while being constrained by the expected rules that dictate how proposals and rejection work in DFDA as well as additional conditions to ensure that the LP simulates proposal/rejection only up to the point where the proposal graph would change.

In more detail, the main variables in the LP are y-s and z-s. For each $i \in N$, y_i denotes the total weight that i proposes in the current iteration, of which y_{ij} denotes the weight proposed to $j \in M$. Similarly, for each acceptor $j \in M$, z_j denotes the total weight rejected by j in the current iteration, of which z_{ji} denotes the weight that j rejects from i.

The first four constraints dictate how proposers can propose.

- Constraints (1) and (2)—$\forall i \in C_t, y_i \leqslant \sum_{j \in M} z_{ji} + w_i$ and $\forall i \notin C_t, y_i = 0$—ensure that only proposers from the current SCC C_t being resolved propose, and they propose weight that is at most the sum of their free weight and their total rejected weight from the current iteration (that is, they cannot propose more weight than they have). The inequality rather than a strict equality in Constraint (1) allows proposers to retain some free weight in the end, which is key to solving the problem as a series of continuous flow problems rather than a series of discrete proposal-rejection sequences.

Algorithm 1: DFDA-SCC

$\forall i \in N, w_i \leftarrow 1$ and $P_i \leftarrow E_{i1}$ // Free weight of i and acceptors i will propose to next
$\forall i \in N, j \in M, x_{ij} \leftarrow 0$ // Current matching
$G \leftarrow \{(i,j) : j \in P_i\}$ // Proposal graph with only proposing edges, no rejections
while $\exists i \in N, w_i > 0$ **do**
  // Key step: SCC decomposition of the rejection graph
  $C_1, \ldots, C_k \leftarrow$ Partition N into strongly connected components based on G, sorted topologically
  for $t \in [k]$ **do**
    if $\exists i \in C_t$ such that $w_i > 0$ **then**
      // Resolve the SCC via an LP and update the matching
      $y^*, z^* \leftarrow$ An optimal solution to the linear program LP-SCC (given in Figure 1) for C_t
      for $i \in N$ **do**
        for $j \in M$ **do**
          $x_{ij} \leftarrow x_{ij} + y^*_{ij} - z^*_{ji}$ // Update matching
        end
        $w_i \leftarrow \sum_{j \in M} z^*_{ji} + w_i - y^*_i$ // Update free weights
      end
      // Collect information for updating acceptors' edges
      for $j \in M$ **do**
        if $|x_j| = 1$ **then**
          $X_j \leftarrow \{i \in N : x_{ij} > 0\}$ // Proposers matched to j
          $\ell \leftarrow \max\{k : X_j \cap E_{j,k} \neq \emptyset\}$
          $A_j \leftarrow E_{j,\ell}$ // Least-preferred proposers
          $A^*_j \leftarrow \operatorname{argmax}_{i \in A_j} x_{ij}$ // Proposers from A_j with max matched weight
          $R_j \leftarrow A^*_j \cup \{i \in N : i \in E_{j,\ell'}, \ell' > \ell\}$ // Updated rejected proposers
        end
      end
      // Collect information for updating proposers' edges
      for $i \in N$ **do**
        $R_i \leftarrow \{j \in M : i \in R_j\}$ // Acceptors who have rejected i
        $\ell \leftarrow \min\{k : E_{i,k} \not\subseteq R_i\}$
        $P_i \leftarrow E_{i,\ell} \setminus R_i$ // Most-preferred acceptors who haven't rejected i
      end
      // Update the proposal graph
      $G' \leftarrow \{(i,j) : j \in P_i\} \cup \left\{(j,i) : i \in A^*_j\right\}$ // New proposal graph
      if $G' \neq G$ **then** // Proposal graph changed, restart the outer loop
        $G \leftarrow G'$
        Go to the start of the While loop (Line 4)
      end
    end
  end
end
return x

- Constraint (3) and (4)—$\forall i \in N, y_{ij} = y_i/|P_i|, \forall j \in P_i$ and $y_{ij} = 0, \forall j \notin P_i$—ensures that proposers propose only to their most-preferred acceptors who have not rejected them and

maximize $\sum_{i \in N} y_i$

subject to // Constraints on proposals

(1) $\forall i \in C_t : y_i \leqslant \sum_{j \in M} z_{ji} + w_i$

(2) $\forall i \notin C_t : y_i = 0$

(3) $\forall i \in N, \forall j \in P_i : y_{ij} = {}^{y_i}/_{|P_i|}$

(4) $\forall i \in N, \forall j \notin P_i : y_{ij} = 0$

// Constraints on rejections

(5) $\forall j \in M, |x_j| = 1 : z_j = \sum_{i \in N} y_{ij}$

(6) $\forall j \in M, |x_j| < 1 : z_j = 0$

(7) $\forall j \in M, \forall i \in A_j^* : z_{ji} = {}^{z_j}/_{|A_j^*|}$

(8) $\forall j \in M, \forall i \notin A_j^* : z_{ji} = 0$

// Constraints that stop the flow at discrete structural changes

(9) $\forall j \in M, |x_j| < 1 : \sum_{i \in N} y_{ij} \leqslant 1 - |x_j|$

(10) $\forall j \in M, \forall i \in A_j^*, \forall i' \in A_j \setminus A_j^* : x_{ij} - z_{ji} \geqslant x_{i'j} + y_{i'j}$

(11) $\forall i \in N, \forall j \in M : z_{ji} \leqslant x_{ij}$

Figure 1: Linear program LP-SCC used to resolve a strongly connected component in DFDA-SCC (Algorithm 1).

propose an equal amount to them.

The next four constraints dictate how acceptors handle the weight proposed to them.

- Constraints (5) and (6)—$\forall j \in M, z_j = \sum_{i \in N} y_{ij}$ when $|x_j| = 1$ and $z_j = 0$ when $|x_j| < 1$—stipulate that a saturated (fully matched) acceptor must reject exactly as much weight as she accepts, while a non-saturated acceptor must not reject. Constraint (9) later ensures that such an acceptor does not accept more weight than her remaining match capacity. This ensures that once an acceptor becomes saturated, they remain saturated for the rest of the algorithm.
- Constraints (7) and (8)—$\forall j \in M, z_{ji} = {}^{z_j}/_{|A_j^*|}$ for all $i \in A_j^*$ and $z_{ji} = 0$ for all $i \notin A_j^*$—ensure fair rejections. Only the least-preferred matched acceptors with the highest matched weight (those in A_j^*) are rejected, and they are rejected equally. Constraint (10) stops the LP once this highest matched weight reduces to the next-highest level, at which point a new acceptor from A_j must be added to A_j^* by the algorithm.

This leaves constraint (11)—$\forall i \in N, \forall j \in M, z_{ji} \leqslant x_{ij}$. This states that an acceptor cannot reject more weight from a proposer than it has available to reject. We do not have to consider any incoming proposed weight from i to j because, due to constraint (8), z_{ji} can only be positive if $i \in A_j^*$, in which case i will not be proposing any weight to j in this iteration, or for the remainder of the algorithm.

The LP maximizes $\sum_{i \in N} y_i$, i.e., the total amount of weight proposed by the proposers. The optimal solution (y^*, z^*) is used by DFDA-SCC to update the matching and the proposal graph.

5.1 Analysis of DFDA-SCC

We are now ready to begin proving polynomial-time termination and doubly-strong ex ante stability of DFDA-SCC (Theorem 2).

5.1.1 Proof of polynomial-time termination

The main technical lemmas we use to prove this fact revolves around a structural observation relating the last three constraints of LP-SCC to key events in the algorithm, which cause progress to be made. Specifically, all these events revolve around changes to the proposal graph, which correspond to proposers being either rejected for the first time by an acceptor, or fully rejected from an acceptor. As we will show, keeping track of such changes is crucial for arguing polynomial-time termination. We give the full proof of Lemma 1 in Appendix F.

Lemma 1. *In any solution to the LP, at least one of the following will be true:*

- *(A)* $\forall i \in C_t, y_i = \sum_{j \in M} z_{ji} + w_i$
- *(B)* $\exists j \in M, |x_j| < 1, \sum_{i \in N} y_{ij} = 1 - x_j$
- *(C)* $\exists i \in N, j \in M, x_{ij} > 0, z_{ji} = x_{ij}$.
- *(D)* $\exists j \in M, \exists i \in A_j^*, \exists i' \in A_j \setminus A_j^*, x_{ij} - z_{ji} = x_{i'j} + y_{i'j}$.

Lemma 1 intuitively states the following: After any iteration of the algorithm, one of these conditions will be true:

- (A) All proposers in C_t have no free weight.
- (B) Some acceptor that was not full at the beginning of the iteration becomes full.
- (C) Some proposer is fully rejected from some acceptor.
- (D) Some proposer is added to A_j^* for some acceptor j.

(A) is a special condition as the proposers not having any free weight is what we want to happen to ensure termination with a perfect matching. The other three conditions, (B), (C), and (D), all correspond to the previously mentioned changes in the proposal graph. We show this formally in Lemma 2.

Lemma 2. *Let y^*, z^* be the variables after resolving some LP in Algorithm 1. The process of updating the current matching using y^*, z^* will change the proposal graph only if at least one of the conditions (B), (C) or (D) are true.*

With this, we can next prove Lemma 3, which shows how the algorithm will come to terminate.

Lemma 3. *In some iteration of the main while loop in Algorithm 1, if for every component C_t of proposers, the LP run on C_t terminates with only condition (A) being true, then the matching produced by the last component being solved will be a perfect matching.*

Finally, leveraging all these technical lemmas, we can show that at each step of DFDA-SCC, progress will be made, changing the proposal graph, and allowing the algorithm to terminate after a polynomial number of iterations.

Theorem 3. *Algorithm 1 terminates in polynomial time, and will output a perfect matching.*

The proof of these lemmas, as well as the final theorem, appear in Appendix F.

5.1.2 Proof of doubly-strong ex ante stability

Finally, we will show that the perfect matching that Algorithm 1 returns in polynomial time will be doubly-strong ex-ante stable, the proof of which appears in Appendix F.

Theorem 4. *The matching produced by Algorithm 1 is doubly-strong ex-ante stable.*

Together, Theorems 3 and 4 yield the two claims made in Theorem 2, concluding its proof.

6 Discussion

While we have established polynomial-time computation of a doubly-strong ex ante stable matching, many exciting questions remain open.

The lingering issue of equivalence to DFDA and proposer-optimality. Recall that both our infinite procedure DFDA and polynomial-time algorithm DFDA-SCC produce a doubly-strong ex ante stable matching, but DFDA has the additional guarantee that its matching is proposer-optimal. Sadly, we are unable to prove proposer-optimality of DFDA-SCC or its equivalence to DFDA. However, we conjecture that the output of the two algorithms will be the same proposer-optimal matching.

Ex ante indifference neutrality and Pareto optimality. In Section 2, we remarked that it is not clear if our ex ante indifference neutrality criterion is intuitively desirable. One formal reason why it may be undesirable is that it is incompatible with Pareto optimality (or, rather, ordinal Pareto undomination). A concrete example is given in Appendix H. Kesten and Ünver [26] show that one can cyclically shift matched weights in the FDA matching to find ordinal improvements for the proposers that retain ex ante stability but introduce ex ante discrimination among proposers; this yields an ex ante stable matching that is ordinally Pareto undominated by any other ex ante stable matching. But whether true ordinal Pareto undomination (by any other matching) can be achieved, possibly while also retaining no ex ante discrimination among proposers and acceptors, remains to be seen.

> **Open Question:** When both proposers and acceptors have weak preferences, does there always exist a fractional matching that is ex ante stable, has no ex ante discrimination among proposers and acceptors, and is ordinally Pareto undominated? What if agents on one side (e.g., proposers) have strict preferences?

Ex ante indifference neutrality nonetheless plays a key role in our reduction to the framework of Alkan and Gale [2] for establishing proposer-optimality of the DFDA matching among the set of doubly-strong ex ante stable matchings (Theorem 1). We believe that it should be possible to drop ex ante indifference neutrality to make the statement stronger:

> **Conjecture:** The DFDA matching is in fact proposer-optimal within the broader set of matchings satisfying ex ante stability and no ex ante discrimination among both proposers and acceptors (but not necessarily ex ante indifference neutrality).

Intuitively, if some proposer i strictly prefers the change from the DFDA matching to another matching, then there must be a shift of matched weights across strict preferences of i, which should lead to the new matching violating one of the other three axioms.

Investigation of stable and fair two-sided matching in the presence of indifferences opens doors to many exciting research questions and connections. Due to the space constraint, we could only mention the most interesting ones above; the rest are deferred to Appendix I.

References

[1] Atila Abdulkadiroğlu, Parag A Pathak, Alvin E Roth, and Tayfun Sönmez. The boston public school match. *American Economic Review*, 95(2):368–371, 2005.

[2] Ahmet Alkan and David Gale. Stable schedule matching under revealed preference. *Journal of Economic Theory*, 112(2):289–306, 2003.

[3] Haris Aziz, Rupert Freeman, Nisarg Shah, and Rohit Vaish. Best of both worlds: Ex ante and ex post fairness in resource allocation. *Operations Research*, 72(4):1674–1688, 2023.

[4] Haris Aziz, Aditya Ganguly, and Evi Micha. Best of both worlds fairness under entitlements. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pages 941–948, 2023.

[5] G. Birkhoff. Three observations on linear algebra. *Universidad Nacional de Tucumán, Revista A*, 5: 147–151, 1946.

[6] Péter Biró. Applications of matching models under preferences. In Ulle Endriss, editor, *Trends in Computational Social Choice*, chapter 18, pages 345–373. AI Access, 2017.

[7] A. Bogomolnaia and H. Moulin. A new solution to the random assignment problem. *Journal of Economic Theory*, 100:295–328, 2001.

[8] Eric Budish, Yeon-Koo Che, Fuhito Kojima, and Paul Milgrom. Designing random allocation mechanisms: Theory and applications. *American Economic Review*, 103(2):585–623, 2013.

[9] Ioannis Caragiannis, Aris Filos-Ratsikas, Panagiotis Kanellopoulos, and Rohit Vaish. Stable fractional matchings. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 21–39, 2019.

[10] Ioannis Caragiannis, David Kurokawa, Hervé Moulin, Ariel D. Procaccia, Nisarg Shah, and Junxing Wang. The unreasonable fairness of maximum Nash welfare. *ACM Transactions on Economics and Computation*, 7(3): Article 12, 2019.

[11] Pierre-André Chiappori and Bernard Salanié. The econometrics of matching models. *Journal of Economic Literature*, 54(3):832–861, 2016.

[12] Benjamin Cookson, Soroush Ebadian, and Nisarg Shah. Constrained fair and efficient allocations. In *39th*, 2025. Forthcoming.

[13] Soroush Ebadian, Rupert Freeman, and Nisarg Shah. Harm ratio: A novel and versatile fairness criterion. In *4th*, pages 1–14, 2024.

[14] Aytek Erdil and Haluk Ergin. Two-sided matching with indifferences. *Journal of Economic Theory*, 171:268–292, 2017.

[15] Michal Feldman, Simon Mauras, Vishnu V Narayan, and Tomasz Ponitka. Breaking the envy cycle: Best-of-both-worlds guarantees for subadditive valuations. In *Proceedings of the 25th ACM Conference on Economics and Computation*, pages 1236–1266, 2024.

[16] Rupert Freeman, Evi Micha, and Nisarg Shah. Two-sided matching meets fair division. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 203–209, 2021.

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

[18] Peter Gärdenfors. Match making: assignments based on bilateral preferences. *Behavioral Science*, 20(3):166–173, 1975.

[19] Xiang Han. A theory of fair random allocation under priorities. *Theoretical Economics*, 19(3): 1185–1221, 2024.

[20] Tadashi Hashimoto, Daisuke Hirata, Onur Kesten, Morimitsu Kurino, and M Utku Ünver. Two axiomatic approaches to the probabilistic serial mechanism. *Theoretical Economics*, 9(1):253–277, 2014.

[21] Martin Hoefer, Marco Schmalhofer, and Giovanna Varricchio. Best of both worlds: Agents with entitlements. *Journal of Artificial Intelligence Research*, 80:559–591, 2024.

[22] Chien-Chung Huang and Telikepalli Kavitha. Popularity, mixed matchings, and self-duality.

Mathematics of Operations Research, 46(2):405–427, 2021.

[23] Aanund Hylland and Richard Zeckhauser. The efficient allocation of individuals to positions. *Journal of Political economy*, 87(2):293–314, 1979.

[24] A. Katta and J. Sethuraman. A solution to the random assignment problem on the full preference domain. *Journal of Economic Theory*, 131:231–250, 2006.

[25] Onur Kesten, Morimitsu Kurino, and M Utku Ünver. Fair and efficient assignment via the probabilistic serial mechanism. *Mimeographed, Boston University*, 2011.

[26] Onur Kesten and M. Utku Ünver. A theory of school-choice lotteries. *Theoretical Economics*, 10(2): 543–595, 2015.

[27] Chi-Kit Lam and C Gregory Plaxton. A (1+ 1/e)-approximation algorithm for maximum stable matching with one-sided ties and incomplete lists. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2823–2840, 2019.

[28] David F Manlove, Robert W Irving, Kazuo Iwama, Shuichi Miyazaki, and Yasufumi Morita. Hard variants of stable marriage. *Theoretical Computer Science*, 276(1-2):261–279, 2002.

[29] Eric McDermid. A 3/2-approximation algorithm for general stable marriage. In *International Colloquium on Automata, Languages, and Programming*, pages 689–700, 2009.

[30] Ioannis Panageas, Thorben Tröbst, and Vijay Vazirani. Time-efficient algorithms for Nash-bargaining-based matching market models. In *Proceedings of the 20th International Conference on Web and Internet Economics*, 2024. Forthcoming.

[31] Alvin E Roth. 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, 1984.

[32] Alvin E Roth. The college admissions problem is not equivalent to the marriage problem. *Journal of economic Theory*, 36(2):277–288, 1985.

[33] Alvin E Roth. Deferred acceptance algorithms: History, theory, practice, and open questions. *international Journal of game Theory*, 36:537–569, 2008.

[34] Alvin E Roth and John H Vande Vate. Random paths to stability in two-sided matching. *Econometrica*, pages 1475–1480, 1990.

[35] Alvin E Roth, Uriel G Rothblum, and John H Vande Vate. Stable matchings, optimal assignments, and linear programming. *Mathematics of operations research*, 18(4):803–828, 1993.

[36] Éva Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, 34(2):250–256, 1986.

[37] Thorben Tröbst and Vijay V. Vazirani. Cardinal-utility matching markets: The quest for envy-freeness, pareto-optimality, and efficient computability. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, page 42, 2024.

[38] Dietrich Weller. Fair division of a measurable space. *Journal of Mathematical Economics*, 14(1): 5–17, 1985.

[39] Xiaowei Wu, Bo Li, and Jiarui Gan. Budget-feasible maximum nash social welfare is almost envy-free. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 465–471, 2021.

Appendix

A Fractional Matching is Required For Fairness

Consider the trivial example in Figure 2(a), where two proposers, i and i', strictly prefer acceptor j to j', while the acceptors are indifferent between the proposers. Both integral matchings are stable, but assign the more preferred acceptor j exclusively to one of the proposers (see Figure 2(b)), which is unfair to the proposers as they are indistinguishable and, hence, should be treated equally. The fractional matching in Figure 2(c) that equally shares j (and, thus, also j') between the two proposers is the only fair outcome. Note that this can be implemented as a lottery over the two integral stable matchings.

Preferences	
i, i':	$j \succ j'$
j, j':	$i \sim i'$

(a) Preferences with ties.

	j	j'
i	1	0
i'	0	1

(b) Unfair integral stable matching.

	j	j'
i	$1/2$	$1/2$
i'	$1/2$	$1/2$

(c) Fair fractional stable matching.

Figure 2: Indifferences mandate fractional matchings for fairness.

B Extended Related Work

Apart from the work already cited in the introduction, there are a few threads of related work that our DFDA should be contrasted against.

Matching under weak preferences. Han [19] extends the celebrated *Probabilistic Serial* (PS) algorithm for one-sided matching (of agents to objects) to two-sided matching, assuming agents (proposers) to have strict preferences but allowing objects (acceptors) to have weak preferences. This unifies PS with FDA, retaining ex ante stability but sacrificing no ex ante discrimination for *ordinal fairness*, a criterion that plays a key role in characterizations of PS [25, 20]. This complements the work of Katta and Sethuraman [24] which extend PS by allowing agents to have weak preferences but still assuming objects to have no preferences (equivalently, assuming every object to be indifferent between all the agents). To the best of our knowledge, there is no known extension of PS to two-sided matching in the full domain where both agents and objects have weak preferences. Huang and Kavitha [22] study *popular matchings*, which are weakly preferred to any other matching by at least half of the agents. It is known that popularity is a weaker notion than fractional stability [18], which is in turn weaker than ex ante stability. Assuming strict preferences on both sides, they prove that the popular matching maximizing any linear objective can be computed efficiently due to elegant half-integral and self-duality properties of such matchings. Popular matchings remain well-defined with weak preferences, but we are not aware of any work investigating this.

Cardinal preferences. The model of cardinal utilities is even more expressive than that of weak preferences. Any algorithms designed for weak preferences (including ours) can be applied to an instance with cardinal utilities as they induce unique weak preferences; however, applying an algorithm designed for strict preferences requires breaking ties and the result can be dependent on tie-breaking. Caragiannis et al. [9] use cardinal utilities to also justify fractional matchings: they show that stable fractional matchings can have arbitrarily larger utilitarian social welfare compared to stable integral matchings, and design approximation algorithms for the NP-hard problem of maximizing welfare subject to stability. Panageas et al. [30] give an algorithm that computes a fractional matching maximizing the Nash social welfare within an error of ε in $O(1/\varepsilon)$ time. But the Nash-optimal solution, while celebrated in fair division for satisfying envy-freeness under additive/linear preferences [10, 38, 13], even under constraints [12, 39], Tröbst and Vazirani [37] show that it provides no approximation to envy-freeness for two-sided matching, and use other means to show the existence of fractional matchings satisfying two sets of axioms: envy-freeness (EF) and Pareto optimality, and justified envy-freeness (JEF) and weak Pareto optimality.

C Algorithms FDA, DFDA, and FDA-Cycle

Algorithm 2: Fractional Deferred Acceptance (FDA) [26]

$\forall i \in N, w_i = 1$
$\forall i \in N, j \in M, x_{ij} = 0$
while $\exists i \in N, w_i > 0$ **do**
    All proposers i with $w_i > 0$ propose their weight simultaneously to their most preferred acceptor j who has not yet rejected any fraction of them.
    All acceptors j whose tentative matching + proposals are greater than their capacity reject some proposers based on the following process:
    Starting with j's highest equivalency class, if j can accept all proposals from that class without exceeding its capacity, then it does so. Otherwise j accepts proposers from this equivalency class as equally as possible, i.e., j increases the amount accepted of each proposer by an equal amount, only stopping the increase for a given proposer for j has accepted all of that proposers weight, or j runs out of capacity. This process is repeated until j reaches capacity.
end
return x

Algorithm 3: Doubly-Fractional Deferred Acceptance (DFDA)

$\forall i \in N, w_i = 1$ `// Free weights`
$\forall i \in N, j \in M, x_{ij} = 0$ `// Current matching`
while $\exists i \in N, w_i > 0$ **do**
    `// Simultaneous fractional proposals`
    All proposers i with $w_i > 0$ propose simultaneously. Let P_i be the set of (equally) most-preferred acceptors of proposer i who have not rejected i yet. Each proposer i evenly splits her proposal across acceptors in P_i, proposing a weight of $w_i/|P_i|$ to each of them.
    `// Simultaneous fractional rejections`
    All acceptors j whose tentative matching + proposals are greater than their capacity reject some proposers based on the following process:
    Starting with j's highest equivalency class, if j can accept all proposals from that class without exceeding its capacity, then it does so. Otherwise j accepts proposers from this equivalency class as equally as possible, i.e., j increases the amount accepted of each proposer by an equal amount, only stopping the increase for a given proposer for j has accepted all of that proposers weight, or j runs out of capacity. This process is repeated until j reaches capacity.
end
return x

Algorithm 4: FDA-Cycle [26]

$\forall i \in N, w_i = 1$
$\forall i \in N, j \in M, x_{ij} = 0$
while $\exists i \in N, w_i > 0$ **do**
  $i_1 \leftarrow$ arbitrary agent with $w_{i^1} > 0$
  if $\exists$ *a cycle* $C = (i_1, j_1, i_2, j_2, i_3, j_3 \ldots, i_\ell, j_\ell, i_1)$ *in the proposal graph* **then**
    $M_s = \{i \sim_{j_s} i_{s+1} : x_{ij_s} > 0\}$
    for $s \in \{2, \ldots, \ell\}$ **do**
      Define equation $y_s = \sum_{i \in M_s} \max\{x_{ij_s} - (x_{i_{s+1}, j_s} - y_{s+1})\}$
    end
    Define equation $y_1 + w_1 = \sum_{i \in M_1} \max\{x_{ij_1} - (x_{i_2, j_1} - y_2)\}$
    Solve the above system for $y_1, y_2, \ldots, y_\ell$, let y_s denote the amount of i_s that gets rejected from j_{s-1}, and update x accordingly.
  else
    i_1 proposes their free weight to their top acceptor. That acceptor rejects any proposers if necessary using the same criteria as they do in FDA.
  end
end
return x

D Failure of FDA-Cycle on Our Counterexample

Proposers	
i_1:	$j_3 \succ j_4 \succ \{j_1, j_2, j_5\}$
i_2:	$j_3 \succ j_5 \succ \{j_1, j_2, j_4\}$
i_3:	$j_3 \succ j_2 \succ j_1 \succ \{j_4, j_5\}$
i_4:	$j_1 \succ j_2 \succ \{j_3, j_4, j_5\}$
i_5:	$j_1 \succ j_3 \succ \{j_2, j_4, j_5\}$

Acceptors	
j_1:	$i_3 \succ i_4 \sim i_5 \succ \{i_1, i_2\}$
j_2:	$i_4 \succ i_3 \succ \{i_1, i_2, i_5\}$
j_3:	$i_5 \succ i_1 \sim i_2 \sim i_3 \succ i_4$
j_4:	$i_1 \succ \{i_2, i_3, i_4, i_5\}$
j_5:	$i_2 \succ \{i_1, i_3, i_4, i_5\}$

Figure 3: Counterexample on which FDA-Cycle fails to terminate. In each ordering, the relation between the agents listed in the set at the end can be arbitrary.

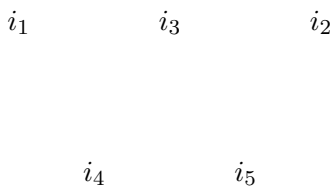
In this section, we will show all the steps of the instance of Figure 3 where FDA-Cycle does not terminate in a finite number of steps. The first several steps of FDA-Cycle on this instance act as expected. We will highlight when we reach the key steps where cycles begin to occur. For each step, we will show both the tentative matching produced, and the current state of the rejection graph that the algorithm uses to resolve cycles.

Consider the instance shown in Figure 3. For the agents listed as a set in the end, we can have arbitrary relationship as long as they are all strictly less preferred than the agents listed previously (e.g., they may form the lowest equivalence class). The infinite looping of FDA-Cycle happens regardless of these relations.

We assume that Algorithm FDA-Cycle breaks any ties lexicographically. That is, when there are multiple proposers in N with free weight, the one with the smallest index among them proposes in the next iteration.

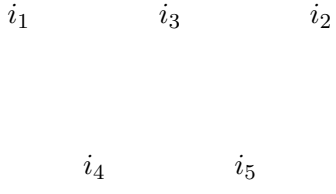
In the first two steps of FDA-Cycle, i_1 and i_2 will both propose to j_3. j_3 will reject half of each of these proposers, and keep the other half.

	i_1	i_2	i_3	i_4	i_5
Free Weight	$1/2$	$1/2$	1	1	1
j_1	0	0	0	0	0
j_2	0	0	0	0	0
j_3	$1/2$	$1/2$	0	0	0
j_4	0	0	0	0	0
j_5	0	0	0	0	0



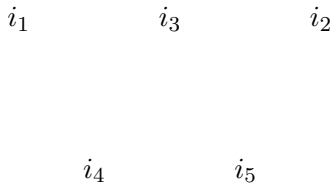
In the next two steps, i_1, now rejected from their top choice j_3, proposes their $1/2$ free weight to j_4. Similarly, j_2 proposes their 1/2 free weight to j_5.

	i_1	i_2	i_3	i_4	i_5
Free Weight	0	0	1	1	1
j_1	0	0	0	0	0
j_2	0	0	0	0	0
j_3	$1/2$	$1/2$	0	0	0
j_4	$1/2$	0	0	0	0
j_5	0	$1/2$	0	0	0



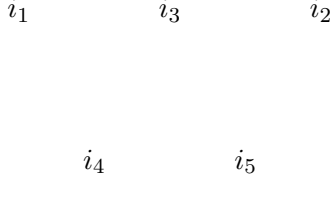
Next, i_3 proposes to j_3. j_3 is indifferent between i_1, i_2, and i_3, so it keeps $1/3$ of each of them, and rejects the rest. In the next two subsequent steps, i_1 and i_2 both take their newly rejected free weight of $1/6$ and propose it to j_4 and j_5 respectively. Note that by [26], this does not cause any edges to appear on the rejection graph, since while there are proposers who are rejected from acceptors while having outstanding weight matched to that acceptor, there are no corresponding proposers who have weight matched to that acceptor and are not yet rejected.

	i_1	i_2	i_3	i_4	i_5
Free Weight	0	0	$2/3$	1	1
j_1	0	0	0	0	0
j_2	0	0	0	0	0
j_3	$1/3$	$1/3$	$1/3$	0	0
j_4	$2/3$	0	0	0	0
j_5	0	$2/3$	0	0	0



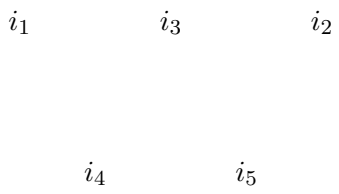
Next, i_3, now rejected by their top choice j_3, proposes their $2/3$ free weight to j_2

	i_1	i_2	i_3	i_4	i_5
Free Weight	0	0	0	1	1
j_1	0	0	0	0	0
j_2	0	0	$2/3$	0	0
j_3	$1/3$	$1/3$	$1/3$	0	0
j_4	$2/3$	0	0	0	0
j_5	0	$2/3$	0	0	0



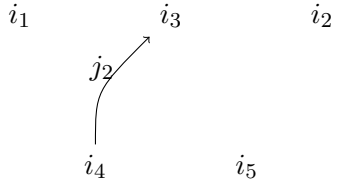
Next, i_4 proposes all their weight to their top choice j_1. Similarly, in the next step, i_5 also proposes all their weight to j_1. j_1 is indifferent between i_4 and i_5, so it keeps $1/2$ of each of them and rejects the rest. Again, this does not cause any rejection edges to appear.

	i_1	i_2	i_3	i_4	i_5
Free Weight	0	0	0	$1/2$	$1/2$
j_1	0	0	0	$1/2$	$1/2$
j_2	0	0	$2/3$	0	0
j_3	$1/3$	$1/3$	$1/3$	0	0
j_4	$2/3$	0	0	0	0
j_5	0	$2/3$	0	0	0



Next, i_4 proposes their $1/2$ free weight to j_2. j_2 prefers i_4 to i_3, so it keeps the $1/2$ weight from i_4 and partially rejects $1/6$ of i_3. This adds an edge between i_4 and i_3 in the rejection graph.

	i_1	i_2	i_3	i_4	i_5
Free Weight	0	0	$1/6$	0	$1/2$
j_1	0	0	0	$1/2$	$1/2$
j_2	0	0	$1/2$	$1/2$	0
j_3	$1/3$	$1/3$	$1/3$	0	0
j_4	$2/3$	0	0	0	0
j_5	0	$2/3$	0	0	0

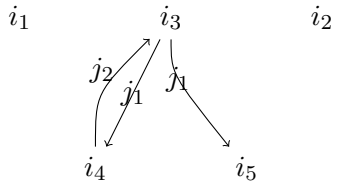


Next, i_3, now rejected from their second choice j_2, proposes their $1/6$ weight to j_1. j_1 prefers i_3 to i_4 and i_5. so it accepts the $1/6$ weight, and rejects $1/12$ from each of the others. This causes edges in the rejection graph between i_3 and i_4, and i_3 and i_5. Notably, this proposal forms a rejection cycle between i_3 and i_4, which the algorithm must solve. Using the FDA-Cycle technique of reduction to linear equation, we are given the following:

$$y_4 + 1/12 = y_3 \tag{1}$$
$$y_3 = 2y_4 \tag{2}$$

Solving this linear system gives us the values $y_4 = 1/12, y_3 = 2/12$, updating the matching with these values give us:

	i_1	i_2	i_3	i_4	i_5
Free Weight	0	0	0	0	$2/3$
j_1	0	0	$1/3$	$1/3$	$1/3$
j_2	0	0	$1/3$	$2/3$	0
j_3	$1/3$	$1/3$	$1/3$	0	0
j_4	$2/3$	0	0	0	0
j_5	0	$2/3$	0	0	0



The key thing to note here is that after solving the linear equation, the same cyclic relationship between i_3 and i_4 still remains. $x_{i_3 j_2} > 0$ and $x_{i_4 j_1} > 0$ still are both true, i_3 has not not yet been rejected from j_1, so it will still propose there the next chance it gets, and the same can be said for i_4 and j_2. Thus, the next time either it is either i_3 or i_4's turn to propose, the same cycle will have to be dealt with again.

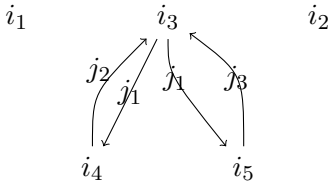
Consider the next step of the FDA-Cycle algorithm in this instance. It is now i_5's turn to propose. i_5 proposes their $2/3$ free weight to j_3. j_3 prefers i_5 to the other 3 agents it is currently accepting fractions of, so it accepts all $2/3$ of i_5, and rejects $2/9$ of i_1, i_2, and i_3. However, this causes edges in the rejection

graph between i_5, and $\{i_1, i_2, i_3\}$. Noticeably, this will cause a rejection cycle between i_5 and i_3, which we can resolve by the following linear system:

$$y_3 + 2/9 = 2y_5 \tag{3}$$
$$y_5 = 3y_3 \tag{4}$$

Solving this linear system gives us $y_3 = 2/45, y_5 = 6/45$. Updating the matching with these values (and running through the non-cyclic proposal steps of i_1 and i_2) gives us:

	i_1	i_2	i_3	i_4	i_5
Free Weight	0	0	0	$6/45$	0
j_1	0	0	$27/45$	$9/45$	$9/45$
j_2	0	0	$1/3$	$2/3$	0
j_3	$3/45$	$3/45$	$3/45$	0	$36/45$
j_4	$42/45$	0	0	0	0
j_5	0	$42/45$	0	0	0



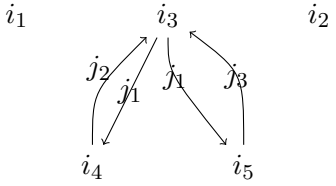
Note that again, the solving of this cycle does not lead to the cyclic relation going away from the tentative matching, the next time i_5 proposes, the cycle will need to be resolved again. We also note that there are technically edges between i_5 and i_1 and i_5 and i_2 in the rejection graph, but they will never become relevant to the algorithm's execution, so we do not include them in our diagram for simplicity.

Next, it is i_4's turn to propose, again, to do this, the cycle between i_3 and i_4 will have to be resolved. This will require solving the following linear system:

$$y_4 + 6/45 = y_3 \tag{5}$$
$$y_3 = 2y_4 \tag{6}$$

This will give us the values of $y_4 = 6/45, y_3 = 12/45$, updating the matching gives us:

	i_1	i_2	i_3	i_4	i_5
Free Weight	0	0	0	0	$6/45$
j_1	0	0	$39/45$	$3/45$	$3/45$
j_2	0	0	$3/45$	$42/45$	0
j_3	$3/45$	$3/45$	$3/45$	0	$36/45$
j_4	$42/45$	0	0	0	0
j_5	0	$42/45$	0	0	0



Now, it is i_5's turn to propose again, and it is still part of the same rejection cycle as previously. One can see that whatever values we get from solving this rejection cycle, will cause i_4 to have free weight in the next step, this will force us to solve i_4's cycle again, which will in turn cause i_5 to have free weight in the next step, with each step along the way, the free weight getting smaller and smaller. It is easy to see how this process continues ad infinium.

To illustrate this, we will solve the i_5 and i_4 cycles with generic values for the free weight.

First the i_5 cycle:

$$y_5 + w_5 = 3y_3 \tag{7}$$
$$y_3 = 2y_5 \tag{8}$$

The solution to this system will be $y_5 = {}^{w_5}/_5, y_3 = {}^{2w_5}/_5$.

Similarly for the i_4 cycle:

$$y_4 + w_4 = y_3 \tag{9}$$
$$y_3 = 2y_4 \tag{10}$$

The solution to this cycle will be $y_4 = w_4, y_3 = 2w_4$.

Therefore, we know that the cycle between i_3 and i_4 will only go away if one of the following happens:

- i_3 gets partially rejected from j_1
- i_4 gets fully rejected from j_1
- i_4 gets partially rejected from j_2
- i_3 gets fully rejected from j_2

Similarly, the cycle between i_3 and i_5 will only go away when:

- i_3 gets partially rejected from j_1
- i_5 gets fully rejected from j_1
- i_5 gets partially rejected from j_3
- i_3 gets fully rejected from j_3

Until one of these events happen, the algorithm will continue to alternate between resolving these two cycles (while also letting i_1 and i_2 propose their unpropsed weight between each step, which will not effect the rest of the process). Clearly, resolving either of these cycles will never cause any of the partial reject conditions to arise, so the algorithm will only exit this cycle resolving loop when one of the following occurs:

- i_3 gets fully rejected from j_2 or from j_3
- i_4 gets fully rejected from j_1
- i_5 gets fully rejected from j_1

Note that due to the way the cycles are resolved, as long as none of these conditions are met, then the amount of i_4 and i_5 matched to j_1 will always be equal. Similarly, the amount of i_1, i_2, and i_3 matched to j_3 will always be equal, so the linear system we have to solve at each cycle removal step will remain the same.

Thus, when we resolve the i_5 cycle next with $w_5 = {}^6/_{45}$, we will reject ${}^6/_{255}$ of i_5 and i_4 from j_1, and ${}^{12}/_{255}$ of i_3, i_1, and i_2 from j_3.

Next, i_4 will have free weight from $6/255$, so resolving its cycle with $w_4 = 6/255$ will mean that $6/255$ of i_4 and i_5 will get rejected from j_1, and $12/255$ of j_3 will get rejected from j_2. We will then have to resolve i_5's cycle with $w_5 = 6/255$.

Putting this together, we have that after k times resolving i_5's cycle from this point, the total amount of i_5 kicked out of j_1 is $\sum_{t=1}^{k} 12/(45*5^t)$, which one can verify approaches $3/45$ in the limit as k approaches infinity, the exact amount that i_5 is matched to j_1 at the beginning of this process.

One can verify that the infinite summations for the other key matrix cells that form the cycles resolve the same way, showing that this sequence will continue forever.

E Alkan-Gale Stability

E.1 Alkan-Gale Matching Model

In [2], the authors defined agent preferences using *choice functions*. They define these choice function in a very broad way such that they generalize a huge range of common matching scenarios, including both integral and fractional matching. For our purposes, we will assume the following simplified definition of a choice function that handles fractional matching scenarios.

Definition 10 (Fractional matching choice function)**.** *Given a set of agents A and a quota q, a choice function $C : \mathbb{R}^A \to \mathbb{R}^A$ is a mapping from one real vector to another (where each entry of this vector corresponds to an amount of some $i \in A$), such that for every $x \in \mathbb{R}^A$, we have that for every $i \in A$, $C(x)_i \leqslant x_i$ (you can only choose at most what is available), and $\sum_{i \in A} C(x)_i \leqslant q$ (your total choice cannot be more than your quota).*

Given our specific definition of choice functions, we also define the join ($\vee$) and meet ($\wedge$) operations as the natural join and meet on the real numbers, i.e., given a set of agents A, and two vectors $x, y \in \mathbb{R}^A$, we say that $x \vee y$ is the vector such that for every $i \in A$, $(x \vee y)_i = \max\{x_i, y_i\}$, and $x \wedge y$ is the vector such that for every $i \in A$, $(x \wedge y)_i = \min\{x_i, y_i\}$.

A given vector $x \in \mathbb{R}^A$ represents all the available ways some agent i can be matched with the agents in A, and when C is i's choice function over A, $C(x)$ represents i's most preferred matching among all these possibilities.

For any two vectors $x, y \in \mathbb{R}^A$, Alkan and Gale [2] gives the following way to determine whether an agent prefers one of these possibilities over the other.

Definition 11 (AG-preference)**.** *For any agent i with choice function C^i over some set of agents A, and for any two vectors $x, y \in \mathbb{R}^A$, we say that $x \succcurlyeq_i^{AG} y$ if $C^i(x \vee y) = x$.*

A full matching problem in the model of [2] gives two sets of agents N and M where each agent $i \in N$ and $j \in M$ has a choice function and a quota. A perfect matching $x \in \mathbb{R}^{[N \times M]}$ in this problem will be such that for all $i \in N$ (resp. $j \in M$) with quota q, we have $\sum_{j \in M} x_{ij} = q$ (resp. $\sum_{i \in N} x_{ij} = q$). In each matching, note that each vector x_i and x_j will be in $\mathbb{R}^M$ and $\mathbb{R}^N$ respectively, so we can use the agents' choice functions to reason about their preferences over their matchings.

Under this model, the notion of a stable matching is defined as follows:

Definition 12 (Saturation)**.** *For some agent $i \in N$, and matching x, i is not j-saturated at x for some $j \in M$ if increasing the amount of j available in x_i would cause i's choice function to choose more of j than x_i has available. i.e., for any $\varepsilon > 0$, define the vector y as $y_{j'} = x_{ij'}$ for all $j' \in A \setminus \{j\}$, and $y_j = x_{ij} + \varepsilon$. If $C^i(y)_j > x_{ij}$ is true, the i is not j-saturated.*

A symmetric definition can be given for when some $j \in M$ is not i-saturated for some $i \in N$.

Definition 13 (AG-stability). *For any matching problem in the Alkan-Gale matching model, a matching x is AG-stable for that problem if for every pair of agents $i \in N$ and $j \in M$, either i is j-saturated or j is i-saturated.*

The goal of [2] is to show a broad class of choice functions such that when all agents in a matching problem have such choice functions, the set of stable matchings for the given problem will form a lattice with respect to each side's AG-preferences, and for a given side, the optimal matching in that lattice can be found through a deferred-acceptance procedure.

To characterize such choice functions, the authors give two properties.

Definition 14 (Consistency). *A choice function C is consistent if for all $x, y \in \mathbb{R}^A$ such that $C(x) \leqslant y \leqslant x$, then $C(y) = C(x)$ is true.*

Definition 15 (Persistence). *A choice function X is persistent if for all $x, y \in \mathbb{R}^A$ such that $x \geqslant y$, then $C(y) \geqslant C(x) \wedge y$.*

With these properties, they are able to state the following:

Theorem 5 (Theorem 2 of [2]). *For any matching problem where all agents have persistent and consistent choice functions, there exists a stable matching x^* that dominates all other stable matchings in terms of the AG-preferences of the agents in N, i.e., for any stable matching y, and all $i \in N$, $x_i^* \succcurlyeq_i^{AG} y_i$. This N-optimal matching can be found by running the procedure of Algorithm 6 with N as the "proposers".*

E.2 Doubly-Strong Ex-Ante Stability Through Choice Functions

We will use a reduction to the model of Alkan and Gale [2] to show that under our preference model, the set of double-strong ex-ante stable matching is equivalent to the lattice of matchings for a given AG matching problems, and the proposer-optimal matching in that lattice can be found through the DFDA procedure.

To do this, we will first define the DFDA choice function in Algorithm 6. For any ordinal matching problem $(N, M, \succcurlyeq_N, \succcurlyeq_M)$, we will assume that each agent has an induced DFDA choice function that is based on their preference ordering.

Algorithm 5: Choice functions in the framework of Alkan and Gale [2] that yield DFDA

INPUT: i's indifference classes E_i; Vector $x \in \mathbb{R}^A$
$c \leftarrow 0^A$
for $E_{ik} \in E_i$ **do**
  if $\sum_{j \in E_{ik}} x_j \leqslant q - \sum_{j \in A} c_j$ **then**
    $\forall j \in E_{ik}, c_j \leftarrow x_j$
  end
  else
    while $\sum_{j \in A} c_j < q$ *and* $c \neq x$ **do**
      Continuously increase c_j for all $j \in E_{ik}$ at the same rate. Only stop increasing c_j for some j if $c_j = x_j$ becomes true.
    end
    return c
  end
end
return c

We will first prove that the DFDA choice function has necessary properties to admit a lattice of stable matching in the Alkan-Gale model.

Lemma 4. *The DFDA choice function is consistent.*

Proof. For contradiction, assume this is false. For some agent i with a preference ordering over a set of agents A, and agent i's induced choice function C^i over those agents with a quota of q, there exists $x, y \in \mathbb{R}^A$ such that $C^i(x) \leqslant y \leqslant x$ but $C^i(y) \neq C^i(x)$.

Let E_{ik} be the lowest equivalency class that $C^i(x)$ chooses agents from before terminating. First consider all the agents from A who are chosen by C^i who are in an equivalency class that is strictly preferred to E_{ik}. From the definition of the DFDA choice function, for each of these agents $i \in A$, C^i would have selected the full amount of j in x. Thus, for each such j, since we have that $C^i(x)_j \leqslant y_j \leqslant x_j$ and $C^i(x)_j = x_j$, it must be the case that $y_j = x_j$. Since C^i starts at the highest equivalency class for i and works its way down, this means that selecting all of each agent strictly preferred to E_{ik} will not exceed C^i's quota q, and thus we must have that $C^i(y)_j = C^i(x)_j = x_j = y_j$ for all $j \in E_{ik'}, k' < k$.

Now consider the agents in E_{ik}. Note that for every $j \in E_{ik}$, we must have that $C^i(x)_j \leqslant y_j \leqslant x_j$. From the definition of the DFDA choice function, we know that the choice function will select agents from this class by continuously increasing the matched amount of all agents in this class at an equal rate, only stopping the increase of an agent if that agent becomes full chosen, quota becomes full, or the vector becomes fully chosen.

By the above arguments, we must have that $C^i(x)_j \neq C^i(y)_j$ for some $j \in E_{ik}$. If $C^i(x)_j > C^i(y)_j$, then consider the exact time in the equivalent increasing process where $C^i(y)$ finishes choosing j. At this point, note that we cannot have that the quota of C^i is full, as by the definition of the DFDA choice function, at the same time in the increasing process for $C^i(x)$, $x \geqslant y$ implies that $C^i(x)$ and $C^i(y)$ will have chosen the same amount of all agents in E_{ik} up to that point, while $C^i(x)_j > C^i(y)_j$ continues increasing after this point. Therefore, it must be the case that $C^i(y)_j$ has been fully chosen by j at this point, and thus $C^i(y)_j = y_j$. However, this fact along with $C^i(x)_j > C^i(y)_j$ would contradict the fact that $C^i(x) \leqslant y$.

If instead we had $C^i(x)_j < C^i(y)_j$, then consider the exact time in the equivalent increasing process where $C^i(x)$ finishes choosing j. At the same point during the process of $C^i(y)$, we know that $C^i(y)$ must have chosen the exact same amount of every agent in E_{ik}. This is due to the fact that $C^i(x) \leqslant y$. Thus, it cannot be the case that $C^i(x)_j$ stops because its quota is full, so it must be the case that it consumed the full amount of x_j. But $C^i(x)_j = x_j$ and $C^i(x)_j < C^i(y)_j \leqslant y_j$ would contradict $y \leqslant x$.

This contradicts the fact that such a j exists, and proves that the DFDA choice function is consistent. □

Lemma 5. *The DFDA choice function is persistent.*

Proof. For contradiction, assume this is false. For some agent i with a preference ordering of a set of agents A, and agent i's induced choice function C^i over those agents with a quota of q, there exists $x, y \in \mathbb{R}^A$ with $x \geqslant y$, and some $j \in A$ such that $C^i(y)_j < \min\left\{C^i(x)_j, y_j\right\}$.

Let E_{ik} be i's lowest equivalency class containing some proposer that is positively chosen by $C^i(x)$. First note that for all $k' < k$, and all $j' \in E_{ik'}$ it must be the case that $C^i(y)_{j'} = y_{j'}$. This follows from the fact that since $x \geqslant y$, we must have $\sum_{k'<k} \sum_{j' \in E_{ik'}} x_{ij'} \geqslant \sum_{k'<k} \sum_{j' \in E_{ik'}} y_{ij'}$, meaning that by the definition of DFDA choice functions, since $C^i(x)$ was able to fully choose every j' strictly preferred to E_{ik}, then $C^i(y)$ will be able to as well.

Therefore, it must be that j is in an equivalency class for i at least as bad E_{ik}, and since $C^i(y)_j < \min\left\{C^i(x)_j, y_j\right\}$ implies that $C^i(x)_j > 0$, then we know that $j \in E_{ik}$ must be true.

Since we must have that $C^i(y)_j < y_j$, it must be the case that E_{ik} is also i's lowest equivalency class containing some proposer that is positively chosen by $C^i(y)$. With this in mind, we can see

that if $C^i(x)_j \geqslant y_j$ were true, then by definition of the DFDA choice function, we should have that $C^i(y)_j = y_j$. This is due to the fact that since $x \geqslant y$, when $C^i(y)$ reaches the class E_{ik}, it will have at least as much free weight to keep choosing as $C^i(x)_j$ did when it reached E_{ik}, and since $y_{j'} \leqslant x_{j'}$ for all $j' \in E_{ik}$, if the process that increases chosen weight in equal amounts for each agent in E_{ik} managed to consume $C^i(x)_j$ of j, then that process should certainly be able to consume at least $y_i \leqslant C^i(x)_j$ of j when choosing from y.

At the same time, if $y_j \geqslant C^i(x)_j$ were true, then we would again reach a contradiction, since by the definition of the DFDA choice function, we would have to have that $C^i(y)_j \geqslant C^i(x)_j$ must be true. To see this, again notice that once $C^i(y)$ reaches equivalency class E_{ik}, it will have at least as much free weight left to choose as $C^i(x)$ did at that same point. Since $C^i(y)_j < C^i(x)_j$, observe the point of the continuous increase in weight where $C^i(y)_j$ first stops increasing. This cannot be because the quota of $C^i(y)$ was reached, or that would contradict the fact that $y \leqslant x$, but it can also not be the case that y_j has been fully chosen, since we have $C^i(y)_j < C^i(x)_j \leqslant y_j$. These contradictions prove that C^i will be persistent. □

Next, we can show that due to the way that we defined DFDA choice function, being AG-Stable with respect to the agents' induced choice functions is exactly equivalent to doubly-strong ex-ante stability.

Theorem 6. *A matching is AG-stable with respect to the agents DFDA choice functions if and only if it is doubly-strong ex-ante stable with respect to the agents' ordinal preferences.*

Proof. First, we will prove the forward direction.

For contradiction, assume this is false, some matching x is AG-Stable with respect to agents' induced choice functions, but not doubly-strong ex-ante stable.

First, we will assume it is not ex-ante stable, thus there exists $i, i' \in N$, $j, j' \in M$ such that $j \succ_i j'$, $i \succ_j i'$, $x_{ij'} > 0$, and $x_{i'j} > 0$. However, it is easy to see that this would violate the definition of AG-Stability with respect to i and j. Since $x_{ij'} > 0$ implies that $C^i(x_i)$ must still have some free space left after choosing j's entire equivalency class. This means that if we increased j by some small amount in x_i, the choice function would choose more, so i is not j-satiated. Similarly, a symmetric argument shows that j is not i-satiated.

Next, we will assume there is ex-ante discrimination on the proposers side, thus there exists $i, i' \in N$, $j, j' \in M$ such that $j \succ_i j'$, $i \sim_j i'$, $x_{ij'} > 0$, and $x_{i'j} > x_{ij}$. Again, in this case we can see that i is not j-satiated in the vector x_i due to the fact that $x_{ij'} > 0$. We can also observe that j will not be i-satiated due to $x_{i'j} > x_{ij}$. If i and i' are not members of the lowest equivalency class matched to j in x_j, then increasing i in x_j will cause $C^j(x_j)$ to select more of it in favor of the lesser preferred proposer it is currently matched to. If i and i' are in the lowest equivalency class for j, then note that since x is a perfect matching, at the point of j's DFDA choice function where the continuous increasing process has chosen x_{ij} of i, it will have not have chosen its entire quota yet, since it still continues on to choose more of $x_{i'j}$. Thus, if x_{ij} were to be increase, this continuous process would choose at least some of that increase, and reach its quota slightly before choosing the full amount of $x_{i'j}$. Note that deriving a contradiction when the matching has ex-ante discrimination on the acceptors side would be a symmetrical argument to above.

Lastly, we will assume that the matching is not ex-ante indifference neutrality, thus there exists $i, i' \in N$, $j, j' \in M$ such that $j \sim_i j'$, $i \sim_j i'$, $x_{ij} < x_{ij'}$, and $x_{ij} < x_{i'j}$. In this case, we can show that i is not j-satiated, and the fact that j is not i-satiated follows from a symmetrical argument. If j and j' are not in i's lowest equivalency class, then increasing j will cause i to accept it in favor of the less preferred acceptor it is currently matched to. If j and j' are in i's lowest equivalency class, then in a identical argument to the previous paragraph, $x_{ij} < x_{ij'}$ implies that increasing x_{ij} will cause i's choice function to select more of j as part of the equivalent increasing process.

This concludes the proof of the forward direction. We will next show the backwards direction. For contradiction, assume that some matching x is doubly-strong ex-ante stable, but is not AG-Stable with respect to the agents' induced choice functions.

Let $i \in N$, $j \in M$, be the pair of agents that violates AG-Stability, i.e., we have that i is not j-satiated, and j is not i-satiated.

Since i is not j-satiated, this means that there exists some vector y_i with $y_{ij'} = x_{ij'} \forall j' \in M \setminus \{j\}$ and $y_{ij} = x_{ij} + \varepsilon$ for some $\varepsilon > 0$, such that $C^i(y_i)_j > C^i(x_i)_j$. Similarly, we since j is not i-satiated, there must exist a symmetrically defined vector y_j such that $C^j(y_j)_i > C^j(x_j)_i$.

Since x is a perfect matching, we must have that $\sum_{j \in M} x_i j = 1$, and thus since the quota of each agents' choice function will be 1, and we have $C^i(y_i)_j > C^i(x_i)_j$, there must be some $j' \in M$ such that $C^i(y_i)_{j'} < C^i(x_i)_{j'}$. Similarly, there must be some $i' \in N$ such that $C^j(y_j)_{i'} < C^j(x_j)_{i'}$. We will now consider each possible case for i's preference ordering over j and j', and j's preference ordering over i and i'.

First, note that it cannot be the case that $j' \succ_i j$. By the definition of the DFDA choice function, if $j' \succ_i j$ is true, then since we have $C^i(y_i)_j > C^i(x_i)_j \geqslant 0$ we must also have that $C^i(y_i)_{j'} = y_{ij'}$. This follows since the only way $C^i(y_i)_{j'} < y_{ij'}$ would be true is if j' is in the lowest equivalency class chosen by i in $C^i(y_i)$. But, $C^i(y_i)_{j'} < C^i(x_i)_{j'} \leqslant x_{ij'} = y_{ij'}$ contradicts this. Meaning that $j \succcurlyeq_i j'$ must be true. By a symmetric argument, we can also say that $i \succcurlyeq_j i'$ must be true.

Case 1: $\mathbf{j \succ_i j', i \succ_j i'}$. In this case, note that $0 \leqslant C^i(y_i)_{j'} < C^i(x_i)_{j'}$ implies that $C^i(x_i)_{j'} > 0$ and thus $x_{ij'} > 0$. Symmetrically, we also have that $x_{i'j} > 0$. This implies that x violates ex-ante stability with respect to i and j, giving a contradiction.

Case 2: $\mathbf{j \succ_i j', i \sim_j i'}$. In this case, we again have that $x_{ij'} > 0$, this means that $x_{ij} \geqslant x_{i'j}$, otherwise this would mean that x violates no ex-ante discrimination for the proposers. Note that this means we have $y_{ji} > x_{ij} \geqslant x_{i'j} = y_{ji'}$. But if this were true, we could not also have that $C^j(y_j)_{i'} < C^j(x_j)_{i'} \leqslant x_{i'j} = y_{ji'}$ and $C^j(y_j)_i > C^j(x_j)_i$. This follows from the definition of the DFDA choice function. Since $C^j(y_j)_{i'} < y_{ji'}$, it must be the case that i', and thus i, are in the lowest equivalency class among accepted proposers in $C^j(y_j)$. Since the only difference between x_j and y_j is that $y_{ji} > x_{ij}$, this means that the choice process will be identical up until the point where x_{ij} of i is chosen. Note at that point, due to $i \sim_j i'$, and the DFDA choice function choosing all agents in the lowest equivalency class at equal proportions, we must have that at that point in the choice process, C^j will have chosen at least $\min\left\{x_{ij}, y_{ji'} = x_{i'j}\right\} = y_{ji'}$ of i', contradicting that fact that $C^j(y_j)_{i'} < y_{i'j}$.

Case 3: $\mathbf{j \sim_i j', i \succ_j i'}$. This follows from a symmetric argument to that of Case 2. Since we have $x_{i'j} > 0$, we must have $x_{ij} \geqslant x_{ij'}$, but this produces a contradiction.

Case 4: $\mathbf{j \sim_i j', i \sim_j i'}$. By the argument presented in Case 2, we know that $x_{ij} \geqslant x_{ij'}$ and $i \sim_j i'$ leads us to a contradiction of the fact that $C^j(y_j)_{i'} < C^j(x_j)_{i'}$. So it must be true that $x_{ij} < x_{ij'}$. However, we also know that using a symmetrical argument to case 3, $x_{ij} \geqslant x_{ij'}$ and $j \sim_i j'$ leads to a contradiction of the fact that $C^i(y_i)_{j'} < C^i(x_i)_{j'}$, so it must also be true that $x_{ij} < x_{i'j}$ is true as well. However, this would mean that x violates ex-ante indifference neutrality, again causing a contradiction.

This shows that a contradiction occurs for every possible ordinal preference ordering of i and j, thus proving the statement. □

In the case of our matching problems, we can relate this proposer optimal matching under AG-preferences back to our traditional notion of preferences through the following lemma:

Lemma 6. *For any two doubly-strong ex-ante stable matchings x, y, if $x_i \succcurlyeq_i^{AG} y_i$ for some i, then $x_i \succcurlyeq_i^{SD} y_i$.*

Proof. For contradiction, assume this is false. For some matchings x, y, we have $x_i \succcurlyeq_i^{\text{AG}} y_i$, and thus

$C^i(x_i \vee y_i) = x_i$, but not $x_i \succcurlyeq_i^{\text{SD}} y_i$.

This means that there is some equivalency class fo i, E_{ik}, such that $\sum_{k'<k} \sum_{j \in E_{ik'}} y_{ij} > \sum_{k'<k} \sum_{j \in E_{ik'}} x_{ij}$. By the fact that x and y are both perfect matchings, this implies that there exists some other equivalency class $E_{ik'}$ with $k' > k$ such that $\sum_{k''<k'} \sum_{j \in E_{ik''}} x_{ij} > \sum_{k''<k'} \sum_{j \in E_{ik''}} x_{ij}$.

From this, we are able to conclude that there exists some j that i places in at least class E_{ik} or higher, such that $y_{ij} > x_{ij}$, and there is some j' that i places in at least class $E_{ik'}$ or lower such that $x_{ij'} > y_{ij'}$. Since $y_{ij} > x_{ij}$, we must also have that $(x_i \vee y_i)_j = y_{ij} > x_{ij}$. Thus, if $C^i(x_i \vee y_i) = x_i$, then we must have that $C^i(x_i \vee y_i)_j < (x_i \vee y_i)_j$. By the way the DFDA choice functions are defined, this would imply that the equivalency class of j is i's lowest equivalency class that has any matchings in $C^i(x_i \vee y_i)$, but this would contradict the fact that $C^i(x_i \vee y_i)_{j'} = x_{ij'} > 0$. □

Algorithm 6: The deferred-acceptance procedure of Alkan and Gale [2] with the choice functions in Algorithm 5.

$B^0 \leftarrow 1^{N \times M}$
$X^0 \leftarrow 1^{N \times M}$
$Y^0 \leftarrow 0^{N \times M}$
$k \leftarrow 0$
while $X^k \neq Y^k$ **do**
  for $i \in N$ **do**
    $X_i^{k+1} \leftarrow C^i(B^k)$
  end
  for $j \in M$ **do**
    $Y_j^{k+1} \leftarrow C^j(X^{k+1})$
  end
  for $i \in N, j \in M$ **do**
    if $Y_{ij}^{k+1} = X_{ij}^{k+1}$ **then**
      $B_{ij}^{k+1} = B_{ij}^k$
    end
    else
      $B_{ij}^{k+1} = Y_{ij}^{k+1}$
    end
  end
  $k \leftarrow k + 1$
end
return X^k

As the final step of this process, in Theorem 1 of [2], the authors provide an algorithm (Algorithm 6) that produces the proposer-optimal matching among AG-preferences. We can show that when agents have DFDA choice functions, this algorithm will be equivalent to the DFDA algorithm.

Theorem 7. *Algorithm 6 is equivalent to Algorithm 3.*

Proof. We can show this equivalence explicitly, by walking through the execution of the DFDA algorithm, while also keeping track of a new matrix b. After each step k of Algorithm 3, we say that for all $i \in N$, $j \in M$, $b_{ij}^k = x_{ij}^k$ if any fraction of i has been rejected by j by that point of the algorithm, and otherwise $b_{ij}^k = 1$. Intuitively, b^k represents the total fraction of each proposer that has not yet been rejected from each acceptor.

For any $i \in N$ and $j \in M$, we can show that i's tentative matching to j at step $k-1$, x_{ij}^{k-1}, plus his tentative proposals to j in round k will equal $C^i(b_i^k)_j$.

To see this, let P_i be the set of agents that i is proposing to in this step. Assume the acceptors in P_i belong to the equivalency class E_{ik}. By definition of DFDA, i must be rejected from all acceptors in the class $E_{ik'}$ for every $k' < k$. This means that for all such acceptors $j' \in \cup_{k'<k} E_{ik'}$, we have that $b_{ij'}^{k-1} = x_{ij'}^{k-1}$. Since x^{k-1} is a valid matching, $C^i(b_i^{k-1})$ must choose the full amount of $x_{ij'}^{k-1}$ for each such j'. After the DFDA choice function has selected these matchings from all preferred equivalency classes, it will select matchings from E_{ik}. Note that it must be the case for every $j' \in E_{ik} \setminus P_i$, i must have been rejected from j', thus will have $b_{ij'}^{k-1} = x_{ij'}^{k-1}$. For all $j' \in P_i$, we will have $b_{ij'}^{k-1} = 1$. Now we can simply observe how the DFDA choice function will choose matchings from this class.

Due to the properties of the DFDA algorithm, at each step of the algorithm, if $j \sim_i j'$, j' has rejected i before step k and j has not yet been rejected, then we must have $x_{ij}^{k-1} \geqslant x_{ij'}^{k-1}$. This means that every $j \in P_i$ is matched to i with at least as much weight as all the acceptors in $E_{ik} \setminus P_i$.

We can note note that $C^i(b_i^{k-1})$ will choose the full available amount of every $j' \in E_{ij} \setminus P_i$. This follows from the fact that x^{k-1} is a valid matching, and x_i^{k-1} must not exceed i's quota of 1. Thus, if $C^i(b_i^{k-1})$ was not able to choose full amount of some acceptor in $j' \in E_{ij} \setminus P_i$ this would contradict that fact or the fact that all acceptors in P_i must have a higher matching that j' in x_i^{k-1}.

Finally, note that in x^{k-1}, it must be true that for all $j, j' \in P_i$, we have that $x_{ij}^{k-1} = x_{ij'}^{k-1}$, this follows from the fact that neither of j and j' have been rejected yet, and thus by the definition of the DFDA, every time i proposed to one of them previously, it proposed to both of them the same amount. Since it must be the case that the matching formed by x_i^{k-1} and i's proposals in step k is perfect matching that exactly meets i's quota, and we know that all agents other that those in P_i from equivalency classes at least as good as E_{ik} will be chosen at exactly their x^{k-1} matching in $C^i(b_i^{k-1})$, it follows that the final part of the $C^i(b_i^{k-1})$ will be for C^i to continue choosing the acceptors from P_i at an equal rate until it has chosen an amount of each of them exactly equal to the amount at which they are matched to i in x^{k-1} plus i's proposals, at which the quota of i will be filled, and it will stop.

One can also note that in each step of the DFDA algorithm, when each acceptor looks at the tentative proposals it received in that step and makes it's rejections, the matching it selects will be identical to the vector selected by the DFDA choice function when run against the vector formed by that acceptor's tentative matching at step $k-1$ plus the proposals it received in step k. Unlike the previous statement, this does not require a nuanced proof, it follows trivially from the definition of the DFDA choice function, and by the described way that the acceptors make their rejections in the DFDA algorithm. It is easy to see that these are the exact same process.

With this in mind, it is easy to see that the Algorithm 6 is performing the exact same steps as the DFDA algorithm. At every step, B^{k-1} is defined equivalently to how we defined b, X^k represents the proposers tentative matchings plus their proposals at this step, and Y^k represents the acceptors rejection choices. The final step of each iteration updates B^k to reflect any rejections that happened this step. From this, we can see that the tentative matching x^k produced after step k of the DFDA algorithm will be equivalent to Y^k in Algorithm 6. □

From Lemma 6, we can conclude that the optimal matching that is guaranteed to exist for AG-preferences is also optimal under our traditional notion of preferences, and through Theorem 7, we can conclude that the process that is known to find this matching is equivalent to DFDA. This completes the proof of Theorem 1, as it shows that DFDA will converge to a proposer-optimal doubly-strong ex-ante stable matching.

F Missing Proofs from Section 5

Lemma 7. *For any acceptor $j \in M$, if at any point during the execution of Algorithm 1 we have $|x_j| = 1$, then $|x_j| = 1$ will remain true for the rest of the algorithm.*

Proof. To see this, it is sufficient to note constraint (5) of the LP, $\forall j \in M, |x_j| = 1 \to z_j = \sum_{i \in N} y_{ij}$. Since $|x_j| = \sum_{i \in N} x_{ij}$, and after each execution of the LP, each x_{ij} will be updated using the formula $x_{ij}^{t+1} \leftarrow x_{ij}^t + y_{ij}^* - z_{ji}^*$, we have that the value of $|x_j|$ gets updated by the formula $\left|x_j^{t+1}\right| \leftarrow \left|x_j^t\right| + \sum_{i \in N} (y_{ij}^* - z_{ji}^*) = \left|x_j^t\right| + \sum_{i \in N} y_{ij}^* - z_j^* = \left|x_j^t\right| = 1$. With the first equality being directly implied by constraints (7) and (8) of the LP. □

Lemma 8. *For any $i \in N$ and $j \in M$, if at any point in Algorithm 1, $i \in R_j \setminus A_j^*$ is true, then it will remain true for the rest of the algorithm.*

Proof. From the logic of Algorithm 1, we can see that the only proposers in R_j who are matched with positive weight to j are those in A_j^*. Thus, it must be the case that i is not matched to j with any positive weight in this step, and thus, we must have that j strictly prefers all the proposers it is currently matched with to i.

Since $i \in R_j$ is true in this step, it follows from the definition of P_i that $j \notin P_i$ must be true. Thus, from condition (4) of the LP, we know that the matching between i and j cannot increase while $i \in R_j$ is true.

Note that this same argument holds for any proposer i' such that $i \succcurlyeq_j i'$. Thus, in the next step of the algorithm, no such agent i' will become positively matched with j. Thus, after the next step, j will still strictly prefer everyone in its matching to all such i', thus $i' \in R_j$ will still be true.

We can continue this argument inductively, and conclude that after every step, j will still prefer everyone in its matching to i and i will never be positively matched to j, thus i will remain in $R_j \setminus A_j^*$ for the rest of the algorithm □

Lemma 9. *For any $i \in N$ and $j \in M$, if at any point in Algorithm 1, $i \in A_j^*$ is true, then i will only leave A_j^* if it is fully rejected from j, and enters $R_j \setminus A_j^*$.*

Proof. A_j^* is defined as the set of proposers who are among the lowest equivalency class matched to j, and among those, the proposers with the highest weight matched to j.

From Lemma 8, we know that this cannot happen because some other proposer from a lower equivalency class becomes positively matched to j. So, it must be the case that some proposer in the same equivalency class to j becomes matched to j with a higher weight than i.

For contradiction, assume that this happens, at some step of the algorithm $i \in A_j^*$ is true, but in the next iteration, after running an instance of the LP and updating the matching, i is no longer in A_j^*, and thus there is some i' with $i' \sim_j i$ who is matched to j at a strictly higher amount than i is.

By condition (7) of the LP, we can see that if $i' \in A_j^*$ were also true, this would lead to a contradiction. Condition (7) ensures that i and i' would be rejected from j the exact same amount during this iteration of the LP, and from the fact that they are in A_j^*, we can conclude that $j \notin P_i$ and $j \notin P_{i'}$ are also true, and thus $y_{ij} = y_{i'j} = 0$ in the LP solution. Therefore, updating the matching with such a solution could not cause i''s matching with j to exceed i.

In the case where $i' \notin A_j^*$, then by the fact that $i' \sim_j i$, we can conclude that $i' \in A_j \setminus A_j^*$ must be true, and we can easily see that this leads to a contradiction by observing condition (10) of the LP. By condition (10) $x_{ij} - z_{ji} \geqslant x_{i'j} + y_{i'j}$ must be true. Again we can conclude that $y_{ij} = 0$ from the fact

that $i \in A_j^*$, and we can also conclude that $z_{ji'} = 0$ from the fact that $i' \notin A_j^*$, following from condition (8) of the LP.

This also shows that in this case, updating the matching with the LP solution will never cause i''s matching with j to exceed i's. □

Lemma 10. *For any $i \in N$ and $j \in M$, at any point of Algorithm 1, if $i \in R_j$ is true, then for all $i' \in N$ such that $i \sim_j i'$, $x_{ij} \geqslant x_{i'j}$ will remain true for the rest of the algorithm.*

Proof. First note that if $i \in R_j \setminus A_j^*$ is true, then it must be matched to j with 0 weight, and j must strictly prefer everyone in its current matching to i. This implies that all i' would have to be matched to j with 0 weight as well, and by Lemma 8, this would continue to hold for the remainder of the algorithm.

Next, from Lemma 9, we know that after the first step where $i \in A_j^*$ is true, it will not leave A_j^* until it is fully rejected by j. Thus, at any point after it gets added to A_j^*, but before it gets it gets fully rejected, we will have that $x_{ij} \geqslant x_{i'j}$ must be true for all i' such that $i \sim_j i'$.

Next, observe that on the step where i gets fully rejected from j, it must be the case that this step, all i' will also be matched to j with weight 0. If this were not true, then it is clear to see that i' would have had to violate condition (7) of the LP (if it were in A_j^*), or condition (10) of the LP (if it were in $A_j \setminus A_j^*$) to be positively matched to j at this step.

From the fact that $i \in A_j^*$ was true in the previous step, we know that all proposers who j strictly prefers i to must in $R_j \setminus A_j^*$, and thus by Lemma 8 will remain there for the rest of the algorithm. By the above analysis, we can also conclude that on the step where i gets fully rejected by j, all proposers in i's equivalency class will be matched to j with weight 0 as well, and thus is must be the case that j strictly prefers everyone in its matching to i at this point. Thus, for all i' such that $i' \sim_j i$, we must have $i' \in R_i \setminus A_j^*$, and by Lemma 8, they will all remain matched with j at 0 forever. □

Lemma 11. *For any $i \in N$ and $j \in M$, at any point of Algorithm 1, if $i \in R_j$ becomes true, then $i \in R_j$ will remain true for the rest of the algorithm.*

Proof. If at this point, $i \in R_j \setminus A_j^*$ is true, then this immediately follows from Lemma 8.

If on the other hand, $i \in A_j^*$ is true at this point, then we know from *Lemma 9*, that it will remain in A_j^* until it is fully rejected, and it is implied by Lemma 10 that once i is fully rejected, it will enter $R_j \setminus A_j^*$, and thus remain there forever. □

Lemma 1. *In any solution to the LP, at least one of the following will be true:*

- *(A)* $\forall i \in C_t, y_i = \sum_{j \in M} z_{ji} + w_i$
- *(B)* $\exists j \in M, |x_j| < 1, \sum_{i \in N} y_{ij} = 1 - x_j$
- *(C)* $\exists i \in N, j \in M, x_{ij} > 0, z_{ji} = x_{ij}$.
- *(D)* $\exists j \in M, \exists i \in A_j^*, \exists i' \in A_j \setminus A_j^*, x_{ij} - z_{ji} = x_{i'j} + y_{i'j}$.

Proof. For contradiction, assume that for some proposal graph G, component C_t, and existing (x, w), the corresponding LP outputs a solution where none of these conditions are true.

First, because condition (A) is false, that means there exists some agent $i^* \in C_t$ such that $y_{i^*} < \sum_{j \in M} z_{ji^*} + w_{i^*}$. Intuitively, this means that after the LP has been resolved and the current matching has been updated with the values of y^*, z^*, i^* will have free weight remaining.

We can show that, since conditions (B), (C), and (D) are also all false, i^* should be able to propose more of their weight without violating the constraints of the LP, leading to a contradiction of the fact that the LP returns a solution maximizing the sum of proposed weights over all the proposers.

Consider what happens if the value of y_{i^*} increases by some very small ε. For each $j \in P_{i^*}$, the value of y_{i^*j} will increase by $\varepsilon/|P_{i^*}|$. For each of these j's, if $|x_j| < 1$, then by the fact that condition (B) is false, we know that $\sum_{i \in N} y_{ij} < 1 - |x_j|$ must be true. As long as epsilon is sufficiently small, it will be the case that $\sum_{i \in N} y_{ij} + \varepsilon/|P_{i^*}| \leqslant 1 - |x_j|$ as well.

Additionally, for all the $j \in P_{i^*}$ such that $|x_j| = 1$, then increasing the proposals to j means that it will have to reject more of the agents in A_j^*. Specifically, since proposals to j are increasing by $\varepsilon/|P_{i^*}|$, j will increase its rejection of each proposer in A_j^* by a factor of $\varepsilon/(|P_{i^*}||A_j^*|)$. Due to condition (C) and (D), such a change will always be possible. We must have that for all $i \in A_j^*$, $z_{ji} < x_{ij}$, and for all $i' \in A_j \setminus A_j^*$, we have that $x_{ij} - z_{ji} > x_{i'j} + y_{i'j}$. Thus, given ε is sufficiently small, the inequalities $z_{ji} + \varepsilon/(|P_{i^*}||A_j^*|) < x_{ij}$ and $x_{ij} - z_{ji} - \varepsilon/(|P_{i^*}||A_j^*|) > x_{i'j} + y_{i'j}$ will still hold.

One can easily verify that the rest of the constraints of the LP will trivially hold after this ε increase as well, as they are all equality constraints that we already implicitly handled above, or have no relation to the variables that we changed.

The above procedure will increase y_{i^*} by a factor of ε, while maintaining all the necessary inequalities of the LP provided that ε is sufficiently small, giving the desired contradiction. □

Lemma 2. *Let y^*, z^* be the variables after resolving some LP in Algorithm 1. The process of updating the current matching using y^*, z^* will change the proposal graph only if at least one of the conditions (B), (C) or (D) are true.*

Proof. For contradiction, assume this is false and that in an LP solution y^*, z^* where conditions (B), (C), and (D) are all false, but the corresponding updating of the matching changes the proposal graph. Consider the different ways that the proposal graph can change.

First, observe the fact that for any $j \in M, i \in N$, an edge from j to i can only change (either appear or disappear) in the proposal graph if A_j^* changed in this iteration. This follows immediately from the fact that by definition, the edge (j, i) exists in the proposal graph if and only if $i \in A_j^*$.

Slightly less trivially, we can observe that for any $i \in N, j \in M$, an edge from i to j can only also change in the proposal graph if $A_{j'}^*$ changed for some $j' \in M$ in this iteration. For any fixed i, the set of j such that (i, j) is an edge in G will be the set of j among i's highest equivalency class such that $j \notin R_i$. Since we know from Lemma 11 that once some acceptor is placed into R_i, it will never be removed for the remainder of the algorithm, it must be the case that if some edge (i, j) is removed from the graph, then j was added to R_i, and similarly, if some edge (i, j) is added to the graph, then some j' must have been added to R_i that lowered the i's highest unrejected equivalency class. Clearly, an acceptor j' can only be added to R_i if they are either directly added to $A_{j'}^*$, or if the lowest equivalency class matched to j changed, which would also cause A_j^* to change.

Thus, it is sufficient to show that if conditions (B), (C), and (D) are all false after some iteration of the LP, then A_j^* will remain the same for all $j \in M$.

In our assumption for contradiction, let j be the acceptor such that A_j^* changes. First consider the case that there is some i that was A_j^* in the previous step, but is not anymore. It cannot be the case that i was fully rejected from j, otherwise that would violate condition condition (C). But from Lemma 9, we know that once some i is in A_j^*, the only way it can leave A_j^* is by being fully rejected. So this cannot be the case.

Therefore, there must some i that was not in A_j^* previously, but is there now. It cannot be the case that i is part of a brand new equivalency class that was not in A_j^* previously, as that could only happen if j just became full for the first time, which would violate condition (B), or the proposers from some

higher equivalency class were fully rejected in the previous step, violating the argument from the last paragraph. Therefore, it must be the case that there are other proposers from the same equivalency class in A_j^*, and i, previously being in $A_j \setminus A_j^*$ has become matched to j at the same weight as them. But clearly this could only happen if a violation of condition (D) occurred. □

Lemma 3. *In some iteration of the main while loop in Algorithm 1, if for every component C_t of proposers, the LP run on C_t terminates with only condition (A) being true, then the matching produced by the last component being solved will be a perfect matching.*

Proof. First note that due Lemma 2, since conditions (B), (C), and (D) are never true during such an iteration of the while loop, the proposal graph will never change, and thus the LP will run on every strongly connected component of proposers.

By definition of condition (A), if an LP for component C_t terminates with (A) being true, then each proposer in C_t will have no free weight after the matching is updated with the LP values. Further, from the fact that the components are solved in a topological ordering, if an agent $i \in C_t$ does not have any free weight after C_t is solved, then there is no component $C_{t'}$ with $t' > t$ whose solution will result in new free weight being pushed back to i.

To formalize this, we can say that for every $C_{t'}$ ordered after C_t, and for all $i' \in C_{t'}$, i is not reachable from i' in the proposal graph. This means that if there is an edge (i', j) for some $j \in M$ in the proposal graph, then there cannot be an edge (j, i) in the graph as well.

It can easily be seen from conditions (5) and (6) of the LP that for any $j \in M$, $z_j > 0$ only if $y_{i'j} > 0$ for some $i' \in N$. Since for an execution of the LP on $C_{t'}$, the only proposers that propose their weight are proposers in $C_{t'}$, we have that for any $j \in M$, $y_{i'j} > 0$ only if $i' \in C_{t'}$ and (i', j) is an edge in the proposal graph. Therefore, for any $C_{t'}$ ordered after C_t, there will never be a $j \in M$ in the LP solving $C_{t'}$ such that $z_{ji} > 0$. Thus, for all $i \in C_t$, we will have $\sum_{j \in M} z_{ji} = 0$, meaning that $w_i = 0$ after updating the matching with the new values from the LP. □

Theorem 3. *Algorithm 1 terminates in polynomial time, and will output a perfect matching.*

Proof. Each iteration of the main for loop simply runs an LP with the number variables and constraints being polynomial in the number of agents, then updates the proposal graph. Clearly a single iteration of this loop terminates in polynomial time,[8] and since it runs once for each strongly connected component in the proposal graph, each instance of this for loop will have at most $|N|$ iterations. Therefore, we just need to show that the algorithm terminates after a polynomial number of iterations of the main while loop.

From Lemma 1, we know when we run the LP on a component C_t, one of four listed conditions—(A), (B), (C), or (D)—must be true.

Condition (A) represents that the LP was able to resolve all the free weight from the proposers in C_t. From Lemma 3, we know that if this happens for every component in a given iteration of the while loop, the the algorithm will terminate with a perfect matching after that iteration.

The other three conditions all correspond to events that cause the proposer graph to change, and thus move the algorithm forward. We will show that each of these conditions can only occur polynomial number of times.

[8] Crucially, note that the optimal solution (y^*, z^*) of each LP affects the updated matching x, which is involved in the right hand side of the subsequent LP. If one uses an arbitrary polynomial-time solver for LPs, this may cause an exponential blow-up in the bit-complexity of the successive LPs and, hence, the time it takes to solve them. This can be prevented by using a polynomial-time algorithm for solving LPs whose running time is independent of the bit-complexity of the right hand side, such as that of Tardos [36].

If condition (B), $\exists j \in M, |x_j| < 1, \sum_{i \in N} y_{ij} = 1 - |x_j|$, is true, that means that there existed some acceptor j that was not full at the beginning of the LP, but is full afterwards. From Lemma 7, we know that this can only happen at most $|M|$ times during the algorithm, since once an acceptor become full, its weight cannot go down again.

If condition (C), $\exists i \in N, j \in M, x_{ij} > 0, z_{ji} = x_{ij}$, is true, this means that some proposer i did have weight matched with some acceptor j at the beginning of the LP, but was fully rejected from j by the results of the LP. By condition (8) of the LP, we can see that z_{ji} can only be positive if $i \in A_j^*$ is true. $i \in A_j^*$ implies that $i \in R_j$ is true, and by Lemma 11, this means that $i \in R_j$ will remain true for every future step of the algorithm. $i \in R_j$ also implies that $j \in P_i$ cannot be true. Finally, following from condition (4) of the LP, in any future iteration of the algorithm, y_{ij} can only be true if $j \in P_i$ is true. This means that once i is fully rejected from j, it can never increase again. Thus, this can only happen once for every i and j, meaning it only happens $|N||M|$ times total.

Finally, if condition (D), $\exists j \in M, \exists i \in A_j^*, \exists i' \in A_j \setminus A_j^*, x_{ij} - z_{ji} = x_{i'j} + y_{i'j}$ is true, this means that there is some acceptor j, and some proposer i that is in A_j (among the lowest equivalency class proposers matched to j), but not in A_j^* (does not have the most weight matched to j among proposers in A_j) who becomes tied for having the most weight matched to j among proposers in A_j, either by x_{ij} increasing and/or $x_{i'j}$ decreasing for all $i' \in A_j^*$. We note that an LP can only terminate with this condition being true once for every $i, i'j$ pair, meaning that it can only happen at most $|N|^2|M|$ times throughout the course of the algorithm.

To see this, first note that condition (D) being true implies that i and i' will be matched to j in the same amount after updating the matching with the LP results. This is because we have $x_{ij} - z_{ji} = x_{i'j} + y_{i'j}$, and we also know from the fact that $i \in A_j^*$ that $y_{ij} = 0$, and from the fact that $i' \notin A_j^*$ that $z_{ji'} = 0$.

Next, note that $i \in A_j^*$ before the LP execution implies that either $i \in A_j^*$ or $i \in R_j \setminus A_j^*$ must be true after updating the matching. This follows from Lemma 11. If $i \in A_j^*$ is true, then by the fact that i and i' are now matched to j with the same amount, then $i' \in A_j^*$ is also true. If $i \in R_j \setminus A_j^*$ is true, then i, and thus, i' must be matched to j at weight 0, and therefore $i' \in R_j \setminus A_j^*$ must also be true. Either way, we have that $i' \in R_j$ is true. By Lemma 11, we know that i' will never leave R_j for the remainder of the algorithm, so therefore, it can never be in $A_j \setminus A_j^*$ again, and thus condition (D) cannot repeat with these agents.

This means that after at most $(|N||M|) + (|N|^2|M|) + |M|$ iterations of the main while loop, either conditions (B), (C), and (D) will have have occurred their maximum number of times, or the algorithm has terminated.

Thus, if the algorithm has not terminated at this point, then in the next iteration of the while loop, condition (A) and none of the other conditions are true after each LP is solved. Following from Lemma 3, the algorithm will terminate after this iteration. □

Theorem 4. *The matching produced by Algorithm 1 is doubly-strong ex-ante stable.*

Proof. First, we will show that Algorithm 1 returns an allocation that is ex-ante stable. For contradiction, assume this is false. This means that there are some $i, i' \in N$ and $j, j' \in M$ such that $j \succ_i j', i \succ_j i'$, $x_{ij'} > 0$, and $x_{i'j} > 0$ are all true.

Note that if $x_{ij'} > 0$, that means that at some point of Algorithm 1, $j' \in P_i$ must have been true. This means that $i \in R_j$ must have been true at that point, or else, $j \in P_i$ would have been the case instead. $i \in R_j$ implies that j can only be currently matched to agents that are weakly preferred to i, and therefore strictly preferred to i'. Thus, we must have that $x_{i'j} = 0$ and $i' \in R_j$ at this point. By Lemma 11, i' will remain in R_j for the rest of the algorithm, and therefore cannot be positively matched to j in the final output, causing a contradiction.

Next we will prove that Algorithm 1 has no ex-ante discrimination for the proposers. For contradiction,

assume this is false. Then there are $i, i' \in N$, $j, j' \in M$ such that $i \sim_j i'$, $j \succ_i j'$, $x_{ij'} > 0$, and $x_{i'j} > x_{ij}$ are all true.

As in the argument for ex-ante stability, $x_{ij'} > 0$ means that at some step of the algorithm, $j' \in P_i$, thus $i \in R_j$. Thus, immediately from Lemma 10, we can conclude that $x_{ij} \geqslant x_{i'j}$, giving a contradiction.

Next, we will prove that the matching produced by Algorithm 1 has no ex-ante discrimination for the acceptors. For contradiction, assume this is false. Then there are $j, j' \in M$, $i, i' \in N$ such that $j \sim_i j'$, $i \succ_j i'$, $x_{i'j} > 0$, and $x_{ij'} > x_{ij}$.

Since, $x_{i'j} > 0$ we know that $i \notin R_j$ was true at every point of the algorithm. Note that by the condition (3) of the LP, whenever i proposes any of its weight to j', since $i \in R_j$ was never true, it must be the case that i will always propose an equal amount of weight to j. Thus, the only way that $x_{ij'} > x_{ij}$ could be true is if j ever rejected some weight from i, which is not possible due to $i \notin R_j$.

Finally, we will prove that x is ex-ante indifference neutral. For contradiction, assume this is false, and there exists agents $i, i' \in N$, $j, j' \in M$, such that $j \sim_i j'$, $i \sim_j i'$, $x_{ij} < x_{ij'}$ and $x_{ij} < x_{i'j}$.

Note that $x_{ij} < x_{ij'}$ implies that at some step in the algorithm, we had $i \in R_j$. This follows from the fact that if $i \in R_j$ was never true, then in an identical argument to the previous paragraph, we know that every time i proposed to j' it must have also proposed the same amount to j. Thus, if $i \in R_j$ were never true, it would have to be the case that $x_{ij} \geqslant x_{ij'}$.

However, $x_{ij} < x_{i'j}$ implies that $i \in R_j$ was never true at any point in the algorithm. This follows from Lemma 10, as $i \in R_j$ would imply $x_{ij} \geqslant x_{i'j}$. This along with the paragraph above clearly cannot be true at the same time, giving us the desired contradiction. □

G Why We Demand Proposers With Free Weights

Suppose there are proposers i, i' and acceptors j, j' with preferences shown on the right, and suppose that at some point of the algorithm, we have $x_{ij'} > 0$ and $x_{i'j} > 0$. Also, suppose that i is rejected from j' and one of the acceptors she will propose to next is j, and i' is rejected by j and one of the acceptors she will propose to next is j'. Finally, suppose i and i' both have no free weight. This would form the following cycle in our proposal graph: $i \to j \to i' \to j' \to i$.

Preferences	
i:	$j' \succ j$
i':	$j \succ j'$
j:	$i \succ i'$
j':	$i' \succ i$

If we did not have the $w_i > 0$ condition, then the LP would maximize flow through this cycle, swapping matched weight on (i, j') and (i', j) for equal weight on (i, j) and (i', j'). However, this leads to the proposers worsening. Also, it does not reflect any actual proposals and rejections that would have happened in DFDA because i and i' had no free weights to kick them off. Adding the condition to Line 7 that some proposer in C_t must have free weight prevents such extra proposals and rejections, thus bringing DFDA-SCC closer to mimicking DFDA.

H Incompatibility With Pareto Optimality

Consider the instance in Figure 4(a), where proposers i and i' are indifferent between acceptors j and j', and j is also indifferent between i and i', but j' strictly prefers i to i'. The only Pareto optimal matching is given in Figure 4(c), which makes j'—the only agent who is not completely indifferent—maximally happy. However, this violates the requirement of ex ante indifference neutrality that $x_{ij} \geqslant \min\{x_{ij'}, x_{i'j}\}$. Note that DFDA and DFDA-SCC produce the matching shown in Figure 4(b) because both proposers initially propose a weight of $1/2$ to both acceptors, which accept them, and the algorithms immediately terminate.

Preferences	
i, i':	$j \sim j'$
j:	$i \sim i'$
j':	$i \succ i'$

(a) Preferences.

	j	j'
i	$1/2$	$1/2$
i'	$1/2$	$1/2$

(b) DFDA matching.

	j	j'
i	0	1
i'	1	0

(c) Ordinally Pareto dominant.

Figure 4: DFDA can be Pareto sub-optimal.

I Extended Discussion

Due to space constraints, the following discussion points are deferred here from Section 6.

One-sided matching with weak agent priorities. Two-sided matching includes one-sided matching, also known as the house allocation problem [23], as a special case, where agents are matched to objects, agents have preferences over the objects, and we can treat every object as being indifferent between all the agents. In this case, DFDA does not seem to coincide with any known algorithm. It cannot ordinally dominate its competitor, probabilistic serial (PS) [7], because PS is ordinally efficient, but we are able to produce instances where PS ordinally dominates DFDA. That said, DFDA yields a natural extension to the case where both sides have weak preferences, whereas for PS, extensions are known only when either agents have weak preferences [24] or objects have weak priorities [19], but not both.

Tradeoffs with other criteria. As mentioned in Section 1, there are various other criteria for fractional two-sided matchings studied in the literature, such as ordinal fairness [19], envy-freeness and justified envy-freeness [37],[9] and popular matching [22]. It is worth exploring the tradeoff between our criteria (particularly, ex ante stability) and these other criteria as well as with utilitarian welfare [9] in two-sided matching.

Best-of-both-worlds guarantees. As mentioned in Section 1, fractional matchings can be implemented as lotteries over integral matchings due to the Birkhoff-von Neumann theorem [5]. This simply finds an arbitrary lottery under which the marginal probability of agents i and j being matched is precisely x_{ij} for all $i \in N$ and $j \in M$. Recently, there is a growing literature on implementing fractional solutions as lotteries while providing "best-of-both-worlds" guarantees: ex ante guarantees on the fractional solution and ex post guarantees on every integral solution in the support [3, 4, 15, 21]. These often use strengthened versions of the Birkhoff-von Neumann theorem such as the bihierarchy extension due to Budish et al. [8]. Whether the fractional matchings returned by DFDA or DFDA-SCC can be implemented while obtaining some ex post guarantees (such as stability of the integral matchings in the support) is an exciting question for the future.

Many-to-many integral matchings. Following the discussion on multi-unit capacities from Section 6, when agents on both sides have multi-unit capacities it is also interesting to investigate *integral* matchings with approximate fairness guarantees: Freeman et al. [16] do so for a relaxation of envy-freeness called EF1, leaving open the question of whether a matching satisfying EF1 for both sides always exists under additive cardinal utilities, but we are not aware of any work doing so for relaxations of stability-inspired criteria.

[9] It should be noted that the justified envy-freeness criterion of Tröbst and Vazirani [37] is different from the no justified envy criterion common in the deferred acceptance literature that coincides with stability for one-to-one matching.

Benjamin Cookson
Department of Computer Science, University of Toronto
Toronto, Canada
Email: bcookson@cs.toronto.edu

Nisarg Shah
Department of Computer Science, University of Toronto
Toronto, Canada
Email: nisarg@cs.toronto.edu