Finding all stable matchings with assignment constraints

Gregory Z. Gutin*  Philip R. Neary†  Anders Yeo†;§

June 16, 2024

Abstract
In this paper we consider stable matchings subject to assignment constraints. These are matchings that require certain assigned pairs to be included, insist that some other assigned pairs are not, and, importantly, are stable. Our main contribution is an algorithm, based on the iterated deletion of unattractive alternatives, that determines if assignment constraints are compatible with stability. Whenever there is a stable matching that satisfies the assignment constraints, our algorithm outputs all of them (each in polynomial time per solution). This provides market designers with (i) a tool to test the feasibility of stable matchings subject to assignment constraints, and (ii) a tool to implement them when feasible.
1 Introduction

In a two-sided market with workers on one side and firms on the other, a stable matching is an assignment of workers and firms from which joint deviations do not pay. In this paper we study many-to-one two-sided matching markets, and our focus is on stable matchings with assignment constraints. These are matchings that require some worker-firm pairs to be included, insist that some other worker-firm pairs are not, and, importantly, are stable.

Consider a labour market with junior doctors on one side and hospitals on the other. Suppose that there is a market designer who is eager for doctor $A$, whose speciality is cardiology, to be matched with one of the four rural hospitals because each, from the designer’s perspective, could use another cardiologist. The designer is also keen for doctor $B$, whose speciality is paediatrics, not to be employed at the down-town hospital since the paediatrics department there is already well-staffed. Suppose further that these two requirements are “all that really matter” to the designer in that how the remaining market participants are matched is immaterial. Naturally, the designer would like to know: does there exist a labour market equilibrium (i.e., a stable matching) that adheres to these criteria?

Any and all variants of the market designer’s question above can be addressed by defining a list of assignment constraints.

**Definition.** Given a many-to-one two-sided market with workers, $W$, on one side and firms, $F$, on the other, a list of assignment constraints specifies a subset of workers $\hat{W} \subseteq W$ and a subset of firms $\hat{F} \subseteq F$ such that:

(i) for each worker $w \in \hat{W}$, there is either a subset of firms, $F^{\text{in}}(w)$, one of which must be $w$’s employer, or there is a subset of firms, $F^{\text{out}}(w)$, at which $w$ must not be employed, or there are both.

(ii) for each firm $f \in \hat{F}$, there is either a subset of workers, $W^{\text{in}}(f)$, such that the positions at $f$ must be filled by workers from the subset, or there is a subset of workers $W^{\text{out}}(f)$, each of whom must not be employed at $f$, or there are both.
The main contribution of this paper is an algorithm that answers the following:

**Constrained Stable Matchings:** Given a many-to-one two-sided market and a list of assignment constraints, is there a stable matching that satisfies the assignment constraints? If yes, generate all such stable matchings.

In a big picture sense, **Constrained Stable Matchings** is about questions of the form: “is there a stable matching in which...?” The designer’s issue is that matchings of the sort that they want might not be stable.\(^1\) The reason is immediate: it is the preferences of individual market participants, and not those of the designer, that determine which assignments are stable. And implementing a matching that is not stable – no matter how desirable to the designer – seems ill advised as it is likely to unravel. Given this, it would be useful for the designer to know in advance if and when constraints are compatible with stability. Our algorithm allows the designer to do precisely this. Furthermore, whenever a stable matching satisfying the constraints exists, our algorithm outputs all of them.

On the surface, **Constrained Stable Matchings** imposes constraints on individual participants. However, upon reflection, constraints of the form that we consider are really about what combinations of pairs are consistent with stability. In fact, we will show that **Constrained Stable Matchings** is equivalent to the following, seemingly far simpler, problem that concerns only pairs.

**Reduced Constrained Stable Matchings:** Given two disjoint collections of worker-firm pairs \(C^{in}\) and \(C^{out}\), is there is a stable matching that includes all worker-firm pairs in \(C^{in}\) and does not include any worker-firm pair in \(C^{out}\)? If yes, generate all such stable matchings.

The rural hospitals theorem confirms that if a market participant is not matched in some stable matching then they are never matched in any, thereby answering the equivalent of **Reduced Constrained Stable Matchings** for individual partic-

---

\(^1\)We are agnostic on why a market designer might rank different assignments differently. Moreover, we do not restrict the designer’s preferences over assignments in any way; as in the example with junior doctors above, we allow for the designer to have strong views on the fortunes of some market participants while being completely disinterested in the circumstances of others.
ipants. While rightly celebrated, the result demonstrates that a market designer who insists on stability is powerless in influencing which collection of market participants “are in” and which collection of market participants “are out”. We believe this underscores the importance for our focus on what collections of pairs can/cannot appear together in some stable assignment, because influencing who the participants that “are in” will be paired with is the only dimension in which a designer can have any sway. For this reason, we hope that our results will be of interest to practitioners of market design and not purely of theoretical interest.

While we view our algorithm’s applicability to questions of interest as our paper’s main contribution, we believe that the algorithm has one marked advantage over existing ones. That advantage is its simplicity. Our methods are easy to explain and even admit an intuitive graphical interpretation (see the figures throughout). This stands in stark contrast to many existing algorithms in the matching literature that use machinery that is not only quite involved but is also difficult to ascribe economic meaning. In fact, arguably all that is required to understand our procedure is familiarity with the definition of an unstable matching. However, it should be noted that we interpret the causes of instability in a non-standard way.

The balance of the paper is as follows. In the remainder of the introduction we provide a quick overview of our algorithm and we discuss how our work fits within the literature. Section 2 introduces many-to-one matching problems, converts these problems to an equivalent formulation using directed graphs (all our results are obtained using this formulation), and restates our question of interest. Section 3 presents the IDUA algorithm that reduces every matching problem to its normal

The observation that the same market participants appear in every stable matching was made first by McVitie and Wilson (1970) for the original one-to-one environment and is sometimes referred as the “lone wolf theorem”. The result was extended to richer many-to-one environments in Roth (1984) and Gale and Sotomayor (1985a,b) where it was also noted that those on the side of the “one” are matched with the same number of individuals in every stable matching. Roth (1984) was a study of the National Resident Matching Program (the NRMP - the system in the United States of America that matches junior doctors to hospitals), and this finding became known as “the rural hospitals theorem” due to the fact that the underserved hospitals (that are always underserved) were typically found in supposedly less desirable (rural) locations. The result was further extended in Roth (1986) where it was shown that any hospital with unfilled positions in some stable matching is, it turns out, assigned the same set of junior doctors in every stable matching.
form. For the reader’s convenience, Section 4 is split into two parts. In Section 4.1 we formally state our algorithm, prove its correctness, and confirm its time complexity. In Section 4.2 we demonstrate how the algorithm operates by laboriously working through the paper’s main example, Example 1, of a market that a fictitious market designer has placed constraints on.

## 1.1 Overview of the algorithm

Here we provide a sketch of how our algorithm operates. We are deliberately loose, focusing primarily on the intuition behind the key steps of the algorithm and how it exploits structural properties of a matching market. As mentioned above, a complete run of the algorithm is described in detail in Section 4.2.

The central tool in our approach is the *iterated deletion of unattractive alternatives* (IDUA), a procedure introduced first in Balinski and Ratier (1997) and used by us in Gutin et al. (2023) to classify unique stable matchings. IDUA reduces a matching problem by repeatedly deleting matchings that cannot be stable by singling out pairs that do not mesh with stability. The issue is how to identify such pairs.

To fix ideas, consider a market that includes the workers $w$ and $w'$ and the firms $f$ and $f'$, and, to keep things simple, let us assume that each firm has only one position. Now, consider a matching $\mu$ that includes the pairs $(w, f')$ and $(w', f)$, and suppose further that both $w$ and $f$ would prefer to be paired together over their current assignment. In this case the pair $(w, f)$ is said to be blocking for the matching $\mu$, and $\mu$ is said to be unstable as a result. A matching is stable if it is not unstable.

The matching $\mu$ above is deemed unstable because the pair $(w, f)$ is, in a sense, “strong” relative to it. The thinking is that since both $w$ and $f$ are better off if they leave their current partners for each other, that they will do exactly that. But we note that the pair $(w, f)$ is not a pair in $\mu$. It is a pair *outside* of $\mu$. We propose an alternative explanation for why $\mu$ is unstable that comes from examining what

---

3Of course, once $w$ and $f$ pair up, both $w'$ and $f'$ are “back on the market” and this can have ramifications since each might be deemed an upgrade by other currently matched participants. Dynamics based on chain reactions of this kind are considered in Roth and Vande Vate (1990) and more recently in Rudov (2024).
is going on inside the matching. In this case, consider the pairs \((w, f')\) and \((w', f)\). Each of these pairs contributes to the breakaway couple. But in much the same way that the resulting blocking pair \((w, f)\) can be viewed as strong relative to \(\mu\), each of these pairs can be regarded as “weak” relative to it. And it is the weaknesses of these pairs that results in the matching being unstable. Here is the key point: there are conditions under which a pair is “always weak” in every matching that contains it. This occurs when one member of the pair has an outside option that not only trumps the current one but is also guaranteed. Our notion of unattractive alternatives formalises this.

Suppose that not only does \(w\) prefer \(f\) to \(f'\), but that \(w\) prefers \(f\) over all other firms. In this case, every matching containing the pair \((w', f)\) is unstable. In fact, any matching in which \(f\) is paired with a worker that it prefers less than \(w\) is unstable. To see why observe that \(f\) could always (at least) propose to \(w\) and such a proposal would certainly be accepted. We refer to pairs like \((w', f)\) as unattractive alternatives, and we note that the submarket with these pairs deleted must have the same set of stable matchings. An important observation is as follows. It is of course possible that some worker in a deleted pair has \(f\) as their top choice. But since being paired with \(f\) is not feasible for this worker, they have a new top choice firm which provides a new guaranteed outside option for that firm. Thus, deleting an unattractive pair, which rules out all matchings containing the pair, can generate unattractive alternatives that were not unattractive before and this can allow for further deletions. In other words: iterate.

Once the IDUA procedure stops – and it must – what remains is the normal form:

---

\(^4\)In Section 2.3 of Gutin et al. (2023), we argue that the IDUA procedure is the matching market analog of the iterated deletion of dominated strategies for strategic games. The parallels seem natural to us. A player’s dominated strategy cannot be part of any solution because it is weak in that the player has an alternative strategy that is guaranteed to pay better no matter what. That is, a dominated strategy is always weak and therefore every strategy profile containing it can be removed from consideration. While a dominated strategy should never be played, its close-but-distant relative, a dominant strategy, must be chosen by a rational player since it is always the best response. That is, dominant strategies are strong relative to all strategy profiles. In a matching market a blocking pair is always strong only if both members of the pair rank each other first. Clearly, in both non-cooperative and cooperative environments, the notion of “always strong” is a more demanding requirement than “always weak”.

---

4
a (sub)matching market that contains all the essential information of the original market in that the set of stable matchings remains unchanged (see Lemma 1). Not only are smaller mathematical objects typically easier to analyse but, as we discuss in Section 3, knowledge of the normal form strengthens the rural hospitals theorem free of charge since, in addition to the well-known insights listed in Footnote 2, the normal form also reveals what worker-firms pairs are not part of any stable matching amongst those workers and firms that always feature.

While running IDUA generates the normal form of a matching market, the same procedure can also be used to explore the set of solutions and when they might accord with assignment constraints. To see how, suppose that some worker $w$ and some single-position firm $f$ view each other as acceptable in that each prefers being paired together over being unmatched. Suppose further that both participants appear in the normal form meaning that $w$ is employed and the position at $f$ is filled in every stable matching. Finally, suppose that the pair $(w, f)$ is part of the normal form and that the market designer would like to know if there is an equilibrium in which $w$ is not employed at $f$.\(^5\) Such a requirement has consequences. Not only can the pair $(w, f)$ not be included, but it must also be the case that $(w, f)$ is not a blocking pair against any candidate outcome.

The above hints at the difficulty. One cannot simply discard all unwanted pairs since one or more of them may be blocking for a matching that satisfies the constraints. However, it turns out that certain unwanted pairs can be ignored. The key insight comes from combining Lemma 2 and Lemma 4.

Lemma 2 notes that both the collection of worker optimal pairs and the collection of firm optimal pairs in the normal form comprise stable matchings. In fact these are precisely the stable matchings found by running deferred acceptance. Lemma 4 then observes that extreme pairs in the normal form can never be blocking in the normal form because, due to the well-known tension in preferences over stable matchings between the opposing sides of the market, if an extreme pair is optimal for one then it is the worst outcome for the other. This has the consequence that a stable matching not including an extreme pair in the normal form is also a stable matching.

\(^5\)If the pair $(w, f)$ is not in the normal form then there is no stable matching containing the pair.
in the normal form with that pair removed (modulo there being a stable matching without that pair). So, if an extreme pair is unwanted, then it can be deleted since the only stable matchings forfeited are those that include the extreme pair. Finally, the removal of an extreme pair creates a new extreme pair which affords IDUA the potential to reduce the market further since some participant has a new top choice.

While deleting unwanted extreme pairs and then running IDUA is permitted, there remains the question of what to do when no extreme pair is unwanted. The answer is to select an extreme pair, any extreme pair will do, and “duplicate” the environment. In the first replica, the algorithm designates this extreme pair as unwanted and proceeds as described above. In the other copy of the environment, the extreme pair is forced to be included. If the extreme pair is given by \((w, f)\), then all other pairs involving \(w\) or containing \(f\) are classified as unwanted and with these new unwanted pairs we proceed as before. This repeated splitting generates a binary search tree that correctly returns all stable matchings satisfying the constraints (Theorem 1).

1.2 Related work

The literature to which our paper is closest falls under the umbrella term of “matchings with constraints” (Kamada and Kojima (2017) is a survey). Constraints can be problematic as they may prove a barrier to stability. What should a designer do when stability cannot be assured? There is a growing literature that addresses this issue and proposes mechanisms that generate assignments deemed adequate in other ways. We do not take a stance on what desiderata an unstable matching should satisfy for it to be deemed up to par. We simply note that just because a stable matching satisfying constraints is not guaranteed to exist is not a guarantee that one does not. Viewed in this way, our algorithm can be used as a first check on whether a particular matching market with constraints requires that stability be relaxed.

To the best of our knowledge, the first paper to consider external constraints in matching markets is the Roth (1984) study of the NRMP referenced in Footnote 2. Amongst other things, Roth considered the consequences of two junior doctors
who are a couple in their private life and, for example, desire to work at the same hospital above all else. Roth showed that satisfying constraints of this type may be incompatible with stability. Ehlers et al. (2014) and Kamada and Kojima (2015) consider school choice problems with diversity constraints, and the Japanese medical residency market that has regional caps, respectively.

In theory, finding all stable matchings subject to any kind of constraints can be answered by brute force. One begins with an algorithm that finds every stable matching (Irving and Leather, 1986; Gusfield, 1987; Dworczak, 2021; Martínez et al., 2004; Bonifacio et al., 2022) and then one compares each against the constraints. However, the number of stable matchings can be exponential (Knuth, 1996), and so, despite the fact that each solution can be found in polynomial time (Gusfield, 1987), checking all of them may not be feasible in practice. Our algorithm spends a polynomial amount of time per solution, and so appears of greatest value when the set of constraints is large in which case there are likely to be few solutions. In Appendix D, we present a market with an exponential number of stable matchings but only four that satisfy the constraints. We know of no other procedure that could identify these four stable matchings in a reasonable time frame.

While the literature on what a designer might or ought do in the presence of constraints is vast, we have not found much work that directly addresses how to find stable matchings satisfying constraints. The following are notable exceptions. Gusfield and Irving (1989) show how to determine the existence of stable matching with forced pairs (pairs that must be included). Dias et al. (2003) extend the query of Gusfield and Irving (1989) to one that finds all stable matchings with both forced pairs and forbidden pairs (pairs that must not be included). The combination of forced pairs and forbidden pairs is similar, albeit less general, to our notion of assignment constraints in many-to-one markets. In addition to considering regret-minimising stable matchings, Mandal and Roy (2021) show how to output the two extreme stable matchings satisfying the constraints of Dias et al. (2003). Finally,

\[6\] Klaus and Klijn (2005) and Klaus et al. (2009) showed that existence is salvaged by restricting the preference domain. Kojima et al. (2013) showed that stable matchings with couples exist provided that there are relatively few couples and preference lists are sufficiently short.
Garg (2020) introduces families of constraints termed lattice-linear predicates. An example of such a constraint is that one worker's regret is less than that of another.\footnote{The arXiv version, Garg (2018), developed a parallel algorithm for generating all stable matchings satisfying lattice-linear predicates. We observe that these constraints are quite different to the kind that we consider.}

In the absence of any constraints our algorithm outputs all stable matchings for the market in question. Given this, it provides another avenue to compute the median stable matching (Teo and Sethuraman, 1998; Fleiner, 2002; Sethuraman et al., 2006; Klaus and Klijn, 2006), the generalised median stable matching (Cheng, 2010; Chen et al., 2016), and quantile stable matchings (Chen et al., 2021).\footnote{Cheng (2010) shows that computing these objects is in general a computationally intractable problem (NP-hard). However, Cheng also describes some families of instances for which the problem can be solved in polynomial time.} This is useful as median stable matchings are viewed as “fair” in that they provide a compromise of sorts between the two sides of the market.\footnote{Echenique et al. (2024) present experimental evidence that the median stable matching is a likely outcome in decentralised two-sided matching markets.}

## 2 Constrained matching problems

In Section 2.1 we define matching problems with assignment constraints. In Section 2.2 we show how queries about assignments constraints in markets can be reduced to a seemingly simpler question concerning specific pairs. In Section 2.3 we show how to reformulate matching problems with assignment constraints using directed graphs, by extending this representation of matching problems, originally due to Maffray (1992), to include assignment constraints.

### 2.1 Matching problems with assignment constraints

The participants in a many-to-one two-sided matching market consist of two disjoint sets of *workers* and *firms*.\footnote{We emphasise that the terms “workers” and “firms” are merely placeholders. The matching framework of Gale and Shapley (1962) has been used to to analyse a wide range of environments. See Roth (2008) for a survey.} Let $W$ denote the set of $m$ workers and let $F$ denote
the set \( n \) of firms. Each firm \( f \in F \) has a quota, \( q(f) \), that represents the maximum number of workers for which \( f \) has positions. Each worker \( w \in W \) can hold at most one position.

Participants rank those on the other side of the market. Specifically, each worker \( w \in W \) has a strict preference ordering over some nonempty list of firms in \( F \), and each firm \( f \in F \) has a strict preference ordering over some nonempty list of workers in \( W \). If firm \( f \) is on the preference list of worker \( w \), we interpret this as \( w \) would prefer to be paired with \( f \) over being unmatched, and we say that \( f \) is acceptable to \( w \). An analogous statement holds for the preference lists of firms. A market participant that is not acceptable is said to be unacceptable. We make the simplifying assumption that \( w \) is acceptable to \( f \) if and only if \( f \) is acceptable to \( w \), and we say that \((w, f)\) is an acceptable pair. Finally, we assume that each firm finds some worker acceptable and that each worker finds at least one firm acceptable.

**Definition 1.** An instance, \( P \), of the many-to-one matching problem is a set of \( m \) workers, \( W \), a set of \( n \) firms, \( F \), and a collection of \( m + n \) preference lists, one for each worker and one for each firm.

A matching, \( \mu \), is a subset of acceptable pairs such that each worker \( w \) appears at most once and each \( f \) appears in at most \( q(f) \) pairs in \( \mu \).

The following is the key definition of Gale and Shapley (1962), modified for the environment of this paper.

**Definition 2.** Given a many-to-one two-sided matching market, \( P \), we say that worker \( w \) and firm \( f \) form a blocking pair with respect to matching \( \mu \) in \( P \), if both prefer each other over their current assignment in \( \mu \). That is, both \( w \) and \( f \) find each other acceptable, \((w, f)\) is not a matched pair in \( \mu \), and (i) \( w \) is not in a matched pair in \( \mu \) and \( f \) is in strictly less than \( q(f) \) matched pairs in \( \mu \), or (ii) \( w \) is not in a matched pair in \( \mu \), \( f \) is in a matched pair with \( w' \) and yet \( f \) prefers \( w \) to \( w' \), or (iii) \( f \) is in strictly less than \( q(f) \) matched pairs in \( \mu \), and \( w \) is in a matched pair with \( f' \) and yet \( w \) prefers \( f \) to \( f' \), or (iv) \( f \) is in \( q(f) \) matched pairs, \( w \) is in a matched pair with \( f' \) and there is a worker \( w' \) in a matched pair with \( f \) and both \( w \) prefers \( f \) to \( f' \) and \( f \) prefers \( w \) to \( w' \).
The solution concept that Gale and Shapley (1962) proposed is termed \textit{stability} and is defined by the absence of a blocking pair.

\textbf{Definition 3.} A matching $\mu$ in $P$ with no blocking pairs is a \textit{stable matching}.

Gale and Shapley (1962) proved that every instance of the matching problem possesses at least one stable matching.

As stated in the introduction, we envisage a hypothetical market designer who is faced with a particular matching market and has preferences over the set of matchings. The market designer’s difficulty is that matchings that they desire may not be stable. This tension can be explored by defining assignment constraints as follows. (While we already provided Definition 4 below in the introduction, we repeat it here so as to make this section self-contained. We do likewise for both \textit{Constrained Stable Matchings} and \textit{Reduced Constrained Stable Matchings}.)

\textbf{Definition 4.} For a given many-to-one matching problem (see Definition 1) a \textit{list of assignment constraints} specifies a subset of workers $\hat{W} \subseteq W$ and a subset of firms $\hat{F} \subseteq F$ such that

(i) for each worker $w \in \hat{W}$, there is either a subset of firms, $F^{\text{in}}(w)$, one of which must be $w$‘s employer, or there is a subset of firms, $F^{\text{out}}(w)$, at which $w$ must not be employed, or there are both.

(ii) for each firm $f \in \hat{F}$, there is either a subset of workers, $W^{\text{in}}(f)$, such that the positions at $f$ must be filled by workers from the subset, or there is a subset of workers $W^{\text{out}}(f)$, each of whom must not be employed at $f$, or there are both.

For a given list of assignment constraints, the market designer would like to know whether there is a stable matching consistent with the list. If yes, the market designer would then like to know all of them. Therefore, the market designer faces the following problem that is the main motivation of our paper.

\textbf{Constrained Stable Matchings:} \textit{Given a many-to-one two-sided market and a list of assignment constraints, is there is a stable matching that satisfies the assignment constraints? If yes, generate all such stable matchings.}
We now provide an example of a many-to-one matching problem with assignment constraints. This will be our working example throughout. We emphasise two features of this example to indicate that our framework is as general as possible. First, the market is unbalanced (there are more workers than positions), and second, preference lists are not complete (some pairs are unacceptable).

**Example 1.** Consider a many-to-one two-sided matching problem with six workers, \(W = \{w_1, w_2, w_3, w_4, w_5, w_6\}\), and four firms, \(F = \{f_1, f_2, f_3, f_4\}\). The firms’ quotas are given by \(q(f_1) = q(f_2) = q(f_3) = 1\) and \(q(f_4) = 2\). The preference lists for each worker and each firm are presented below, with the acceptable participants on the other side of the market listed in order of decreasing preference.

\[
\begin{align*}
  w_1 & : f_1, f_2, f_3, f_4 & f_1 & : w_5, w_4, w_3, w_2, w_1, w_6 \\
  w_2 & : f_2, f_1, f_4, f_3 & f_2 & : w_3, w_5, w_4, w_1, w_2, w_6 \\
  w_3 & : f_3, f_4, f_1, f_2 & f_3 & : w_2, w_1, w_5, w_4, w_3 \\
  w_4 & : f_4, f_3, f_2, f_1 & f_4 & : w_5, w_1, w_2, w_3, w_4, w_6 \\
  w_5 & : f_4, f_1, f_2, f_1 & & \\
  w_6 & : f_2, f_1, f_4 & & 
\end{align*}
\]

Now return to the hypothetical market designer. Suppose that, for whatever reason, the market designer wants to know if there is a stable matching in which either worker \(w_1\) or worker \(w_6\) is employed at firm \(f_2\), worker \(w_4\) is not employed at firm \(f_1\), and worker \(w_6\) is not employed at firm \(f_4\)? In the language of Definition 4, the market designer would like an answer to the following question:

**Question 1.** Given the stable matching problem defined above, is there a stable matching consistent with the list of assignment constraints: \(W^{out}(f_1) = \{w_4\}\), \(W^{in}(f_2) = \{w_1, w_6\}\), and \(W^{out}(f_4) = \{w_6\}\)?

In the next subsection we show how to reduce Constrained Stable Matchings into a far simpler but equivalent formulation.
2.2 Reducing Constrained Stable Matchings

The procedure for reducing Constrained Stable Matchings has two main components.

The first step is to run the Iterated Deletion of Unattractive Alternatives (IDUA) procedure on the equivalent one-to-one environment without constraints.\(^{11}\) The reason for reducing to a one-to-one environment is that the IDUA procedure is defined only for one-to-one environments, and the IDUA procedure is the key tool in our algorithm. Running IDUA on the one-to-one variant reduces the unconstrained environment to its normal form – a matching problem that contains the same set of stable matchings, and yet is no larger than the original problem.

Step two involves checking how the constraints stack up against the normal form instead of the market as a whole. Given that the normal form is a (weakly) smaller object, often the list of constraints can be shrunk since some will be rendered redundant. On occasion, it will be clear that a list of assignment constraints leave Constrained Stable Matchings infeasible.

The normal form of a matching problem is always a balanced market (Lemma 3) because workers and positions that do not appear in any stable matching have been discarded by IDUA. As such, assignment constraints that require that a particular worker be employed at one of a certain number of firms is clearly not satisfiable if that worker is never employed in any stable matching. Similarly, a constraint requiring that the worker not be employed at some subset of firms is rendered redundant if that worker does not appear in the normal form.

The original market of Example 1 was unbalanced with more workers than positions and so something has to give. That is, there must be at least one worker (and possibly some position) that is unmatched in every stable matching. Indeed, we will see that worker \(w_6\) is not employed in any stable matching because \(w_6\) does not appear in the normal form. Since \(w_6\) appears in the constraint \(W^{in}(f_2) = \{w_1, w_6\}\) relevant to firm \(f_2\), the constraint is still valid, albeit we reduce from \(W^{in}(f_2) = \{w_1, w_6\}\) to

\(^{11}\)The word “equivalent” here means that there is a bijection between the sets of stable matchings for the two environments. We quickly sketch how this reduction is carried out towards the end of this section but we note that it is a standard procedure, e.g., Gale and Sotomayor (1985b).
\(W^{in}(f_2) = \{w_1\}\). If instead that the market designer’s Question 1 had stipulated that \(F^{in}(w_6)\) was nonempty, then clearly the assignment constraints cannot be satisfied because \(w_6\) is never employed in any stable matching.

Once the constraints in Definition 4 have been reduced as above, we can reformulate them as constraints on worker-firm pairs as follows.

Constraints given by \(W^{out}(f)\) and \(F^{out}(w)\) are easily handled as they immediately specify pairs that cannot be included in the desired output.

Suppose that worker \(w\) and a position at firm \(f\) both appear in the normal form, meaning that \(w\) is always employed and the position at \(f\) is filled in every stable matching. Then a constraint of the form \(F^{in}(w) = \{f\}\) states that the pair \((w, f)\) must be included in any desired stable matching.

More care must be taken with “in” constraints whenever they contain two or more elements. To see this suppose that \(w\) and a position at firm \(f'\) and a position at firm \(f''\) are all included in the normal form, and consider the constraint \(F^{in}(w) = \{f', f''\}\) that stipulates that \(w\) must be employed either at \(f'\) or \(f''\). By definition \(w\) cannot be simultaneously employed at both firms, and so we cannot simply force the matching to include the pairs \((w, f')\) and \((w, f'')\). However, we can instead ask if there is a stable matching at which \(w\) is not employed at any firm other than \(f'\) or \(f''\). We can do this because \(w\) is employed in every stable matching, and so insisting that \(w\) is not employed at any firm other than \(f'\) or \(f''\) is equivalent to requiring that \(w\) is employed at either \(f'\) or \(f''\). The constraint \(F^{in}(w) = \{f', f''\}\) is therefore equivalent to \(F^{out}(w) = \{f : f \neq f', f''\}\). So whenever an “in” constraint contains two or more elements, we convert the constraint into the corresponding “out” constraint.

We have then shown how to reduce Constrained Stable Matchings to a simpler problem, Reduced Constrained Stable Matchings, defined as follows.

**Reduced Constrained Stable Matchings:** Given two disjoint collections of worker-firm pairs \(C^{in}\) and \(C^{out}\), is there a stable matching that includes all worker-firm pairs in \(C^{in}\) and does not include any worker-firm pair in \(C^{out}\)? If yes, generate all such stable matchings.

We note that the reduction in constraints above requires that each worker and
each firm appears at most once in the collection of pairs $C^m$.

Let us now recall how to reduce a many-to-one environment to a one-to-one environment. First we “split” each firm $f$ with $q(f) \geq 2$ (i.e., those firms with two or more positions) into $q(f)$ identical copies, $f^1, \ldots, f^{q(f)}$, such that each copy has the same preferences over workers as $f$. Next, we amend the preferences of workers such that the $q(f)$ copies of firm $f$ appear consecutively and in the same order in the preference list of every worker who views firm $f$ as acceptable. Doing this yields a new one-to-one matching problem where $W$ is a set of $m$ workers, and $F$ is a set of $\sum_{i=1}^n q(f_i)$ firms, $\{f_1^{(1)}, \ldots, f_1^{(q(f_1))}, \ldots, f_n^{(1)}, \ldots, f_n^{(q(f_n))}\}$, with preferences inherited as described.

We now convert the many-to-one matching problem of Example 1 into its equivalent one-to-one formulation. This will be our working example going forward.

**Reduced Example 1.** Having split firm $f_4$ into two separate positions, $f_4^1$ and $f_4^2$, and assumed that all workers strictly prefer $f_4^1$ to $f_4^2$, the preference lists for each worker and position in the corresponding one-to-one matching problem are as presented below. As before, participants on the other side of the market are listed in order of decreasing preference.

\[
\begin{align*}
\text{w}_1: & \ f_1, f_2, f_3, f_4^1, f_4^2 && \text{f}_1: & \ w_5, w_4, w_3, w_2, w_1, w_6 \\
\text{w}_2: & \ f_2, f_1, f_4^1, f_4^2, f_3 && \text{f}_2: & \ w_3, w_5, w_4, w_1, w_2, w_6 \\
\text{w}_3: & \ f_3, f_4^1, f_4^2, f_1, f_2 && \text{f}_3: & \ w_2, w_1, w_5, w_4, w_3 \\
\text{w}_4: & \ f_4^1, f_4^2, f_3, f_2, f_1 && \text{f}_4^1: & \ w_5, w_1, w_2, w_3, w_4, w_6 \\
\text{w}_5: & \ f_4^1, f_4^2, f_1, f_2, f_1 && \text{f}_4^2: & \ w_5, w_1, w_2, w_3, w_4, w_6 \\
\text{w}_6: & \ f_2, f_1, f_4^1, f_4^2
\end{align*}
\]

The market designer’s Question 1 can now be expressed more simply as:

**Reduced Question 1.** Is there a stable matching that includes the pair $(w_1, f_2)$ or the pair $(w_6, f_2)$ and does not include the pair $(w_4, f_1)$ nor the pair $(w_6, f_4^1)$ nor the pair $(w_6, f_4^2)$?
We have determined that $w_6$ is never employed in any stable matching and that all other workers are always employed and all positions are always filled. Therefore, applying the reductions described above the question can be reformulated as querying whether there is a stable matching in which:

$$C^{in} = \{(w_1, f_2)\} \text{ and } C^{out} = \{(w_4, f_1)\}$$

As mentioned before, our first port of call will be to reduce a matching problem to its normal form using the IDUA procedure. Before formally introducing the IDUA procedure, that we postpone until Section 3, we show how to represent a matching problem with assignment constraints as a directed graph.

### 2.3 Matching problems with assignment constraints as digraphs

Let $P$ be an instance of the one-to-one two-sided matching market where $W$ denotes the set of $m$ workers and $F$ denotes the set of $n$ firms. (We assume that the reduction from many-to-one to one-to-one has occurred, and so it is possible that some firms are copies of each other.) Given $P$, we define the associated matching digraph, $D(P) = (V, A)$, where $V$ is the vertex set and $A$ is the arc set. The vertex set $V$ is defined as $V \subseteq W \times F$, where $v = (w, f) \in V$ if and only if $(w, f)$ is an acceptable pair. The arc set $A$ is defined as follows.

$$A_W := \{(w, f_i)(w, f_j) \mid w \text{ prefers } f_j \text{ to } f_i\}$$

$$A_F := \{(w_i, f)(w_j, f) \mid f \text{ prefers } w_j \text{ to } w_i\}$$

$$A := A_W \cup A_F$$

We now show how to depict the matching problem with constraints of Reduced Example 1 using digraphs. Since there are six workers and five positions, the matching digraph is a $6 \times 5$ grid with rows indexed by workers ($w_1$ through $w_6$) and columns indexed by positions ($f_1$ through $f_5$). Preferences are depicted by arcs, horizontal for

---

12The graph-theoretic terminology and notation that we employ is identical to that in Gutin et al. (2023). For the reader’s convenience this notation is recapped in Appendix A.
workers and vertical for firms, that point from a pair to a preferred one. The matching digraph is depicted in Figure 1, where we note that arcs implied by transitivity have been suppressed for readability.

Figure 1: The matching digraph representation of Reduced Example 1, where arcs implied by transitivity have been suppressed for readability. Assignment constraints are depicted by colour coding vertices.

A matching, $M$, in $D(P)$ is an independent set of vertices. Due to the grid-like structure of $D(P)$, independence forbids that any two vertices in $M$ are in the same row or the same column. A stable matching in $D(P)$ is a matching such that every vertex in the digraph is either in the matching or has an out-neighbour in the matching. Such an object is known as a kernel.\textsuperscript{13}

\textsuperscript{13}While kernels are by now well-studied objects in graph theory, they were first introduced by von Neumann and Morgenstern (1944) in studying solutions to cooperative games.
The assignment constraints stipulated by the market designer’s Reduced Question 1 are depicted by colour coding the vertices in Figure 1. The requirement that the position at firm $f_2$ is filled either with worker $w_1$ or worker $w_6$ is conveyed by colouring the vertex (1, 2) in green. This is because worker $w_6$ is not employed at any stable matching, implying that $w_1$ must be employed at $f_2$. Similarly, the three vertices coloured red, (4, 1), (6, 4) and (6, 5), represent those pairs that are forbidden by the market designer, where we note that both (6, 4) and (6, 5) could be left uncoloured as the entire bottom row (corresponding to $w_6$) is irrelevant since $w_6$ will never be employed in any stable matching. The remaining vertices shaded grey are those over which the market designer has no opinion.

Figure 2 shows the two extremal stable matchings for this example. Some vertices
are coloured yellow, others are coloured blue, and vertex $(5, 4)$ is a hybrid of yellow and blue. The yellow vertices together with the hybrid vertex make up the worker-optimal stable matching; the blue vertices together with the hybrid vertex make up the firm-optimal stable matching. Each of these stable matchings is found by running the deferred acceptance algorithm of Gale and Shapley (1962) with the relevant group in the role of proposer. We highlight that neither of these stable matchings satisfy the assignment constraints in Reduced Question 1 since neither contains the required pair $(w_1, f_2)$. That is, neither contain the green vertex of Figure 1. Finally we note that there are no yellow vertices nor blue vertices in the row corresponding to $w_6$. This means that $w_6$ does not appear in any stable matching.

Related to the above, we note that the reduced one-to-one two-sided market may be an unbalanced one (e.g., in our example there are more workers than positions). In the next section we present the IDUA procedure. One upshot of running IDUA is that it will always convert the unbalanced market into a balanced market such that only those participants that are matched in every stable matching remain. Moreover, for those participants who are matched in every stable matching, the normal form will reveal some pairings that cannot be part of any stable matching. This is a strengthening the rural hospitals theorem.

3 The iterated deletion of unattractive alternatives (IDUA)

The first step of our algorithm is to run the Iterated Deletion of Unattractive Alternatives (IDUA) procedure. Since our primary focus is on how IDUA assists in addressing matching problems with assignment constraints we will be brief in describing how it arrives at the normal form. For how the procedure works in general, the interested reader may wish to consult Gutin et al. (2023). In Section 2 the procedure and its behavioural interpretation are discussed and in Figure 3 in Section
4.2 an entire run of the IDUA procedure is spelled out.\textsuperscript{14}

IDUA starts out by observing that some pairs in a matching problem are not a barrier to stability. To see how, suppose that worker \( w \) is the favourite worker of firm \( f \). Then there cannot be a stable matching in which \( w \) is paired with a firm that \( w \) prefers less than \( f \). The reason is that \((w, f)\) would be a blocking pair against any such matching, since \( w \) would propose a joint deviation and \( f \) would certainly accept. Therefore all firms below \( f \) in \( w \)'s preference list can be removed from consideration by \( w \). Such firms are \textit{unattractive} to \( w \). The firms that \( w \) deems unattractive reciprocate by removing \( w \) from their preference lists.

The following deletes unattractive pairs using the language of digraphs.

\textbf{Definition 5.} Given a matching digraph \( D(P) \), we define \( R(D(P)) \) as the subdiagram of \( D(P) \) with the following vertices (and any arcs incident on them) deleted.

(i) If no arc in \( A_F \) leaves \((w, f) \in V(D(P))\) then delete all vertices \((w, f_i)\) such that \((w, f_i)(w, f) \in A_W\).

(ii) If no arc in \( A_W \) leaves \((w, f) \in V(D(P))\) then delete all vertices \((w_i, f)\) such that \((w_i, f)(w, f) \in A_F\).

The operation described in Definition 5 reduces the size of the matching digraph leaving behind a new smaller one. We note that deleting a vertex does not simply remove a pair, it deletes all matchings that contain that pair so the number of candidate solutions is shrunk. An important observation is that there may exist vertices that are not removed by applying \( R \) to \( D(P) \), but which will be removed when applying \( R \) to \( R(D(P)) \). Formally, repeated application of \( R \) to \( D(P) \) defines the following iterative procedure.

\textbf{Definition 6 (The iterated deletion of unattractive alternatives (IDUA))}. Given an instance of the matching problem, \( P \), and its associated matching digraph, \( D(P) = (V, A_W \cup A_F) \), define \( D^0 = D(P) \), and for each \( k \geq 1 \), define \( D^k = R(D^{k-1}) \). Finally,

\textsuperscript{14}In Gutin \textit{et al.} (2023), IDUA is applied to the original one-to-one environment of Gale and Shapley (1962) and is first defined in terms of preferences rather than digraphs. For more general environments, we believe it is easier to apply IDUA by first converting to the digraph formulation.
define the normal form of $P$, $D^*(P)$, as $D^k$ with minimum $k$ such that $D^k = R(D^k)$, i.e., no further reductions will take place for such $D^k$.

Lemma 2 in Gutin et al. (2023) shows that $D^*(P)$ is uniquely defined. This is important to point out since clearly there may be multiple vertices that satisfy the requirements (e.g., part (i) in Definition 5). The lemma assures that the order in which vertices satisfying the condition are chosen (and hence the order in which other vertices are deleted) will not impact the final outcome.

The following lemma confirms that while the IDUA procedure can discard information, no “important” information is lost in the sense that the set of stable matchings for $D(P)$ can be computed using only the normal form $D^*(P)$.

**Lemma 1** (Balinski and Ratier (1997) and Gutin et al. (2023)). The iterated deletion of unattractive alternatives does not change the set of stable matchings. That is, $D(P)$ and $D^*(P)$ contain exactly the same stable matchings.

Let us now show the effect that running IDUA has on the matching problem of Reduced Example 1. In Figure 3 there are two panels. The left hand panel, panel (a), depicts $D(P)$ and is the same as that in Figure 1. The right hand panel, panel (b), depicts the normal form, $D^*(P)$. In the interest of space we will not display here how the IDUA procedure incrementally reduced $D(P)$ to $D^*(P)$.

We now highlight some observations concerning the structure of $D^*(P)$ in panel (b) of Figure 3. First of all, while it is not possible to display all the stable matchings on one image, by comparing the digraph in panel (b) of Figure 3 with the digraph in Figure 2 one can see that the worker-optimal and firm-optimal stable matchings of $D(P)$ remain in $D^*(P)$. (As indeed they must by Lemma 1.) In fact, closer inspection reveals that the worker-optimal and firm-optimal stable matchings of $D(P)$ are made up of the collections of vertices that have no horizontal arcs out of them in $D^*(P)$ and the vertices that have no vertical arcs out of them in $D^*(P)$, respectively (see Lemma 2). Thus, the extremal stable matchings of $D^*(P)$ are precisely those found by running deferred acceptance twice, once with each side in the role of proposer. Or, to put it another way, the normal form $D^*(P)$ is a smaller matching market than
Figure 3: In (a) we depict $D(P)$ associated with Reduced Example 1. After applying IDUA we obtain the digraph, $D^*(P)$, depicted in (b).

$D(P)$ that is “boxed in” by the two extremal stable matchings since all information from preferences lists “beyond” these two extremal matchings is removed.

The second thing to note is that there is only one vertex in row 5 and column 4 of $D^*(P)$. This means that worker $w_5$ is certain to be employed at firm $f_4$ in every stable matching. This same observation could be made by examining Figure 2 wherein vertex $(5, 4)$ was a hybrid of the colours yellow and blue indicating that it was in both the worker-optimal stable matching and the firm-optimal stable matching, and hence, due to the lattice structure of the set of stable matchings, must be included in every stable matching. We also note that there is no vertex in the row labelled by $w_6$ of $D^*(P)$. From this we can conclude that in every stable matching all positions are filled but worker $w_6$ is never employed.

A balanced two-sided market is one with the same number of workers as positions. A perfect matching in a balanced two-sided market is one in which every
worker is employed and every position is filled. The third and final observation we make regarding panel (b) of Figure 3 concerns balanced markets and perfect stable matchings. While $D^*(P)$ is a balanced market, in this case with five workers and five positions, it is also the case that every stable matching is a perfect matching with all five workers employed and all five positions filled.

The next two results, Lemma 2 and Lemma 3, show that the observations above are not artefacts of this specific example. Rather they are general properties that hold for the normal form of every two-sided matching market. Before the formal statements, some additional notation is needed. Let $d_{W}^{*,+}(x)$ denote the number of arcs out of $x \in V(D^*(P))$ in $A_W$ that still remain in $D^*(P)$ and analogously let $d_{F}^{*,+}(x)$ denote the number of arcs out of $x \in V(D^*(P))$ in $A_F$ that still remain in $D^*(P)$.

**Lemma 2** (Balinski and Ratier (1997) and Gutin et al. (2023)). Let $P$ be an instance of the matching problem. The following two sets of vertices, $M_W$ and $M_F$, are both stable matchings in $P$.

$$M_W = \{ x \in V(D^*(P)) \mid d_{W}^{*,+}(x) = 0 \}$$

$$M_F = \{ x \in V(D^*(P)) \mid d_{F}^{*,+}(x) = 0 \}$$

Lemma 2 states that all remaining participants on the same side of the market have a different most-preferred partner in the normal form, and that such an assignment must be a stable matching. Clearly such an assignment must be a matching because otherwise two participants on the same side, two workers say, have the same most-preferred firm. But if this were the case, then the IDUA procedure is not yet finished. Such a matching must also be a stable matching because every participant on one side of the market is paired with their most preferred feasible partner, implying that no participant on that side of the market is willing to break with their current partner which means there cannot be any blocking pairs. That is, the extremal matchings in $D^*(P)$ are precisely the extremal stable matchings in $D(P)$.

The next result, the proof of which is found in Appendix B, confirms that an analyst interested in stable outcomes in two-sided matching markets can always
confine attention to perfect stable matchings in balanced markets. In fact, the result extends the rural hospitals theorem as it also observes that some worker-position pairs may be discarded even amongst those workers and positions that appear in every stable matching. For example, from panel (b) in Figure 3 it is clear that worker \( w_5 \) will never be paired with firm \( f_2 \) in any stable matching. And this despite both appearing in every stable matching.

**Lemma 3.** In \( D^*(P) \) there remain \( r \) non-empty rows and \( r \) non-empty columns (for some \( r \)) and all stable matchings in \( D^*(P) \) contain \( r \) vertices (that is, they are perfect matchings in \( D^*(P) \)).

We emphasise that Lemma 3 holds no matter how unbalanced the original market is and no matter how incomplete preference lists are. As such, any queries concerning the stable matchings of an unbalanced two-sided matching market can be answered by posing the same (reduced) queries of perfect stable matchings in the normal form. It is not uncommon for researchers to limit attention to balanced markets. But since the normal form is always balanced, this is not as restrictive an assumption as first appears.

Together Lemma 2 and Lemma 3 paint a detailed picture of the structure of the normal form. In the next section we show how these structural properties can guide us in how to determine the feasibility of assignment constraints.

## 4 The algorithm

The algorithm of this section solves **Reduced Constrained Stable Matchings**. That is, given \( D(P) \) and disjoint sets \( V^{in} \) and \( V^{out} \) of vertices of \( D(P) \), return all stable matchings of \( D(P) \), each of which includes all vertices in \( V^{in} \) and does not contain any vertex in \( V^{out} \). We remind the reader that our reduction in Section 2.2 ensures that each worker and each firm appears in at most one pair in \( V^{in} \).

For readability, this section is split in two. In Section 4.1 we state precisely the algorithm, and we prove its correctness and time complexity. In Section 4.2 we give a
high level overview of the algorithm by showing how it addresses Reduced Question 1 concerning the matching problem in Reduced Example 1.

4.1 The algorithm: in detail

We now state the algorithm. Recall that we assume that the many-to-one environment has been reduced to its equivalent one-to-one formulation.

To find all solutions satisfying the constraints we define a recursive algorithm, Algorithm 2, which is called by our main algorithm, Algorithm 1, defined as follows.

Algorithm 1. The algorithm has 4 steps.

Step 1.1. Determine $D^*(P)$ by running IDUA.

Step 1.2. Let $AllSolutions = \emptyset$ and let $r$ be equal to the number of non-empty rows (or columns) in $D^*(P)$.

Step 1.3. Call Algorithm 2 $(D^*(P), V^{in}, V^{out})$

Step 1.4. Return $AllSolutions$.

The following is now the recursive part of the algorithm.

Algorithm 2. $(D, V^{in}, V^{out})$

Note that Algorithm 2 is called using the following command: Algorithm 2 $(D, V^{in}, V^{out})$, where $D, V^{in}$ and $V^{out}$ make up the input to the algorithm.

Step 2.1. If $V^{in} \setminus V(D) \neq \emptyset$, then return. For all vertices, $(w, f) \in V^{in}$, place all vertices, different from $(w, f)$, in the same row or column as $(w, f)$ in $V^{out}$.

Step 2.2. While there is a vertex $(w, f) \in V^{out}$ where $d_W^+((w, f)) = 0$ or $d_F^+((w, f)) = 0$, then modify $D$ by letting it be the digraph obtained by running IDUA on $D - (w, f)$.

Step 2.3. Now $(w, f) \not\in V^{out}$ for all $(w, f) \in V(D)$ where $d_W^+((w, f)) = 0$ or $d_F^+((w, f)) = 0$. Define $M_W$ and $M_F$ as follows.

26
• Let $M_W$ contain all $(w, f) \in V(D)$ where $d_W^+(w, f) = 0$.
• Let $M_F$ contain all $(w, f) \in V(D)$ where $d_F^+(w, f) = 0$.

**Step 2.4.** If $|M_W| < r$ (which is equivalent to $|M_F| < r$ as $|M_W| = |M_F|$) or if $V^{in} \not\subseteq V(D)$ then return.

**Step 2.5.** If $M_W = M_F$, then add $M_W$ to $AllSolutions$ and return.

**Step 2.6.** If $M_W \neq M_F$, then let $(w, f) \in M_W \setminus M_F$. (Such an $(w, f)$ exists since $M_W \neq M_F$ and $|M_W| = |M_F|$. Moreover, if there are multiple vertices in $M_W \setminus M_F$, it does not matter which is chosen.) Now make the following two recursive calls before returning.

**Step 2.6a.** Call Algorithm 2 $(D, V^{in} \cup (w, f), V^{out})$.

**Step 2.6b.** Call Algorithm 2 $(D, V^{in}, V^{out} \cup (w, f))$.

A core idea underlying the algorithm is that if recursive calls are made in Step 2.6, then the stable matching $M_W$ will be found in the recursive call made in Step 2.6a and the stable matching $M_F$ will be found in the recursive call made in Step 2.6b. Therefore the only time we can possibly return from a call to Algorithm 2 without adding a solution to $AllSolutions$ (in Steps 2.1 or 2.4) is if no solutions exist to our original problem in which case there will be only one call to Algorithm 2 (from Algorithm 1). If $M_W = M_F$ in Step 2.5, then $M_W$ is the unique feasible solution in the current call to Algorithm 2.

We will now prove that Algorithm 1 returns all stable matchings containing all pairs in $V^{in}$ and not containing any pairs in $V^{out}$ in polynomial time per solution. We first prove the correctness of the algorithm before proving its time complexity. For both results we need Lemma 4 below that is proved in in Appendix B. Before stating the lemma, we introduce some notation. If $D_i$ is a subdigraph of $D^*(P)$ then let $d_{W,i}^+(x)$ ($d_{F,i}^+(x)$, respectively) denote the number of arcs out of $x \in V(D_i)$ in $A_W$ ($A_F$, respectively) that are also present in $D_i$.  

27
Lemma 4. Let $D_i$ be any digraph obtained by running IDUA on some instance of the one-to-one two-sided matching market. Assume that $D_i$ has $r$ rows and $r$ columns (and therefore, by Lemma 3, all stable matchings are perfect matchings of size $r$). Let $v \in V(D_i)$ be arbitrary, such that $d_{W,i}^+(v) = 0$ or $d_{F,i}^+(v) = 0$ in $D_i$. Then, the following two conditions are equivalent.

(i) $M$ is a stable matching in $D_i$ not containing $v$.

(ii) $M$ is a stable matching in $D_i - v$.

Let us now provide intuition for Lemma 4 and show how it sheds further light on the structure of a matching market. We then explain its importance for the algorithm.

Suppose first that $D_i$ in the statement of the lemma represents the normal form, and recall that the normal form is a balanced market where the size of each stable matching is given by $r$. Now, consider some pair $v = (w, f) \in D_i$ that (i) is forbidden to be part of any stable matching, and also (ii) has out degree zero in one direction but not the other. By Lemma 2, $v$ is contained in the optimal stable matching for one party, say worker $w$. Moreover, due to the tension in preferences over stable matchings that exists between the two sides, $v$ must also be contained in the least desirable stable matchings for $f$ since $w$ is $f$’s least preferred worker in the normal form. The key insight of Lemma 4 is that $v$ cannot be a blocking pair. (In fact, no extreme pair can ever be blocking.) To see why, consider a matching in the normal form in which $f$ is paired with some other worker, say $w' \neq w$. It must be that $f$ prefers $w'$ over $w$, and so $f$ would never leave the pair $(w', f)$ for the pair $(w, f)$.

Given that extreme pair $v = (w, f)$ cannot be blocking against any matching in the normal form, it can be removed without impacting the stable matchings that $v$ is not a part of. That is, while deleting $v$ will certainly remove some matchings, it will only remove those stable matchings that contain $v$ and these are stable matchings that we want to discard. Since $v$ was the optimal feasible pair for $w$, with $v$ removed it must be that $w$ has a new favourite firm. Given this, it is possible that the new market can be reduced further by running IDUA again (and recall that running
IDUA never changes the set of stable matchings). Lastly, note that $D_i$ need not refer only to the normal form, so this procedure can be used ad infinitum.

Armed with Lemma 4 we can now state the main theorem, the proof of which is found in Appendix B. Before stating the theorem, we introduce some notation. Recall that $m$ is the number of workers and $n$ the number of positions in $P$. Let $N = \max\{m, n\}$ and let $M = \min\{m, n\}$. We denote by $s$ the number of stable matchings outputted by the algorithm.

**Theorem 1.** Algorithm 1 correctly solves Reduced Constrained Stable Matchings and does so in time $O(N^3 + N^3s)$. Moreover, the time between two consecutive stable matchings outputted by the algorithm is $O(N^3)$.

We conclude this section by recalling that if a stable matching satisfying the constraints stipulated by Reduced Constrained Stable Matchings exists, then Algorithm 1 will not only confirm but will output all of them. Of course it is possible that a market designer might be content with a single stable matching satisfying the constraints. This is not a problem. It is straightforward to modify our algorithm in order to output either side’s optimal stable matching satisfying the constraints. One simply has to output $M_W$ or $M_F$ in the first call to Algorithm 2 instead of making recursive calls in Step 2.6.

### 4.2 The algorithm: an example

Let us now illustrate how our algorithm works. To do so we use the two-sided market from Reduced Example 1 and the market designer’s Reduced Question 1 pertaining to it.

The input to Algorithm 1 will be the preference lists from Reduced Example 1. After the reduction to the one-to-one environment has been performed this is equivalent to the digraph $D(P)$ in Figure 1. The variant of Reduced Constrained Stable Matchings submitted to Algorithm 2 is Reduced Question 1: decide if there exists a stable matching in $D(P)$ containing the vertex $(w_1, f_2)$ and also avoiding the vertex $(w_4, f_1)$. The output will be the empty set if there is no such stable matching and all such stable matchings in the event that one exists.
Step 1.1 of Algorithm 1 runs IDUA to generate the normal form. The result of this is depicted in Figure 3 where the colour coding of vertices is as follows: vertices are coloured green if they are required to be in the stable matching and coloured red if required not to be. Once the IDUA procedure terminates and the normal form $D^*(P)$ has been arrived at, Algorithm 1 then calls Algorithm 2.

The first step of Algorithm 2, Step 2.1, makes two stipulations. In the first, it checks if any of the green vertices were deleted when applying IDUA. If so, then there is no stable matching satisfying the requirements and the algorithm exits. Note that upon running IDUA for the example, vertex $(1,2)$ in $D(P)$ in the left hand panel of Figure 3 remains in $D^*(P)$ in the right hand panel. We observe that two of the red vertices, vertices $(6,4)$ and $(6,5)$, that were required not to be part of a desired stable matching are no longer present in $D^*(P)$ in the right hand panel of Figure 3. This simply confirms that these vertices are not part of any stable matching, and so a constraint requiring that neither be contained in a particular stable matching is rendered redundant. To say it another way, if red vertices are deleted in going from $D(P)$ to $D^*(P)$, the algorithm proceeds unaffected.

Step 2.1 also converts all queries about required pairs (green vertices) into corresponding queries about avoided pairs (red vertices). Since the green vertex, $(1,2)$, remains after applying IDUA, Lemma 3 confirms that worker $w_1$ is employed in every stable matching and that the single position at $f_2$ is filled in every stable matching. But given this, the query about worker $w_1$ being employed at firm $f_2$ can be converted into a collection of equivalent queries about worker $w_1$ not being employed at any firm other than $f_2$ and firm $f_2$’s position not being filled by any worker other than $w_1$. Clearly any stable matching that avoids all these pairs must contain the pair $(w_1,f_2)$.

Thus the problem of determining if there exists a stable matching in which worker $w_1$ is employed at firm $f_2$ is equivalent to one of determining if there is a stable matching in the normal form that does not include vertices $(1,1)$, $(1,3)$, $(1,5)$ and vertices $(2,2)$, $(3,2)$, $(4,2)$, where the first collection of three vertices require that $w_1$ must not be employed at $f_1$, $f_3$, or $f_5$ and the second collection of three vertices require that the position at $f_2$ is not filled by $w_2$, $w_3$, or $w_4$. Panel (a) in Figure 4
displays the result of converting the query about the required (green) vertex \((1, 2)\) in \(D^*(P)\), shown in panel (b) of Figure 3, to an analogous query about (red) vertices being avoided in \(D^*(P)\).

![Diagram](image)

(a) \(D^*(P)\) after converting queries about \(V^\text{in}\) (green) into queries about \(V^\text{out}\) (red).

(b) Result of applying Step 2.2. Note that no red vertices remain.

Figure 4: In (a) we depict \(D^*(P)\) having converted the problem into one of avoiding vertices. In (b) we depict the submatching problem that is arrived at after completing Step 2.2 in the first call to Algorithm 2.

Step 2.2 is one of the key steps. It states that whenever there is a red vertex with either horizontal out-degree zero or vertical out-degree zero, that this vertex is deleted from the digraph and the IDUA procedure is run on the subdigraph that remains. Recall that deleting red vertices with either horizontal out-degree zero or vertical out-degree zero is justified by Lemma 4.

In our example, the vertices satisfying this requirement can be seen in panel (a) of Figure 4. They are \((1, 1)\), \((2, 2)\), \((1, 5)\), \((3, 2)\), and \((4, 1)\), where the first two vertices have horizontal out-degree equal to zero and the latter three vertices have vertical out-degree equal to zero. While any of the vertices satisfying this constraint can be selected at point Step 2.2 – the order is immaterial – we note that not all red vertices need satisfy the constraint. For our example, from panel (a) of Figure 4 it is
clear that neither vertex (1, 3) nor vertex (4, 2) satisfy the requirement as both have positive horizontal out-degree and positive vertical out-degree. We emphasise this point strongly as it is not possible simply to run IDUA on $D^* - V_{out}$.

Panel (b) of Figure 4 depicts the submatching problem that is arrived at after completing Step 2.2 in the first call to Algorithm 2. By definition there will be no red vertices with out-degree zero remaining. (Though we recall that red vertices with positive out-degree could in theory remain, albeit none did in this example. That is, the vertices (1, 3) and (4, 2) were removed when the IDUA procedure was applied to some reduced subdigraph; they were not directly called by Step 2.2.) Moreover, this step can also delete other vertices. For example, in going from panel (a) to panel (b) in Figure 4, we note that red vertices (1, 3) and (4, 2) and the non-red vertex (2, 3) were all deleted by Step 2.2. In the proof of correctness of Algorithm 2 in Theorem 1 we show that the order in which red vertices satisfying the requirement of Step 2.2 are deleted and then apply IDUA does not impact the final outcome.

Step 2.3 now collects the worker-optimal pairs for every worker, $M_W$, and the firm-optimal pairs for every firm, $M_F$. Step 2.4 checks to see that both $M_W$ and $M_F$ are the size of a stable matching. If not, then there cannot be a stable matching satisfying the constraints and the Algorithm states as much. If both $M_W$ and $M_F$ are the size of a stable matching, then Step 2.5 checks to see if they are one and the same. If they are, that is if $M_W = M_F$, then there is exactly one stable matching satisfying the constraints and the algorithm returns it.

If $M_W \neq M_F$, then we can make two observations. First, there are multiple (at least two) stable matchings satisfying the constraints. Moreover, given our result on uniqueness in Gutin et al. (2023), there must be some cycles in the remaining digraph because cycles that cannot be broken by the IDUA procedure are the barrier to uniqueness. We observe that there are two cycles of length 4 in panel (b) of Figure 4 and Figure 5. Second, it must be that $M_W$ ($M_F$) is the worker-optimal (firm-optimal) stable matching satisfying the constraints. For the market of Reduced Example 1 we depict $M_W$ and $M_F$ in Figure 5 below. We adopt the same colour coding as that in

---

In Appendix C we present an example that shows that deleting all vertices in $V_{out}$ and then applying IDUA may yield an incorrect conclusion.
Figure 2. That is, vertices solely in the worker-optimal stable matching satisfying the
constraints are coloured yellow, vertices solely in the firm-optimal stable matching
satisfying the constraints are coloured blue, and vertices in both are coloured a hybrid
of yellow and blue.

Figure 5: The two extremal stable matchings satisfying the constraints.

Since $M_W \neq M_F$, as seen in Figure 5, then Step 2.6 of the algorithm is called.
The first thing this step does is to identify a vertex, $(w, f)$, that is contained in $M_W$
but not in $M_F$. (Given that $M_W \neq M_F$ there must be at least one such vertex, and
if there is more than one such vertex which is selected is immaterial.) We now run
the recursive part of Algorithm 2 on two subcases: one where $(w, f)$ is required to be
included (so that $(w, f)$ is added to $V^{in}$) and the other where it is required not to
be included (and hence $(w, f)$ is contained in $V^{out}$). As can be seen from Figure 5,
running left to right and down the page the first vertex we encounter of this form
is the vertex $(2, 1)$. Thus we now run Algorithm 2 on $(D, V^{in} \cup (2, 1), V^{out})$ and on
$(D, V^{in}, V^{out} \cup (2, 1))$. 

33
This repeated splitting into two subcases generates a binary search-tree that we depict in Figure 6. There are five pairs of panels in Figure 6 labelled (a) through (e). In every pair of panels, the left panel depicts the temporary constraints while the right hand panel depicts the extremal matchings associated with the temporary constraints. A path through the tree ends in a pair of panels that provide a stable matching satisfying the constraints. Paths through the tree can be of different length (as indeed they are in Figure 6).

The first pair of panels in Figure 6, labelled (a), depicts the first time that Step 2.6 of Algorithm 2 is called. The left-hand panel reproduces panel (b) from Figure 4 while the right hand panel reproduces Figure 5. While there is nothing new depicted in these two panels, we include them to emphasise that this is the starting point of the recursive part of the algorithm. From this point, the procedure “splits” indicated by the two thick black arrows from the top row to the second row. One arrow goes “left” to the panels labelled (b) and the other goes “right” to the panels labelled (c).

Let us first consider the two adjoining panels in (b). Note that in addition to vertex (1, 2) being required, it is now also the case that vertex (2, 1) is required. This is indicated by colouring vertex (2, 1) green in the left panel. Given that vertex (2, 1) is required, it must be that vertices (2, 5) and (3, 1) cannot be simultaneously part of the stable matching (this is simply applying Step 2.1 to this subcase). As such (2, 5) and (3, 1) are coloured red in the left hand panel. Contrast this with the left panel of (c). Here we note that vertex is (2, 1) is coloured red, indicating that it is forbidden. In other words, (b) begins the part of the search tree where (1, 2) is required to be included and (2, 1) is required to be included; while (c) begins the part of the search tree where (1, 2) is required to be included and (2, 1) is required not to be included. Clearly there can be no overlap in the set of stable matchings found by each branch.

In the right hand panel of (c), we see the consequences of requiring that vertex (2, 1) not be included. Given this constraint, it must be that (2, 5) and (3, 1) are to be part of any stable matching. Skipping some details, this immediately yields the stable matching \{((1, 2), (2, 5), (3, 1), (4, 3), (5, 4))\}.

The same is not true of the right hand panel of (b). Note that here there remains an anti-clockwise cycle involving vertices \{(3, 3), (4, 3), (4, 5), (3, 5)\}. Again
Figure 6: The binary search-tree.
by appealing to Theorem 1 of Gutin et al. (2023), there must be at least two stable matchings satisfying the current constraints in this branch of the subtree. Indeed this is true and is indicated by the fact that in the right hand panel of (b), $M_W \neq M_F$. Therefore, one more split is required. The vertex that is pivoted around this time is $(3,3)$. In the left hand panel of (d), this vertex is coloured green indicating it must be included, while in the left hand panel of (e) this vertex is coloured red indicating it must not be included.

We conclude by noting that the algorithm identifies three stable matchings for the market of Reduced Example 1 that accord with the hypothetical market designer’s assignment constraints as stipulated by Reduced Question 1. In order of decreasing worker preference, these three stable matchings are:

\[
M^*_1 = \{(1,2), (2,1), (3,3), (4,5), (5,4)\}
\]
\[
M^*_2 = \{(1,2), (2,1), (3,5), (4,3), (5,4)\}
\]
\[
M^*_3 = \{(1,2), (2,5), (3,1), (4,3), (5,4)\}
\]

As such, the market designer’s Question 1 concerning the many-to-one matching market of Example 1 is answered in the affirmative with the three stable matchings satisfying the assignment constraints given by:

\[
\mu^*_1 = \{(w_1, f_2), (w_2, f_1), (w_3, f_3), (w_4, f_4), (w_5, f_4)\}
\]
\[
\mu^*_2 = \{(w_1, f_2), (w_2, f_1), (w_3, f_4), (w_4, f_3), (w_5, f_4)\}
\]
\[
\mu^*_3 = \{(w_1, f_2), (w_2, f_4), (w_3, f_1), (w_4, f_3), (w_5, f_4)\}
\]
APPENDIX

A Digraph terminology and notation

A directed graph (or just digraph) $D$ consists of a non-empty finite set $V(D)$ of elements called vertices and a finite set $A(D)$ of ordered pairs of distinct vertices called arcs. We shall call $V(D)$ the vertex set and $A(D)$ the arc set of $D$ and write $D = (V(D), A(D))$. For an arc $xy$ the first vertex $x$ is its tail and the second vertex $y$ is its head. Moreover, $x$ is called an in-neighbour of $y$ and $y$ an out-neighbour of $x$. We also say that the arc $xy$ leaves $x$ and enters $y$. We say that a vertex $x$ is incident to an arc $a$ if $x$ is the head or tail of $a$. For a vertex $v \in V(D)$, the out-degree of $v$ in $D$, $d^+_D(v)$, is the number of out-neighbours of $v$. Similarly, the in-degree of $v$ in $D$, $d^-_D(v)$, is the number of in-neighbours of $v$. A vertex $u$ is isolated if $d^+_D(u) = d^-_D(u) = 0$.

A walk, $W$, in a digraph $D$ is a sequence of vertices $x_1, x_2, \ldots, x_p$ for which there is an arc from each vertex in the sequence to its successor in the sequence. Such a walk is written as $W = x_1x_2\ldots x_p$. Special cases of walks are paths and cycles. A walk $W$ is a path if the vertices of $W$ are distinct. If the vertices $x_1, x_2, \ldots, x_{p-1}$ are distinct, for $p \geq 2$ and $x_1 = x_k$, then $W$ is a cycle.

For a digraph $D = (V, A)$ and an arc $xy \in A$, deletion of $xy$ from $D$ results in the digraph $D - xy = (V, A \setminus \{xy\})$. For a vertex $v \in V$, deletion of $v$ from $D$ results in the digraph $D - v = (V \setminus \{v\}, A \setminus A_v)$, where $A_v$ is the set of arcs in $A$ incident to $v$. A digraph $D'$ is called a subdigraph of $D$ if $D'$ is obtained from $D$ by deleting some vertices and arcs. If only vertices are deleted, $D'$ is an induced subdigraph of $D$.

B Proofs omitted from the main text

Lemma 3. In $D^*(P)$ there remain $r$ non-empty rows and $r$ non-empty columns (for some $r$) and all stable matchings in $D^*(P)$ contain $r$ vertices (that is, they are perfect matchings in $D^*(P)$).

Proof. Let $r_W$ denote the number of non-empty rows in $D^*(P)$ and let $r_F$ denote the number of non-empty columns in $D^*(P)$. By definition, every row contains exactly
one sink of that row (i.e., a vertex \( v \) such that \( d_{W,i}^+(v) = 0 \)). Moreover, by Lemma 2 this vertex \( v \) must be a source of a column (i.e., \( d_{F,i}^-(v) = 0 \)) and each column contains at most one source. Putting this together, we have that \( r_F \geq r_W \). A mirror image argument shows that \( r_W \geq r_F \), and so we have that \( r_W = r_F = r \).

To show that all stable matchings in \( D^*(P) \) are perfect matchings in \( D^*(P) \) of size \( r \), assume to the contrary. Then there must be a stable matching, \( M \), of size strictly less than \( r \) (clearly \( M \) cannot contain more than \( r \) pairs). Now, let \( w \) be an unmatched worker in \( M \) and let \( f \) denote the firm that has \( w \) at the top of its preference list in \( D^*(P) \) (i.e., the vertex \((w, f)\) is the sink of the column corresponding to firm \( f \)). Either (i) \( f \) is not matched in \( M \), or (ii) \( f \) is paired with worker \( w' \neq w \) in \( M \). No matter which holds, the matching \( M \) is not stable as the pair \((w, f)\) is a strict improvement for both \( w \) and \( f \).

**Lemma 4.** Let \( D_i \) be any digraph obtained by running IDUA on some instance of the one-to-one two-sided matching market. Assume that \( D_i \) has \( r \) rows and \( r \) columns (and therefore, by Lemma 3, all stable matchings are perfect matchings of size \( r \)). Let \( v \in \mathcal{V}(D_i) \) be arbitrary, such that \( d_{W,i}^+(v) = 0 \) in \( D_i \). Then, the following two conditions are equivalent.

(a) \( M \) is a stable matching in \( D_i \) not containing \( v \).

(b) \( M \) is a stable matching in \( D_i - v \).

Proof. Without loss of generality assume that \( d_{W,i}^+(v) = 0 \).

We first show that (b) implies (a). Assume that \( M \) is a stable matching in \( D_i - v \) of size \( r \). Clearly \( v \notin M \) since \( v \notin D_i - v \), and so it suffices to show that \( M \) is a stable matching in \( D_i \). As \( d_{W,i}^+(v) = 0 \) in \( D_i \), we note that \( v \) can have no arcs into it in \( D_i \) from \( A_F \) (as \( D_i \) is a digraph obtained by running IDUA on some instance). But then, if \( v = (w, f) \) and \( u = (w', f) \in M \) (\( u \) must exist as the size of \( M \) is \( r \)), we must have that \( vu \in A(D_i) \). This implies that \( v = (w, f) \) is not a blocking pair for \( M \) in \( D_i \). Finally, because no vertex in \( D_i - v \) is a blocking pair for \( M \) in \( D_i \) (as they are not blocking for \( M \) in \( D_i - v \)), we have that \( M \) must be a stable matching in \( D_i \).
Now assume that (a) holds so that $M$ is a stable matching in $D_i$ not containing $v$. Clearly $M$ is also a stable matching in $D_i - v$ because no vertex in $D_i$ is blocking for $M$, and so (b) holds.

Before stating the theorem, recall that $m$ is the number of workers and $n$ is the number of firms in $P$. Also, $N = \max\{m, n\}$, $M = \min\{m, n\}$ and $s$ is the number of stable matchings outputted by the algorithm.

**Theorem 1.** Algorithm 1 correctly solves Reduced Constrained Stable Matchings and and does so in time $O(N^3 + N^3s)$. Moreover, the time between two consecutive stable matching outputted by the algorithm is $O(N^3)$.

By Lemma 1 the digraph $D^\ast(P)$ used in the call to Algorithm 2 in Step 1.3 will contain exactly the same stable matchings as $D(P)$. So, if Algorithm 2 adds all the desired stable matchings to $AllSolutions$ in the desired time, then Theorem 1 holds. We will now prove this. The proof is partitioned into two parts, the proof of correctness and the proof of the running time claim. In what follows, stable matchings in $D(P)$ are referred to as solutions and stable matchings that satisfy assignment constraints as feasible solutions.

**Proof of correctness.**

Clearly if $V^{in} \setminus V(D^\ast) \neq \emptyset$ holds in Step 2.1, then there can be no solution in $D$ containing all the vertices in $V^{in}$, so the algorithm correctly returns without adding anything to $AllSolutions$. Furthermore, as all solutions with $(w, f)$ in the solution contain no vertices different from $(w, f)$, in the same row or column as $(w, f)$, we note that the second part of Step 2.1 does not change which solutions satisfy the constraints.

By Lemma 1 and Lemma 4 we note that every operation in Step 2.2 maintains the same set of feasible solutions.

By Lemma 2 we note that both $M_W$ and $M_F$ defined in Step 2.3 are stable matchings in $D$. As $(w, f) \not\in V^{out}$ for all $(w, f) \in V(D)$ where $d^+_{W}((w, f)) = 0$ or $d^+_{F}((w, f)) = 0$ we furthermore note that neither $M_W$ or $M_F$ contain any vertices
from $V^{\text{out}}$. If $|M_W| = r$ (which is equivalent to $|M_F| = r$ as $|M_W| = |M_F|$) then by the operation in Step 2.1 we note that all vertices in $V^{\text{in}}$ belong to both $M_W$ and $M_F$ (as no vertices from the rows or columns containing vertices in $V^{\text{in}}$ belong to $M_W$ or $M_F$). So if $|M_W| = r$ then both $M_W$ and $M_F$ are stable matchings satisfying the constraints.

If $|M_W| < r$ in Step 2.4, then clearly there is no stable matching in $D$ of size $r$, which implies that no stable matching can be added to $\text{AllSolutions}$ (as all solutions in $\text{AllSolutions}$ must have size $r$, by Step 1.2 and Lemma 3). So if Algorithm 2 returns in Step 2.4, then there is no feasible solution in $D$.

If Algorithm 2 does not return at Step 2.4, we can be sure that the digraph $D$ being considered at Step 2.5 has both $|M_W| = r$ and $|M_F| = r$, where $r$ is the size of a stable matching. If $M_W = M_F$ in Step 2.5, then $M_W$ is the unique solution in $D$. We know this to be the case by appealing to Lemma 3 and Theorem 1 in Gutin et al. (2023). That is, Lemma 3 in Gutin et al. (2023) confirms that both $M_W$ and $M_F$ are always perfect stable matchings of size $r$ in the normal form, and Theorem 1 in Gutin et al. (2023) shows that if $M_W = M_F$ then there is a unique stable matching of size $r$ in the normal form.

In Step 2.6 we note that every solution in $D$ that satisfies the constraints will be found in the recursive call in Step 2.6a if $(w, f)$ belongs to the solution and it will be found in the recursive call in Step 2.6b if $(w, f)$ does not belong to the solution. So every solution will be found in one of the recursive calls. Furthermore we note that no solution can be found in both the recursive calls in Step 2.6 as the solution either contains $(w, f)$ or it does not. Therefore all solutions will be found and no solution will be found more than once. This proves the correctness of the algorithm.

Before we prove the time complexity claim, we state and prove two useful lemmas.

We note that all steps, that do not involve a call to Algorithm 2, in Algorithm 1 and in Algorithm 2 can be performed in polynomial time. This is not difficult to see as IDUA runs in polynomial time and in Step 2.2 we cannot delete more than a polynomial number of vertices. We will now show the following:

**Lemma 5.** The number of calls to Algorithm 2 is bounded by $2s + 1$, where $s$ is the
total number of feasible solutions outputted by Algorithm 1.

Proof. Consider the binary search tree $T$ where every node corresponds to a call to Algorithm 2 and the root corresponds to the very first call to Algorithm 2. If, from an iteration of Algorithm 2 corresponding to node $x$ in $T$, we make two recursive calls in Step 2.6 then the two nodes corresponding to these two calls will be the two children of $x$ in the search tree. And if the iteration returns in Steps 2.1, 2.4 or 2.5 then $x$ is a leaf of the tree. Therefore every node of the tree either has two children or none (i.e., it is a leaf). Let $l$ denote the number of leaves in $T$. It is well-known that $|V(T)| \leq 2l - 1$ (this is true if $|V(T)| = l = 1$ and if $|V(T)| > 1$ we can use induction on the two subtrees of the root).

As the stable matching $MW$ will be found in the recursive call made in Step 2.6a and the stable matching $MF$ will be found in the recursive call made in Step 2.6b we note that the only way that a call to Algorithm 2 will not produce a solution is if it is made from Algorithm 1. Therefore every leaf, apart from possibly the root, will add a solution to $AllSolutions$. Therefore the following holds.

$$|V(T)| \leq 2l - 1 \leq 2(s + 1) - 1 = 2s + 1$$

As $|V(T)|$ is the number of calls to Algorithm 2 and $s$ was the total number of feasible solutions returned, we note that our algorithm performs a polynomial amount of work per solution found (if there is no solution the algorithm also performs a polynomial amount of work). This completes the proof.

Recall that $N = \max\{n, m\}$. Now, for each worker, $w$, let $f'(w)$ denote the firm such that $d^+_{w'}((w, f'(w))) = 0$, and for each firm, $f$, let $w'(f)$ denote the worker such that $d^+_{F}(w'(f), f)) = 0$. The workers and firms above can be found in $O(nm)$ time (we need $O(m)$ time for each row in $V(D)$ and $O(n)$ time for each column in $V(D)$).

To every vertex $(w, f)$, we assign a flag, $\theta(w, f)$, that tells us if the vertex belongs to $V^{in}$ or $V^{out}$ or neither. These flags can also be initialised in $O(nm)$ time. Another lemma required to show the desired complexity is the following:
Lemma 6. If IDUA removes \( p \) vertices from \( D \) then it runs in at most \( O(N + Np) \) time, even if it has to update \( f'(w) \) and \( w'(f) \) for some \( w \) and/or \( f \).

Proof. IDUA repeatedly performs the following steps.

- If for some \( w \) we have \( d_F((w, f'(w))) > 0 \), then remove all vertices \((w^*, f'(w))\) such that \((w^*, f'(w))(w, f'(w)) \in A_F\) and update \( f'(w^*) \) accordingly.

- If for some \( f \) we have \( d_W((w'(f), f)) > 0 \), then remove all vertices \((w'(f), f^*)\) such that \((w'(f), f^*)(w'(f), f) \in A_W\), and update \( w'(f^*) \) accordingly.

Clearly, if no vertex is removed by IDUA, then

\[
d_F((w, f'(w))) = 0 \quad \text{and} \quad d_W((w'(f), f)) = 0
\]

for all \( w \) and \( f \), and we have spent \( O(N) \) time. As updating \( f'(w^*) \) and \( w'(f^*) \) above can be done in constant time, we note that every time IDUA removes a vertex we have not spent more than \( O(N) \) time since removing the previous vertex, which completes the proof. \( \square \)

Proof of complexity. By Lemma 6 we note that Step 1.1 in Algorithm 1 can be done in time \( O(N^3) \) time (in fact it can be done in \( O(N^2) \) time, but we do not need that here). Step 1.2 can be done in \( O(N) \) time and Steps 1.3 and 1.4 can be done in constant time. So Algorithm 1 can be performed in \( O(N^3) \) time plus the complexity of Algorithm 2, which we analyse now.

In Algorithm 2, Step 2.1 can be done in \( O(N^2) \) time. We now analyse Step 2.2. By considering the vertices \((w, f'(w))\) and \((w'(f), f)\) for all \( w \) and \( f \), using the flags \( \theta(w, f) \), and using Lemma 6, we note that Step 2.2 can be performed in \( O(N^3 + Np) \) time, where \( p \) is the number of vertices that get removed in Step 2.2 and \( p \leq N^2 \). Therefore Step 2.2 can be performed in \( O(N^3) \) time. Steps 2.3, 2.4 and 2.5 can all be performed in \( O(N) \) time. Step 2.6 can be performed in \( O(N^2) \) time even if we make a new copy of our instance to be used in each of the recursive calls.
So the total time spent in one iteration of Algorithm 2 is $O(N^3)$ and as, by Lemma 5, the number of such iterations is proportional to the number of solutions found, we obtain the complexity $O(N^3 + N^3s)$.

The analyses above of the running time of Algorithm 1 without the complexity of Algorithm 2 and of one iteration of Algorithm 2 imply that the time between outputting two consecutive stable matchings by Algorithm 1 is $O(N^3)$. \hfill \Box

C Example showing that one can not just delete all vertices in $V^{out}$

![Diagram of a matching problem](image)

Figure 7: The example depicted does not change by running IDUA on it since both extreme matchings are stable matchings. (Arcs implied by transitivity are not shown).

Consider the matching problem $P$ equivalent to the digraph $D(P)$ in Figure 7. Let $V^{out} = \{(1,1), (1,3), (2,2), (2,3), (3,1), (3,2)\}$ (the dark vertices in Figure 7). One cannot just remove all vertices in $V^{out}$ and then find a stable matching in the remaining digraph. Doing so would find the matching $M = \{(1,2), (2,1), (3,3)\}$ (the
white vertices) that is stable in the remaining subdigraph. However $M$ is not a stable matching in the original problem as $(3, 1)$ is a blocking pair for $M$.

In fact, if $V^{out} = \{(3, 1)\}$ then we note that $M$ (the white vertices in Figure 7) is a stable matching in $D - (3, 1)$, but $M$ is not a stable matching in $D$ avoiding $(3, 1)$. This explains why we can only delete vertices with no horizontal arcs out of them or vertices with no vertical arcs out of them in Step 2.2 of our algorithm. This also illustrates why the condition $d_{W,i}^+(v) = 0$ (or $d_{F,i}^+(v) = 0$) is required in Lemma 4.

### D Example with exponential stable matchings

It is always possible to solve Constrained Stable Matchings by using one of the algorithms that outputs the full collection of stable matchings and then checking each against the assignment constraints. One drawback to this approach is that the number of stable matchings can be exponential and so carrying out this procedure can be highly inefficient. The inefficiency from proceeding in this manner is exacerbated if the number of stable matchings satisfying the constraints is small (or zero).

We highlight this potential issue with the following example. Suppose that the normal form of the matching market under question has the digraph depicted in Figure 8 with $r$ workers and $r$ positions.

Each 4-cycle contains two stable components, one worker optimal and one firm optimal. So the market as a whole has $2^r$ stable matchings. However, while there are exponential number of stable matchings, we will see that only four of them satisfy the assignment constraints. We have highlighted the two extremal stable matchings using colour coding as before (worker-optimal in yellow and firm-optimal in blue).

Consider constraints of the form $V^{out} = \{(k, k)\}_{k=5}^r$. That is, for all $k \geq 5$, the constraints do not permit that a worker with index $k$ be employed at a position with the same index. Referencing Figure 8, these constraints rule out any stable matching that includes a pair shaded yellow, other than those yellow vertices in positions $(1, 1), (2, 2), (3, 3)$ and $(4, 4)$. As such, from index $k = 5$ onwards, only blue vertices can be selected. We note that there are only four stable matchings satisfying the constraints.
This example is one that, to the best of our knowledge, only our algorithm can handle efficiently. There are an exponential number of stable matchings but only a small number, specifically four, satisfying the constraints. (While we concede that the example is somewhat artificial, we have chosen it as other, potentially more realistic, examples would not be so cleanly depicted.)
References

Michel Balinski and Guillaume Ratier. Of stable marriages and graphs, and strategy and polytopes. *SIAM Review*, 39(4):575–604, 1997. doi: 10.1137/S0036144595294515. URL https://doi.org/10.1137/S0036144595294515.

Agustín G. Bonifacio, Noelia Juarez, Pablo Neme, and Jorge Oviedo. Cycles to compute the full set of many-to-many stable matchings. *Mathematical Social Sciences*, 117:20–29, 2022. ISSN 0165-4896. doi: https://doi.org/10.1016/j.mathsocsci.2022.03.001. URL https://www.sciencedirect.com/science/article/pii/S0165489622000154.

Peter Chen, Michael Egesdal, Marek Pycia, and M. Bumin Yenmez. Median stable matchings in two-sided markets. *Games and Economic Behavior*, 97:64–69, 2016. ISSN 0899-8256. doi: https://doi.org/10.1016/j.geb.2016.03.004. URL https://www.sciencedirect.com/science/article/pii/S0899825616300021.

Peter Chen, Michael Egesdal, Marek Pycia, and M. Bumin Yenmez. Quantile stable mechanisms. *Games*, 12(2), 2021. ISSN 2073-4336. doi: 10.3390/g12020043. URL https://www.mdpi.com/2073-4336/12/2/43.

Christine T. Cheng. Understanding the generalized median stable matchings. *Algorithmica*, 58(1):34–51, 2010. doi: 10.1007/s00453-009-9307-2. URL https://doi.org/10.1007/s00453-009-9307-2.

Vânia M.F. Dias, Guilherme D. da Fonseca, Celina M.H. de Figueiredo, and Jayme L. Szwarcfiter. The stable marriage problem with restricted pairs. *Theoretical Computer Science*, 306(1):391–405, 2003. ISSN 0304-3975. doi: https://doi.org/10.1016/S0304-3975(03)00319-0. URL https://www.sciencedirect.com/science/article/pii/S0304397503003190.

Piotr Dworczak. Deferred acceptance with compensation chains. *Operations Research*, 69(2):456–468, 2021. doi: 10.1287/opre.2020.2042. URL https://doi.org/10.1287/opre.2020.2042.
Federico Echenique, Alejandro Robinson-Cortés, and Leeat Yariv. An experimental study of decentralized matching, 2024.

Lars Ehlers, Isa E. Hafalir, M. Bumin Yenmez, and Muhammed A. Yildirim. School choice with controlled choice constraints: Hard bounds versus soft bounds. *Journal of Economic Theory*, 153:648–683, 2014. ISSN 0022-0531. doi: https://doi.org/10.1016/j.jet.2014.03.004. URL https://www.sciencedirect.com/science/article/pii/S0022053114000301.

Tamás Fleiner. Some results on stable matchings and fixed points. *Egerváry Res. Group, Budapest, Hungary, Tech. Rep. TR-2002-08*, 2002.

D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962. ISSN 00029890. URL http://www.jstor.org/stable/2312726.

David Gale and Marilda Sotomayor. Ms. machiavelli and the stable matching problem. *The American Mathematical Monthly*, 92(4):261–268, 1985a. ISSN 00029890, 19300972. URL http://www.jstor.org/stable/2323645.

David Gale and Marilda Sotomayor. Some remarks on the stable matching problem. *Discrete Applied Mathematics*, 11(3):223–232, 1985b. ISSN 0166-218X. doi: https://doi.org/10.1016/0166-218X(85)90074-5. URL https://www.sciencedirect.com/science/article/pii/0166218X85900745.

Vijay K. Garg. Applying predicate detection to the constrained optimization problems, 2018. URL https://arxiv.org/abs/1812.10431.

Vijay K. Garg. Predicate detection to solve combinatorial optimization problems. In *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '20, pages 235–245, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450369350. doi: 10.1145/3350755.3400235. URL https://doi.org/10.1145/3350755.3400235.

47
Dan Gusfield. Three fast algorithms for four problems in stable marriage. *SIAM Journal on Computing*, 16(1):111–128, 1987. doi: 10.1137/0216010. URL https://doi.org/10.1137/0216010.

Dan Gusfield and Robert W. Irving. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, Cambridge, MA, USA, 1989. ISBN 0262071185.

Gregory Z. Gutin, Philip R. Neary, and Anders Yeo. Unique stable matchings. *Games and Economic Behavior*, 141:529–547, 2023. ISSN 0899-8256. doi: https://doi.org/10.1016/j.geb.2023.07.010. URL https://www.sciencedirect.com/science/article/pii/S0899825623001033.

Robert W. Irving and Paul Leather. The complexity of counting stable marriages. *SIAM Journal on Computing*, 15(3):655–667, 1986. doi: 10.1137/0215048. URL https://doi.org/10.1137/0215048.

Yuichiro Kamada and Fuhito Kojima. Efficient matching under distributional constraints: Theory and applications. *The American Economic Review*, 105(1):67–99, 2015. ISSN 00028282. URL http://www.jstor.org/stable/43497054.

Yuichiro Kamada and Fuhito Kojima. Recent developments in matching with constraints. *American Economic Review*, 107(5):200–204, May 2017. doi: 10.1257/aer.p20171047. URL https://www.aeaweb.org/articles?id=10.1257/aer.p20171047.

Bettina Klaus and Flip Klijn. Stable matchings and preferences of couples. *Journal of Economic Theory*, 121(1):75–106, 2005. ISSN 0022-0531. doi: https://doi.org/10.1016/j.jet.2004.04.006. URL https://www.sciencedirect.com/science/article/pii/S0022053104001085.

Bettina Klaus and Flip Klijn. Median stable matching for college admissions. *International Journal of Game Theory*, 34(1):1–11, 2006. doi: 10.1007/s00182-006-0009-6. URL https://doi.org/10.1007/s00182-006-0009-6.
Bettina Klaus, Flip Klijn, and Toshifumi Nakamura. Corrigendum to “stable matchings and preferences of couples” [j. econ. theory 121 (1) (2005) 75–106]. *Journal of Economic Theory*, 144(5):2227–2233, 2009. ISSN 0022-0531. doi: https://doi.org/10.1016/j.jet.2009.06.001. URL https://www.sciencedirect.com/science/article/pii/S0022053109000738.

D.E. Knuth. *Stable Marriage and Its Relation to Other Combinatorial Problems: An Introduction to the Mathematical Analysis of Algorithms*. CRM proceedings & lecture notes. American Mathematical Soc., 1996. ISBN 9780821870129. URL https://books.google.co.uk/books?id=HqyCIky3bHwC.

Fuhito Kojima, Parag A. Pathak, and Alvin E. Roth. Matching with Couples: Stability and Incentives in Large Markets*. *The Quarterly Journal of Economics*, 128(4):1585–1632, 08 2013. ISSN 0033-5533. doi: 10.1093/qje/qjt019. URL https://doi.org/10.1093/qje/qjt019.

Frédéric Maffray. Kernels in perfect line-graphs. *Journal of Combinatorial Theory, Series B*, 55(1):1–8, 1992. ISSN 0095-8956. doi: https://doi.org/10.1016/0095-8956(92)90028-V. URL https://www.sciencedirect.com/science/article/pii/009589569290028V.

Pinaki Mandal and Souvik Roy. Matchings under Stability, Minimum Regret, and Forced and Forbidden Pairs in Marriage Problem. MPRA Paper 107213, University Library of Munich, Germany, April 2021. URL https://ideas.repec.org/p/pra/mprapa/107213.html.

Ruth Martínez, Jordi Massó, Alejandro Neme, and Jorge Oviedo. An algorithm to compute the full set of many-to-many stable matchings. *Mathematical Social Sciences*, 47(2):187–210, 2004. ISSN 0165-4896. doi: https://doi.org/10.1016/j.mathsocsci.2003.07.002. URL https://www.sciencedirect.com/science/article/pii/S0165489603000891.

D. G. McVitie and L. B. Wilson. Stable marriage assignment for unequal sets. *BIT*
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. doi: 10.1086/261272. URL https://doi.org/10.1086/261272.

Alvin E. Roth. On the allocation of residents to rural hospitals: A general property of two-sided matching markets. *Econometrica*, 54(2):425–427, 1986. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1913160.

Alvin E. Roth. Deferred acceptance algorithms: history, theory, practice, and open questions. *International Journal of Game Theory*, 36(3):537–569, 2008.

Alvin E. Roth and John H. Vande Vate. Random paths to stability in two-sided matching. *Econometrica*, 58(6):1475–1480, 1990. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/2938326.

Kirill Rudov. Fragile stable matchings, 2024.

Jay Sethuraman, Chung-Piaw Teo, and Liwen Qian. Many-to-one stable matching: Geometry and fairness. *Mathematics of Operations Research*, 31(3):581–596, 2006. ISSN 0364765X, 15265471. URL http://www.jstor.org/stable/25151746.

Chung-Piaw Teo and Jay Sethuraman. The geometry of fractional stable matchings and its applications. *Mathematics of Operations Research*, 23(4):874–891, 1998. ISSN 0364765X, 15265471. URL http://www.jstor.org/stable/3690636.

John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior (60th Anniversary Commemorative Edition)*. Princeton University Press, 1944. ISBN 9780691130613. URL http://www.jstor.org/stable/j.ctt1r2gkx.