

# The Dichotomous Affiliate Stable Matching Problem: Approval-Based Matching with Applicant-Employer Relations

Marina Knittel, Samuel Dooley, John P. Dickerson

Computer Science Department, University of Maryland

{mknittel, sdooley1, john}@cs.umd.edu

## Abstract

While the stable marriage problem and its variants model a vast range of matching markets, they fail to capture complex agent relationships, such as the affiliation of applicants and employers in an interview marketplace. To model this problem, the existing literature on matching with externalities permits agents to provide complete and total rankings over matchings based off of both their own and their affiliates' matches. This complete ordering restriction is unrealistic, and further the model may have an empty core. To address this, we introduce the Dichotomous Affiliate Stable Matching (DASM) Problem, where agents' preferences indicate dichotomous acceptance or rejection of another agent in the marketplace, both for themselves and their affiliates. We also assume the agent's preferences over entire matchings are determined by a general weighted valuation function of their (and their affiliates') matches. Our results are threefold: (1) we use a human study to show that real-world matching rankings follow our assumed valuation function; (2) we prove that there always exists a stable solution by providing an efficient, easily-implementable algorithm that finds such a solution; and (3) we experimentally validate the efficiency of our algorithm versus a linear-programming-based approach.

## 1 Introduction

In many markets, two classes of participants seek to be paired with each other. For example, in labor markets, workers are paired with firms [Perrault *et al.*, 2016]; in online advertising, eyeballs are paired with advertisements [Shen *et al.*, 2020; Dickerson *et al.*, 2019]; and, in morally-laden settings such as refugee resettlement and organ donation, refugees are paired with new housing locations [Jones and Teytelboym, 2018] and donors are paired with needy recipients [Ashlagi and Roth, 2021; Li *et al.*, 2014], respectively. The field of market design purports to provide analytically-sound and empirically-validated approaches to the design and fielding of such matching markets, and necessarily joins fields such as economics and computer science [Roth, 2002; Roth, 2018].

The seminal work of Gale and Shapley [1962] characterized the stable marriage problem, where both sides of a market—workers and firms, refugees and settlement locations, etc.—express preferences over the other side, and the goal is to find a robust matching that does not unravel in the face of agents' selfish behavior. Myriad generalizations were proposed in the following decades; see Manlove [2013] for an overview of the history and variants of these problems. Largely, these models assume that agents' preferences only consider the direct impact of an outcome on that agent.

One extension of stable marriage is matching with externalities wherein agents on each side of a two-sided market have preferences over their own match *and* the matches of others. These models often incorporate many more realistic and complex assumptions which makes for a richer and harder to analyze matching setting [Pycia, 2012; Echenique and Yenmez, 2007; Baccara *et al.*, 2012]. Sasaki and Toda [1996] first introduced matching with externalities, where agents' decisions to deviate from a proposed match depended on reasonable assumptions for the reaction of other agents to the deviation. Hafalir [2008] and Mumcu and Saglam [2010] expand upon this stability notion for one-to-one matchings with further restrictions on agent behavior; while Bando [2012; 2014] extends the analysis to many-to-one matchings where firms consider other firms' externalities.

Much work in the matching with externalities literature focuses on the appropriateness of various stability definitions. Much analysis then centers on the complexity and hardness of the proposed matching algorithms. For instance, Brânzei [2013] models agent values in matching with externalities as arbitrary functions and creates a valuation as a sum over the agent's values over all matches. In our work, an agent values a match as either acceptable or unacceptable (dichotomously), and we do a (weighted) sum over all relevant matches for the agent to get their valuation. While there is existing work on the complexities of these matchings, eliciting general preferences over a complex market can be intractable, both with respect to human ability and computational/communication complexity [Rastegari *et al.*, 2016; Sandholm and Boutilier, 2006]. One commonly imposed assumption is that of dichotomous preferences [Bogomolnaia and Moulin, 2004], which coarsely places alternatives into acceptable or unacceptable bins.

This work is inspired by Dooley and Dickerson [2020], which explores matching with externalities in academic fac-

ulty hiring; however, in our work, we analyze the marketplace with dichotomous preferences. Our main motivation is the academic faculty *interview* marketplace, where we match interview slots for universities and graduating students, and universities care about their graduating students' matches. Other motivations include playdate matching, study abroad, student project allocation, and the dog breeding market.

We note that the only simplification we introduce to the Dooley and Dickerson model is that of binary preferences. This assumption is prevalent in various matching settings like resource allocation [Ortega, 2020] and more specifically in the allocation of unused classrooms in a school setting [Kurokawa *et al.*, 2018] and barter exchange [Aziz, 2020]. With this additional assumption, we are able to provide positive and constructive principled approaches to clearing (dichotomous) affiliate matching markets.

**Our contributions.** We view our contributions as follows.

- We introduce the Dichotomous Affiliate Stable Matching (DASM) Problem, which characterizes the affiliate matching problem under dichotomous preferences to better accommodate realistic preference elicitation constraints, along with a valuation function for agents to rank matches based off their preferences, parameterized by an employer's relative valuation of its affiliates' and its own matches (§2);
- We run a human survey to provide support for the model design choices, showing that real people in some situations may, indeed, adhere to our valuation function under different parameters (§3);
- We propose an efficient algorithm to solve the DASM Problem (§4), i.e., yield a stable matching; and
- We perform experimental validation of our algorithmic approach to verify its correctness and scalability (§5).

## 2 Model Definition

We now present our matching model. This model represents hiring interview markets where applicants have previous affiliations with employers and preferences are encoded as binary values that denote interest or disinterest (i.e., *dichotomous preferences* [Bogomolnaia and Moulin, 2004]). We describe the model in the most general many-to-many setting. We formalize the model, including defining valuation functions (§2.1), stability, and other useful concepts (§2.2).

### 2.1 The Dichotomous Affiliate Stable Matching Problem

In the Dichotomous Affiliate Stable Matching (DASM) Problem, we are given sets  $A$  of  $n$  applicants and  $E$  of  $m$  employers. For every  $a \in A$  (resp.  $e \in E$ ), we are given a complete preference function  $\text{pr}_a : E \rightarrow \{0, 1\}$  (resp.  $\text{pr}_e^a : A \rightarrow \{0, 1\}$ , the notational difference will be clear later). If  $u$  and  $v$  are on opposite sides, then we say  $u$  is *interested in* or *likes*  $v$  if  $\text{pr}_u(v) = 1$  (or  $\text{pr}_u^u(v) = 1$  if  $u \in E$ ), otherwise  $u$  is *disinterested in*  $v$ . The *many-to-many* matching scenario specifies that  $u$  might be matched with as many as  $q(u)$  agents on the other side of the market, where  $q(u)$  is the *capacity* of  $u$ . A valid matching is a function  $\mu : A \cup E \rightarrow 2^{A \cup E}$  such that for any  $a \in A$  (resp.  $e \in E$ ):  $\mu(a) \subseteq E$  (resp.  $\mu(e) \subseteq A$ ),  $|\mu(a)| \leq q(a)$  (resp.  $|\mu(e)| \leq q(e)$ ), and  $e \in \mu(a)$  if and only if  $a \in \mu(e)$ .

One defining aspect of this market is the notion of *affiliates* which represent previous relationships between agents. Let  $\text{aff} : E \rightarrow 2^A$  return an employer's set of affiliate. For instance, in Figure 1,  $a_1$  is  $e_1$ 's affiliate, and  $a_2$  and  $a_3$  are both  $e_2$ 's affiliates. Then  $\text{aff}(e_1) = \{a_1\}$ ,  $\text{aff}(e_2) = \{a_2, a_3\}$ , and  $\text{aff}(e_3) = \emptyset$ . Note that  $\text{aff}$  over all  $e \in E$  forms a disjoint cover of  $A$ , so each applicant is the affiliate of exactly one employer. In this model,  $e$  cares about its affiliates' matches. To express this, for any  $a \in \text{aff}(e)$ ,  $e$  has preferences  $\text{pr}_e^a : E \rightarrow \{0, 1\}$ . To account for these preferences,  $e$ 's valuation of matchings is over tuples of its own *and* its affiliates' matches. While these valuations may be general, we will examine a natural and flexible additive valuation method.

**Definition 1.** For any  $e \in E$  and  $a \in A$ , we define the **weighted valuation function** over a match  $\mu$  for a given weight  $\lambda \in [0, 1]$  as:

$$\begin{aligned} \text{val}_e(\mu) &= \sum_{a^* \in \mu(e)} \text{pr}_e^e(a^*) + \lambda \sum_{a_i \in \text{aff}(e)} \sum_{e^* \in \mu(a_i)} \text{pr}_e^{a_i}(e^*) \\ \text{val}_a(\mu) &= \sum_{e^* \in \mu(a)} \text{pr}_a(e^*). \end{aligned}$$

And we say  $e$  or  $a$  **prefers**  $\mu$  to  $\mu'$  ( $\mu \succ_e \mu'$  or  $\mu \succ_a \mu'$ ) if and only if  $\text{val}_e(\mu) > \text{val}_e(\mu')$  or  $\text{val}_a(\mu) > \text{val}_a(\mu')$ .

This function does not necessarily create a total order over matchings, as an agent may have the same valuation for two distinct matchings. Then it does not prefer one matching to another. In the employer version,  $\lambda$  parameterizes how employers weigh the value of their affiliates' matches with respect to their own matches. If  $\lambda = 1$ , then an employer cares about each affiliate match as much as each of its own matches. If  $\lambda = 0$ , employers do not care about their affiliates' matches. Setting  $\lambda = \epsilon$  for small  $\epsilon > 0$  yields a lexicographic valuation where employers care about their matches first and only use affiliates' matches as tiebreakers.

To understand the role of  $\lambda$ , consider again our example in Figure 1 and let  $\lambda = 1$ . Then  $e_2$ 's valuations of  $\mu$  and  $\mu'$  are:  $\text{val}_{e_2}(\mu) = 3$  (since it likes one of its matches and both of its affiliates' matches) and  $\text{val}_{e_2}(\mu') = 2$  (since it likes both of its matches). Therefore  $\mu \succ_{e_2} \mu'$ . If  $\lambda = \epsilon$ , then  $e_2$ 's valuations of the matchings are:  $\text{val}_{e_2}(\mu) = 1 + 2\epsilon$  and  $\text{val}_{e_2}(\mu') = 2$ . This means  $\mu \prec_{e_2} \mu'$ . As we discuss later, this illustrates how  $\lambda$  can affect which matchings are stable.

### 2.2 Blocking Tuples and Stability

Next, we define the notion of stability in the DASM Problem. As in the standard stable marriage problem [Gale and Shapley, 1962] and its many variants, our notion of stability relies on the (non)existence of *blocking tuples*. The blocking tuple—traditionally, blocking *pair*—is designed to identify a set of unmatched individuals who might cheat in order to match with each other. We formalize cheating as follows:

**Definition 2.** Consider an instance of the DASM Problem with matching  $\mu$ . An agent  $a \in A \cup E$  **cheats** if they break a match with some agent  $a' \in \mu(a)$ .

Like in the standard stable marriage problem, we would

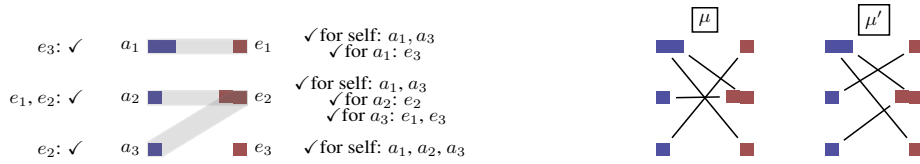


Figure 1: An example matching problem. On the left are affiliations and preferences for each agent. The capacity of each agent is represented with squares. For instance,  $e_2$  has  $\text{aff}(e_2) = \{a_2, a_3\}$ ,  $q(e_2) = 2$ , and it approves of  $a_1$  and  $a_3$  for itself,  $e_2$  for affiliate  $a_2$ , and  $e_1$  and  $e_3$  for affiliate  $a_3$ . On the right, we have a potential matching  $\mu$  with an alternate matching  $\mu'$ . Note that  $\mu'$  is the swapped matching of  $\mu$  with respect to  $\mathcal{T} = (a_3, a_2, a_2, e_2, e_1, e_1)$ .

like to ensure that no two agents<sup>1</sup>  $a \in A$  and  $e \in E$  will agree to cheat on their assigned match to (possibly) match with each other assuming no other agents cheat. In stable marriage, the only way (up to) two agents will cheat, and thus cause instability, is if they prefer each other to their matches.

In our setting, stability is not as simple. Let  $a \in A$  and  $e \in E$  be two agents who consider cheating on matches  $e' \in \mu(a)$  and  $a' \in \mu(e)$  respectively. Once  $e'$  and  $a'$  lose their matches with  $a$  and  $e$ , they could naturally consider filling that empty space in their capacities by matching with each other (this is not cheating). In stable marriage, we do not need to model  $e'$ 's and  $a'$ 's reaction because this does not impact  $e$ 's and  $a$ 's decisions to cheat. In the DASM Problem, however,  $e$ 's preference profile is more complicated. For instance, if  $a' \in \text{aff}(e)$ , and  $e$  wants  $a'$  to be matched with  $e'$ ,  $e$  might decide to cheat on  $a'$  so that  $a'$  will fill its new empty capacity by matching with  $e'$ . Note that this still does not require  $a'$  or  $e'$  to cheat, since they are only forming matches instead of breaking them.

To this end, we define our notion of the blocking tuple to capture not only the cheating between two agents, but also the responses of those whose matches have been broken.

**Definition 3.** Consider an instance of the DASM Problem with matching  $\mu$  and some tuple  $\mathcal{T} = (a_1, \dots, a_k)$  where  $a_i \in A \cup E$  for all  $i \in [k]$ . Construct  $\mu'$  with the following process:

1. Allow up to two agents in  $\mathcal{T}$  to cheat on one match each. If exactly two agents cheat, they then match with each other. Otherwise, the sole cheater may match assuming this new match does not violate any capacities.
2. All other agents in  $\mathcal{T}$  are then allowed to form new matches up to their capacity.

Then  $\mathcal{T}$  is a **blocking tuple** if: (1) all cheating agents strictly prefer  $\mu'$  to  $\mu$ , and (2) for any other agent  $a \in \mathcal{T}$  that forms a match with  $a' \in \mathcal{T}$ ,  $a$  strictly prefers  $\mu'$  to  $\mu \setminus \{(a, a')\}$ . An instance of the DASM Problem is **stable** if and only if it contains no blocking tuples.

In stable marriage, this becomes the same standard notion of cheating, and therefore this is a natural extension of the stable marriage blocking pair. In Definition 3, a cheating agent would only cheat if they know other agents could respond in a way such that the resulting matching is preferable to the original matching. After cheating occurs, non-cheating agents will respond with a match if they would prefer to have that

<sup>1</sup>In future work, this could be generalized to *coalitions*, or larger sets of agents.

match in the final matching. Thus, non-cheating agents are not performing calculated activity; they are simply reacting.

Interestingly, we only need to consider tuples of size at most six that satisfy certain properties to determine if a blocking tuple exists. See Proposition 2 and Appendix B for reasoning and a proof. These properties are captured by the *potential blocking tuple*, a tuple of size at most six with the appropriate agents such that, given the right preference profiles, they could be blocking tuples. To accommodate the sextuplet notation, we introduce the “empty agent”,  $\gamma$ , and a set  $\mathcal{E} = \{\gamma\}$ , which represents the absence of an agent. Formally,  $\gamma$  is an agent in the system with no side or affiliation and zero capacity. Additionally, given a matching  $\mu$ , we must notate the agents who have remaining capacity. Let  $N_\mu^A = \{a \in A : |\mu(a)| < q(a)\}$  and  $N_\mu^E = \{e \in E : |\mu(e)| < q(e)\}$ .

**Definition 4.** Consider an instance of the DASM Problem with matching  $\mu$ . A tuple  $\mathcal{T} = (a, a', a'', e, e', e'')$  is a **potential blocking tuple** for  $\mu$  if all the following hold:

1.  $a \in A$
2.  $e \in E \setminus \mu(a)$
3.  $a' \in \mu(e) \cup \mathcal{E}$ , where  $a' \in \mathcal{E}$  only if  $e \in N_\mu^E$
4.  $e' \in \mu(a) \cup \mathcal{E}$ , where  $e' \in \mathcal{E}$  only if  $a \in N_\mu^A$
5.  $a'' \in (N_\mu^A \cup \mathcal{E} \cup \{a'\}) \setminus \mu(e')$ , where  $a'' \in \mathcal{E}$  if  $e' \in \mathcal{E}$
6.  $e'' \in (N_\mu^E \cup \mathcal{E} \cup \{e'\}) \setminus \mu(a')$ , where  $e'' \in \mathcal{E}$  if  $a' \in \mathcal{E}$
7.  $a'' = a' \notin \mathcal{E}$  if and only if  $e'' = e' \notin \mathcal{E}$

We now clarify the purpose of each condition respectively:

1.  $a$  must be an applicant.
2.  $e$  must be an employer that is not matched with  $a$  (else they cannot form a blocking tuple).
3.  $a'$  is  $e$ 's old match that is broken. If  $e$  simply has additional capacity, then  $a' \in \mathcal{E}$  is an empty agent.
4.  $e'$  is  $a$ 's old match that is broken. If  $a$  simply has additional capacity, then  $e' \in \mathcal{E}$  is an empty agent.
5.  $a''$  is  $e'$ 's new match. It must have unmatched capacity, or (if  $e'$  and  $a'$  decide to match) is instead  $a'$  itself. It could be an empty agent if  $e'$  does not rematch (and must be if  $e'$  is an empty agent). Additionally, we must ensure it was not previously matched to  $e'$ .
6.  $e''$  is  $a'$ 's new match. It must have unmatched capacity, or (if  $e'$  and  $a'$  decide to match) is instead  $e'$  itself. It could be an empty agent if  $a'$  does not rematch (and must be if  $a'$  is an empty agent). Additionally, we must ensure it was not previously matched to  $a'$ .

7. If  $e'$  matches with  $a'$  (where  $a' = a'' \notin \mathcal{E}$ ), then  $a'$  must match with  $e'$  (where  $e' = e'' \notin \mathcal{E}$ ).

Consider Figure 1 and tuple  $(a_3, a_2, a_2, e_2, e_1, e_1)$ . Some agents are duplicated in this tuple; this is okay. This potential blocking tuple describes the following changes: (1)  $a_3$  breaks its match with  $e_1$ , (2)  $e_2$  breaks its match with  $a_2$ , (3)  $a_3$  and  $e_2$  match together, and (4)  $a_2$  and  $e_1$  match together. Note that the last part occurs because  $a_2$  and  $e_1$  appear twice in the tuple. If we rewrite the tuple as  $(a_3, a_2, a'_2, e_2, e_1, e'_1)$  to distinguish duplicate instances, then  $a'_2$  indicates that  $e_1$  matches with  $a_2$  and  $e'_1$  indicates  $a_2$  matches with  $e_1$ .

Definition 4’s matching constraints ensure that broken and formed matches in this process make sense (e.g., no two agents will be matched to each other twice). When we consider a blocking tuple, we must compare the matching to the alternative matching that occurs after swapping as described.

**Definition 5.** Consider an instance of the DASM Problem with matching  $\mu$  and a potential blocking tuple  $\mathcal{T} = (a, a', a'', e, e', e'')$ . Let  $\mu'$  be defined by starting at  $\mu$ , breaking the matches  $(a, e')$  and  $(a', e)$ , adding match  $(a, e)$ , and adding matches  $(a', e'')$  and  $(a'', e')$  if and only if those variables are not in  $\mathcal{E}$  respectively. Then  $\mu'$  is the **swapped matching** of  $\mu$  with respect to  $\mathcal{T}$ .

In Figure 1,  $\mu'$  is the swapped matching of  $\mu$  with respect to  $(a_3, a_2, a_2, e_2, e_1, e_1)$ . Note that it is a valid matching.

**Proposition 1.** If  $\mu$  is a matching and  $\mu'$  is the swapped matching of  $\mu$  with respect to some potential blocking tuple  $\mathcal{T}$ , then  $\mu'$  is a matching.

See Appendix B for a proof. Now we show that the set of potential blocking tuples are sufficient consideration to show that an instance of the DASM Problem is unstable.

**Proposition 2.** Consider an instance of the DASM Problem with matching  $\mu$ . Then  $\mu$  is unstable if and only if there exists a potential blocking tuple  $\mathcal{T} = (a, a', a'', e, e', e'')$  with respective swapped matching  $\mu'$  for  $\mu$  such that:

1.  $\mu \prec_a \mu'$
2.  $\mu \prec_e \mu'$
3. If  $a' \notin \mathcal{E}$ , then  $\mu' \setminus \{(a', e'')\} \prec_{a'} \mu'$
4. If  $e' \notin \mathcal{E}$ , then  $\mu' \setminus \{(a'', e')\} \prec_{e'} \mu'$
5. If  $a'' \notin \mathcal{E}$ , then  $\mu' \setminus \{(a'', e')\} \prec_{a''} \mu'$
6. If  $e'' \notin \mathcal{E}$ , then  $\mu' \setminus \{(a', e'')\} \prec_{e''} \mu'$

See Appendix B for a proof. In the rest of the paper, we assume all blocking tuples take this form. The first two conditions ensure that  $a$  and  $e$  prefer the new match  $\mu'$  to  $\mu$ . The next four state that in the context of  $\mu'$ , all of  $a', e', a''$ , and  $e''$  must actively desire the new match. In these conditions, we only care if  $a', a'', e'$ , and  $e''$  are not in  $\mathcal{E}$  (i.e., they exist).

Consider again Figure 1 when  $\lambda = 1$ . Recall that  $\mathcal{T} = (a_3, a_2, a_2, e_2, e_1, e_1)$  is a potential blocking tuple for  $\mu$  and  $\mu'$  is the swapped matching of  $\mu$  with respect to  $\mathcal{T}$ . Under our weighted valuation function with  $\lambda = 1$ , we already showed that  $e_2$  prefers  $\mu'$  to  $\mu$ . Additionally, since  $a_3$  doesn’t like its match in  $\mu$  but likes its match in  $\mu'$ , it also prefers  $\mu'$ . Once  $a_3$  and  $e_2$  have broken their matches with  $a_2$  and  $e_1$  respectively,  $a_2$  and  $e_1$  have an active interest in matching. This implies that  $\mathcal{T}$  satisfies all constraints in Proposition 2

Scenario	$\lambda = 1$							
	1	2	3	4	5	6	7	8
Random	10%	5%	5%	3%	100%	10%	10%	3%
Observed Unprimed	<b>41%</b>	31%	30%	<b>56%</b>	100%	42%	<b>25%</b>	30%
Observed Primed	<b>41%</b>	<b>44%</b>	<b>35%</b>	55%	100%	<b>51%</b>	22%	<b>45%</b>
Scenario	$\lambda = \epsilon$							
	1	2	3	4	5	6	7	8
Random	3%	1%	1%	3%	20%	3%	3%	1%
Observed Unprimed	14%	15%	18%	<b>56%</b>	30%	17%	<b>16%</b>	17%
Observed Primed	<b>18%</b>	<b>34%</b>	<b>20%</b>	55%	<b>37%</b>	<b>28%</b>	10%	<b>34%</b>

Table 1: Percentage of respondents who followed a weighted valuation function with  $\lambda = 1$  and  $\lambda = \epsilon$  for both primed and unprimed subjects. These are compared to the expected percentage if individuals were choosing randomly.

and is a blocking tuple. When  $\lambda = \epsilon$ , since  $e_2$  does not prefer  $\mu'$  to  $\mu$ , this is not a blocking tuple.

### 3 Evidence from a Human Experiment

This survey strives to evaluate the applicability of our valuation function proposed for the DASM Problem. More details of methods and results can be found in Appendix C.

In the survey, participants were asked to identify as a university in a DASM Problem instance with an affiliated graduating student. Across multiple problem instances, the survey presented the user with binary preferences over relevant matches and five possible matchings. It then asked the users to rank the matchings. For each question, we found the preference profiles that emerge from our weighted valuation function when  $\lambda \in \{\epsilon, 1\}$  and then computed: (1) the chance of randomly selecting the profiles, and (2) empirical adherence to the profiles. These results are depicted in Table 1.

Our results show that participants’ ranking adherence to each valuation function is statistically significant, though not consistent. For a deeper quantitative analysis, see Appendix C. More qualitatively, participants expressed differing philosophies. Some participants were very direct with their strategies, even stating: “My needs first, then Ryan’s”, where Ryan is the example affiliate. We see that our two valuation function versions align with this general strategy. On the other end of the spectrum, there were participants who were uncomfortable with the ability to express a preference over their affiliate’s match. One participant said, “If Ryan doesn’t get matched with my school, why would I care what others he matched with? Is it any of my business?” This indicates that there are clearly different strategies, but also different philosophical approaches to the affiliate matching problem.

### 4 DASM Solved in Quadratic Time

We now introduce a quadratic (in the number of agents) algorithm, **SmartPriorityMatch**, to solve the DASM Problem for general weights. We assume the inputs are provided as a set  $A$  of applicants and  $E$  of employers, where each agent reports all appropriate approval lists (i.e., lists of binary values). Let  $n = |A|$  and  $m = |E|$ . All proofs are in the Appendix.

**Theorem 1.** *SmartPriorityMatch* solves the DASM Problem in  $O(nm)$  time for  $\lambda \in [0, 1]$ .

Theorem 1 is proved at the end of Appendix D.2. **SmartPriorityMatch**, shown in Algorithm 1 in Appendix D.4, ef-

fectively puts a “priority level” on each applicant-employer pair. For instance, a pair  $(a, e) \in A \times E$  where  $a \in \text{aff}(e)$  and each agent is maximally interested in the match ( $\text{pr}_e^e(a) = 1$ ,  $\text{pr}_a^e(e) = 1$ , and  $\text{pr}_a(e) = 1$ ) is a “highest” priority edge. For each priority level, we construct bipartite graphs with partitions  $A$  and  $E$  where edges correspond to pairs of that priority level. The edge sets for the different priority levels (from highest to lowest priority) are as follows:

- $G_0$ - Edges between an  $e \in E$  and  $a \in \text{aff}(e)$  if they have maximum interest for the match:  $\text{pr}_e^e(a) = 1$ ,  $\text{pr}_a^e(e) = 1$ , and  $\text{pr}_a(e) = 1$ .
- $G_1$ - Edges between an  $e \in E$  and  $a \in A \setminus \text{aff}(e)$  if they are interested in each other:  $\text{pr}_e^e(a) = 1$  and  $\text{pr}_a(e) = 1$ .
- $G_2$ - Edges between an  $e \in E$  and  $a \in \text{aff}(e)$  if they are interested in each other for their own match:  $\text{pr}_e^e(a) = 1$  and  $\text{pr}_a(e) = 1$ .
- $G_3$ - Edges between an  $e \in E$  and  $a \in \text{aff}(e)$  if  $e$  is interested in itself for  $a$  and  $a$  is interested in  $e$ :  $\text{pr}_e^e(e) = 1$  and  $\text{pr}_a(e) = 1$ .

Next, we would *like* to run simple maximal  $b$ -matchings (i.e., many-to-many matchings) on these graphs in this order, decreasing the quotas as matches are made. Unfortunately, this method cannot ensure stability. Call this algorithm `PriorityMatch`. While this does not provide us with the desired results, it will set a strong foundation for `SmartPriorityMatch`.

**Lemma 1.** *There exists a DASM Problem instance where `PriorityMatch` may not find a stable matching.*

See Appendix D.2 for a proof. Intuitively, `PriorityMatch`’s fault is that it is not sufficiently forward-looking. For instance, an employer  $e$  that can match with one of two of its affiliates  $a_1$  and  $a_2$  in  $G_0$  cannot greedily distinguish between the two. Therefore it could arbitrarily match with  $a_1$ , and  $a_2$  could match with some other employer  $e'$  in  $G_1$ . Perhaps  $e$  likes the match  $(e', a_1)$  and not  $(e', a_2)$ , in which case it should have matched with  $a_2$  and let  $a_1$  match with  $e'$ . This creates a blocking tuple  $(e, e', e', a_2, a_1, a_1)$ . This problem only arises when an employer might match with its affiliates who have the opportunity to match with other employers later on, which only happens on  $G_0$ . However, we find that `PriorityMatch` *could* find a stable matching for these examples given a smart enough way to find the maximal  $b$ -matchings.

**Lemma 2.** *`PriorityMatch` solves the DASM Problem with parameter  $\lambda \in [0, 1]$  in  $O(nm)$  time if it can ensure that for any potential blocking tuple  $\mathcal{T} = (a, a', a'', e, e', e'')$  such that  $a, a' \in \text{aff}(e)$  and  $a$  prefers the swapped matching of  $\mu$  with respect to  $\mathcal{T}$ , then:*

$$\text{pr}_e^e(a') + \lambda \text{pr}_e^e(e') + \lambda \text{pr}_e^{a'}(e) \geq \text{pr}_e^e(a) + \lambda \text{pr}_e^e(e) + \lambda \text{pr}_e^{a'}(e'').$$

See Appendix D.2 for a proof. To achieve this, our maximal  $b$ -matchings must be dependent on lower-priority graphs. Instead of running the maximal  $b$ -matchings in order, we will use *reserved matchings*, defined as follows.

**Definition 6.** *Consider a graph  $G = (V, E, q, \mathcal{S}, r)$ , where  $V, E$ , and  $q$  are the standard  $b$ -matching parameters,  $\mathcal{S} \subseteq 2^V$  is the set of **affiliations**, and  $r : \mathcal{S} \rightarrow \mathbb{N}$  is a **reservation function** such that  $r(S) \leq |S|$  for all  $S \in \mathcal{S}$ . A **reserved***

**maximal  $b$ -matching**  $\mu$  is a  $b$ -matching that is maximal under the additional constraint that for each  $S \in \mathcal{S}$ , there are at least  $r(S)$  elements in  $S$  that have not met their capacity:  $|\{s \in S : |\mu(s)| < q(s)\}| \geq r(S)$ .

Consider an affiliation with 10 vertices, each with 100 capacity. The affiliation might have a reservation of 9 (of a max possible 10). We could match each vertex in the affiliation 99 times and one vertex 100 times. Only one vertex has reached its capacity, thus satisfying the reservation. We defer to Appendix D.1 for a simple greedy solution to this problem.

Our algorithm starts with a reserved matching on  $G_1$ , where reservations are used to ensure we can still find a maximal matching on  $G_0$  afterwards. Since each  $a \in A$  may only be adjacent to  $\text{aff}^{-1}(a)$  in  $G_0$ ,  $G_0$  is a set of disjoint stars with centers  $e \in E$ . This lack of interference allows  $e$  to match to any subset  $S \subseteq N_0(e)$  of size exactly  $|S| = \min(|N_0(e)|, q_0(e))$ , where  $N_0(e)$  is the neighborhood around  $e$  in  $G_0$  and  $q_0(e)$  is the capacity of  $e$  in  $G_0$  (which is equivalent to its starting capacity). To ensure  $e$  can do this after a reserved maximal  $b$ -matching in  $G_1$ , we must reduce the quota of  $e$  in  $G_1$  to  $q_1 = q_0(e) - \min(|N_0(e)|, q_0(e))$  and ensure that at least  $\min(|N_0(e)|, q_0(e))$  of its neighbors in  $N_0(e)$  have at least one capacity remaining via a reservation on  $N_0(e)$ . Therefore, when we run the reserved maximal  $b$ -matching on  $G_1$ , we use affiliations  $\mathcal{S}_1 = \{N_0(e), e \in E\}$  with reservations  $r(N_0(e)) = \min(|N_0(e)|, q_0(e))$ .

The algorithm thus works as follows: run a reserved maximal  $b$ -matching on  $G_1$  and then proceed with the standard `PriorityMatch` process on  $G_0, G_2$ , and  $G_3$ . For more details, see the pseudocode of Algorithm 1 in Appendix D.4. It is not hard to see that the four resulting matchings are disjoint. We can show in our proofs that `SmartPriorityMatch` is in fact an intelligent implementation of `PriorityMatch`.

**Lemma 3.** *`SmartPriorityMatch`’s output will always be equivalent to that of `PriorityMatch` with a specific maximal matching function.*

See Appendix D.2 for a proof. Finally, we can show that `SmartPriorityMatch` satisfies the conditions posed in Lemma 2. This concludes Theorem 1. We briefly note that this algorithm solves the problem for any weight  $\lambda \in [0, 1]$ ,<sup>2</sup> however the algorithm itself does not depend on  $\lambda$ . Thus, there must exist a matching that is stable for all  $\lambda$ . We conjecture that increasing the value of  $\lambda$  simply makes stability more difficult to achieve (i.e., for  $1 \geq \lambda > \lambda' \geq 0$ , a stable solution for  $\lambda$  is also stable for  $\lambda'$ ).

## 5 Scalability Experiments

This section provides experimental validation for the polynomial-time scalability of `SmartPriorityMatch`, as analyzed in Theorem 1. To the best of our knowledge, our model is new, so there is no direct benchmark from the literature. Because of this, following the path of others (e.g., recently Cooper and Manlove [2020]), we instead model our problem as an integer linear program (ILP) and compare against that baseline. As in Cooper and Manlove [2020] and other works, we also use that ILP as a “safety check” to ensure

<sup>2</sup>We additionally note that with a slight modification to edge priority, our algorithm could work for  $\lambda \in [0, \infty)$ .

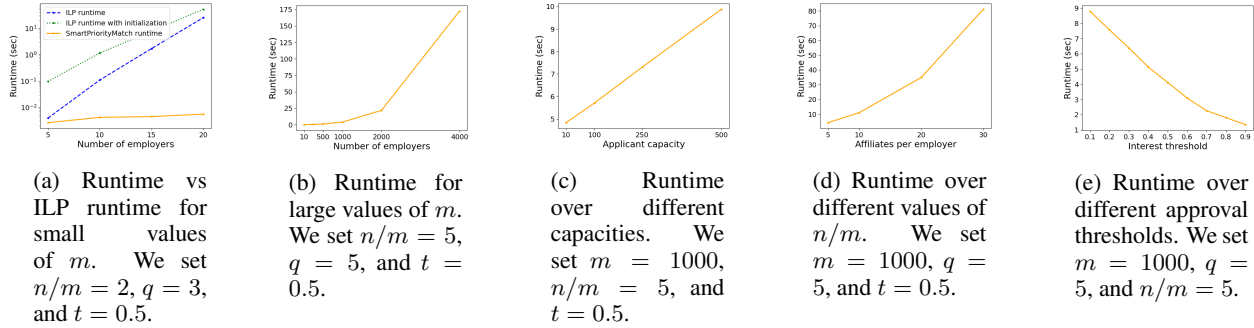


Figure 2: Runtime of SmartPriorityMatch while varying: number of employers ( $m$ ), the capacities of the applicants ( $q$ ), and the number of affiliates per employer ( $n/m$ ). Note that the number of applicants is  $m$  and the capacity of the employers is  $q \cdot n/m$ .

that our algorithmic approach and a general mathematical-programming-based solution method align in their results. The formulation of the ILP and its proof can be found in Appendix E. It translates the set of potential blocking tuples (i.e., our blocking tuple search space) into ILP constraints. Since there are  $O(n^3 \cdot m^3)$  potential blocking tuples on  $n$  applicants and  $m$  employers, the ILP has  $O(n^3 \cdot m^3)$  constraints.

To confirm the efficiency of SmartPriorityMatch, we compare it to the baseline ILP described in Appendix E. We use the same runtime experiments used by Tziavelis et al. [2019] adapted to the DASM setting.<sup>3</sup> We have four parameters: (1)  $m$ , the number of employers, (2)  $n/m$ , the number of affiliates per employer, (3)  $q$ , the capacity for each applicant, and (4)  $t \in (0, 1)$ , a threshold parameter. This means that the number of applicants is  $n$ , and we let employer capacity be  $q \cdot n/m$ . We use Tziavelis et al. [2019]’s Uniform data, where we find a uniform random total ranking for each agent and we use the threshold parameter such that an agent with ranking  $r$  is approved if  $r > t \cdot n$ . In other words, for each agent, we assign a preference of 1 to the top  $100t\%$  of its uniformly randomly ranked preferences. In Figure 2, we run 50 trials for each setting and take the average runtime.

We then vary  $m$  from 5 to 20 and compare the performances of SmartPriorityMatch and the ILP (Figure 2a) with  $n/m = 2$ ,  $q = 3$ , and  $t = 0.5$  fixed. Since the ILP requires  $O(n^3 \cdot m^3)$  constraints, its runtime is very large for even small  $n$ . Due to our system’s space constraints, we were only able to go up to  $n = 20$ . We plotted the performance of the ILP with and without the time to initialize the ILP. We see that SmartPriorityMatch exhibits much better performance, particularly when we include the time to initialize the ILP itself.

Next we plot SmartPriorityMatch’s performance on larger sets, varying parameters one at a time. With  $m$  from 10 to 4000 (Figure 2b), we further support its scalability over the ILP. Varying  $q$  from 5 to 500 (Figure 2c), we see SmartPriorityMatch is dependent on capacity, but in practical ranges, it has less of an impact than varying  $m$ . With  $n/m$  from 5 to 30 (Figure 2d), we see that increasing  $n/m$  has a significant impact on runtime. Finally, varying  $t$  from 0.10 to 0.90, smaller

thresholds appear to increase the runtime (i.e., when agents have a lower bar for expressing interest in other agents).

## 6 Conclusions & Future Research

We propose a new model, the DASM Problem, that characterizes Dooley and Dickerson [2020]’s affiliate matching problem under dichotomous preferences. Dichotomous, or approval-based, preferences are often more realistic for preference elicitation and their application to this model allows for stronger theoretical results. To rank matchings, we use a weighted function that computes agent matching valuations based off their and their affiliates’ preferences. In a human survey, we support the real-world value use of the valuation function with different weights. We then develop (and prove) a quadratic time algorithm to solve the DASM Problem, experimentally validating its efficiency against a baseline ILP.

This work could be extended by considering more general valuation functions, particularly by giving employers more freedom over the relative value of their and their affiliates’ matches. We may draw intuition from recent “same-class” preference extensions to the stable marriage problem such as the work of Kamiyama [2020] or from stable matching work with constraints [Kawase and Iwasaki, 2020]. Similarly, we should consider concerns of fairness (other than stability). Fair stable matching has a long history [Feder, 1995; McDermid and Irving, 2014], with hardness results [Gupta et al., 2019] for various forms of matching (e.g., with incomplete preferences [Cooper and Manlove, 2020] or other fairness constraints such as median-ranked assignment [Sethuraman et al., 2006], equitable matching [Tziavelis et al., 2019], procedural fairness [Tziavelis et al., 2020], etc.), many of which could be applied to the DASM setting.

## References

[Ashlagi and Roth, 2021] Itai Ashlagi and Alvin E Roth. Kidney exchange: an operations perspective. *Management Science*, 2021.

[Aziz, 2020] Haris Aziz. Strategyproof multi-item exchange under single-minded dichotomous preferences. *AAMAS*, 2020.

<sup>3</sup>That work addresses the traditional stable marriage problem and is thus not directly comparable to ours, but we adapt their experimental setup to our setting.



- [Baccara *et al.*, 2012] Mariagiovanna Baccara, Ayşe İmrohoroğlu, Alistair J Wilson, and Leeat Yariv. A field study on matching with network externalities. *AER*, 2012.
- [Bando, 2012] Keisuke Bando. Many-to-one matching markets with externalities among firms. *Journal of Mathematical Economics*, 2012.
- [Bando, 2014] Keisuke Bando. A modified deferred acceptance algorithm for many-to-one matching markets with externalities among firms. *Journal of Mathematical Economics*, 2014.
- [Bogomolnaia and Moulin, 2004] Anna Bogomolnaia and Hervé Moulin. Random matching under dichotomous preferences. *Econometrica*, 2004.
- [Brânzei *et al.*, 2013] Simina Brânzei, Tomasz Michalak, Talal Rahwan, Kate Larson, and Nicholas R Jennings. Matchings with externalities and attitudes. In *AAMAS*, 2013.
- [Cooper and Manlove, 2020] Frances Cooper and David Manlove. Algorithms for New Types of Fair Stable Matchings. In *SEA*, 2020.
- [Dickerson *et al.*, 2019] John Dickerson, Karthik Sankararaman, Kanthi Sarpatwar, Aravind Srinivasan, Kung-Lu Wu, and Pan Xu. Online resource allocation with matching constraints. In *AAMAS*, 2019.
- [Dooley and Dickerson, 2020] Samuel Dooley and John P Dickerson. The affiliate matching problem: On labor markets where firms are also interested in the placement of previous workers. *arXiv preprint arXiv:2009.11867*, 2020.
- [Echenique and Yenmez, 2007] Federico Echenique and M Bumin Yenmez. A solution to matching with preferences over colleagues. *GEB*, 2007.
- [Feder, 1995] Tomás Feder. *Stable networks and product graphs*, volume 555. AMS, 1995.
- [Gale and Shapley, 1962] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 1962.
- [Gupta *et al.*, 2019] Sushmita Gupta, Sanjukta Roy, Saket Saurabh, and Meirav Zehavi. Balanced stable marriage: How close is close enough? In *WADS*. Springer, 2019.
- [Hafalir, 2008] Isa E Hafalir. Stability of marriage with externalities. *IJGT*, 2008.
- [Jones and Teytelboym, 2018] Will Jones and Alexander Teytelboym. The local refugee match: Aligning refugees' preferences with the capacities and priorities of localities. *Journal of Refugee Studies*, 2018.
- [Kamiyama, 2020] Naoyuki Kamiyama. On stable matchings with pairwise preferences and matroid constraints. In *AAMAS*, 2020.
- [Kawase and Iwasaki, 2020] Yasushi Kawase and Atsushi Iwasaki. Approximately stable matchings with general constraints. In *AAMAS*, 2020.
- [Kurokawa *et al.*, 2018] David Kurokawa, Ariel D Procaccia, and Nisarg Shah. Leximin allocations in the real world. *ACM TEAC*, 2018.
- [Li *et al.*, 2014] Jian Li, Yicheng Liu, Lingxiao Huang, and Pingzhong Tang. Egalitarian pairwise kidney exchange: fast algorithms via linear programming and parametric flow. In *AAMAS*, 2014.
- [Manlove *et al.*, 2022] David F. Manlove, Duncan Milne, and Sofiat Olaosebikan. Student-project allocation with preferences over projects: Algorithmic and experimental results. *Discret. Appl. Math.*, 2022.
- [Manlove, 2013] David Manlove. *Algorithmics of matching under preferences*. World Scientific, 2013.
- [McDermid and Irving, 2014] Eric McDermid and Robert W Irving. Sex-equal stable matchings: Complexity and exact algorithms. *Algorithmica*, 2014.
- [Mumcu and Saglam, 2010] Ayşe Mumcu and Ismail Saglam. Stable one-to-one matchings with externalities. *Mathematical Social Sciences*, 2010.
- [Ortega, 2020] Josué Ortega. Multi-unit assignment under dichotomous preferences. *Mathematical Social Sciences*, 2020.
- [Perrault *et al.*, 2016] Andrew Perrault, Joanna Drummond, and Fahiem Bacchus. Strategy-proofness in the stable matching problem with couples. In *AAMAS*, 2016.
- [Pycia, 2012] Marek Pycia. Stability and preference alignment in matching and coalition formation. *Econometrica*, 2012.
- [Rastegari *et al.*, 2016] Baharak Rastegari, Paul Goldberg, and David F. Manlove. Preference elicitation in matching markets via interviews: A study of offline benchmarks (extended abstract). In *AAMAS*, 2016.
- [Roth, 2002] Alvin E Roth. The economist as engineer: Game theory, experimentation, and computation as tools for design economics. *Econometrica*, 2002.
- [Roth, 2018] Alvin E Roth. Marketplaces, markets, and market design. *AER*, 2018.
- [Sandholm and Boutilier, 2006] Tuomas Sandholm and Craig Boutilier. Preference elicitation in combinatorial auctions. *Combinatorial Auctions*, 2006.
- [Sasaki and Toda, 1996] Hiroo Sasaki and Manabu Toda. Two-sided matching problems with externalities. *J. Econ. Theory*, 1996.
- [Sethuraman *et al.*, 2006] Jay Sethuraman, Chung-Piaw Teo, and Liwen Qian. Many-to-one stable matching: geometry and fairness. *Math. Oper. Res.*, 2006.
- [Shen *et al.*, 2020] Weiran Shen, Pingzhong Tang, Xun Wang, Yadong Xu, and Xiwang Yang. Learning to design coupons in online advertising markets. In *AAMAS*, 2020.
- [Tziavelis *et al.*, 2019] Nikolaos Tziavelis, Ioannis Gianakopoulos, Katerina Doka, Nectarios Koziris, and Panagiotis Karras. Equitable stable matchings in quadratic time. In *NeurIPS*, 2019.
- [Tziavelis *et al.*, 2020] Nikolaos Tziavelis, Ioannis Gianakopoulos, Rune Quist Johansen, Katerina Doka, Nectarios Koziris, and Panagiotis Karras. Fair procedures for fair stable marriage outcomes. In *AAAI*, 2020.

## Appendix

In the Appendix, we provide additional problem motivation, proofs, pseudocode, and survey details that were omitted in the body of the paper.

### A Problem Motivation (§1)

In our introduction, we mention multiple motivating examples for our model. In this section, we discuss them and their application to the model in further details.

**Academic Faculty Interview Market** Our main motivating example is the academic faculty *interview* marketplace. Here, graduating students applying for faculty positions are affiliated with their alma mater. Both students and universities indicate which agents on the other side of the market they are interested in interviewing with along with an interview slot capacity. Additionally, for each affiliate-university pair, the university indicates which universities they would like their affiliate to interview with, either for their own prestige or the well-being of the student. Note that this does not place a restriction on a student’s matches based on their university’s interest, but rather models another factor that may influence a university’s preference over a complete interview matching.

The interview market, as opposed to the hiring market which motivated the model proposed by Dooley and Dickerson [2020], is more appropriate in this setting for two main reasons. First, dichotomous preferences seem more appropriate for interview matching, as interviews are intended to gauge interest on both sides, and thus the preference profile need not be refined. In faculty hiring, on the other hand, a preference profile maybe be inherently more complex than binary approval/disapproval. Second, the interview market lends itself to many-to-many matches, as both students and universities may desire multiple interviews, whereas the hiring market only generalizes to many-to-one matches, as students are only hired by one university. This simply better expresses the power of our solution which addresses the general many-to-many setting.

**Playdate Matching** Another application is playdate matching. Consider a group of parents  $\mathcal{P}$ , each with an associated child, denoted by the set  $\mathcal{C}$ . Obviously, the affiliations are defined by parent-child relations, i.e.,  $\text{aff}(p)$  is the child of  $p \in \mathcal{P}$ . These also denote either side of the market. A match between a parent  $p \in \mathcal{P}$  and a child  $c \in \mathcal{C}$  indicates that the child  $c$  will go to  $p$ ’s house to have a playdate with child  $\text{aff}(p)$ . This can be a many-to-many matching, where quotas are how many playdates a child would like to go on and how many children a parent would like to host. Children can dis/approve of parents according to their interest in having a playdate with that child and/or going to their house, and parents dis/approve of children based off their interest in hosting the child and their own child’s interest in the playdate. Additionally, parents’ preferences over their child matches may come from whether or not the parent can drive a child to another parents’ house or other related reasons. Instability indicates that a parent and child would forgo playdates to form a new playdate. Thus this is a nice application for the DASM Problem.

**Study Abroad** In the study abroad matching problem, we have a two-sided market consisting of current students interested in studying abroad and universities. Students are affiliated with the schools they attend, and they express interest or non-interest in other schools to go to study abroad. Schools express approval or non-approval for students that attend their program, as well as approval or non-approval of what study abroad programs they prefer to offer their own students. This also can be nicely modeled in terms of the DASM Problem, where a stable solution ensures a university would never alter their accepted students in favor of other willing students in order to improve their valuation of the entire matching.

**Student Project Allocation** Our next motivating example is the student project allocation market, where we use a quite similar (yet not identical) process to that of Manlove et al. [Manlove *et al.*, 2022]. In this problem, there is a set of students  $\mathcal{S}$ , a set of lecturers  $\mathcal{L}$ , and a set of projects  $\mathcal{P}$ . Projects are proposed by lecturers, which defines a natural affiliation. Students express a preference over projects and lecturers express a preference over students for their affiliated projects. Note that we use approval-based preferences, whereas Manlove et al. use a combination of approval-based and ranked preferences. We believe it is reasonable in this application to elicit entirely dichotomous preferences.

To model this in the DASM Problem, let the sides of the market be  $\mathcal{P}$  and  $\mathcal{S} \cap \mathcal{L}$  respectively. Note that we put  $\mathcal{P}$  on the first side of the market because is the side that will be affiliated with agents on the other side of the market. Let  $\text{aff}(s) = \emptyset$  for all  $s \in \mathcal{S}$  and  $\text{aff}(l) = P_l$  where  $P_l$  is the set of proposed projects by  $l \in \mathcal{L}$ . This forms a disjoint cover over  $\mathcal{P}$ . As projects do not have preferences over students, we simply set project preferences to approve of all students. The quota of a project is the number of students that may work on the project. Similarly, since faculty are not assigned to projects (this is a slight deviation from Manlove et al., where we assume lecturers are automatically assigned to all proposed projects), they must have zero quota. However, they exhibit preferences over the matches of their proposed projects. Finally, students also exhibit preferences over their matches with projects, and may have varying quotas depending on how many projects they are allowed to match with.

In this example, however, we note that stability is not entirely relevant. As lecturers are the only individuals with affiliations, and they have no quota, affiliations will actually not impact the stability of a matching. However, it does impact the overall value of a matching. Therefore, it may be interesting to explore other concepts of fairness in this model in light of this application.

**Dog Breeding** Dog breeding is another problem that can be modeled using the DASM Problem. In the dog breeding market, dog breeders have male and female dogs they would like to breed. In this application, we make the light simplifying assumption that the breeder who owns the female dog receives the offspring and the breeder who owns the male dog sells their services. To that end, the two sides of our market are as follows: in the first side, we have the male dogs, and on the second side, we have breeders, which encapsulates all female dogs they own. Clearly, the male dogs from a breeder are affiliated with that breeder.



Since male dogs do not have preferences, we simply assume male dogs approve of all possible matches (though they have some realistic capacity for matches). Breeders express their interest in male dogs they would like to purchase the services of (i.e., interest in their own matches) based off of the perceived breeding potential. They also express their preferences over breeders they would like their male dogs to service (i.e., interest in their affiliates' matches) based off of offered money, distance, etc.

Like the last example, stability in this model of dog breeding is not entirely compelling as dogs do not have agency to cheat as we describe in this model. However, as before, other notions of fairness may be of interest with respect to this application.

## B Model Definition Proofs (§2)

Here we prove the three propositions presented in Section 2. We start with Proposition 1, which shows that a swapped matching of a matching with respect to a potential blocking tuple is still a matching. The proof is short and direct.

*Proof of Proposition 1.* We know  $\mu'$  is a valid matching if no edge is matched across twice and no capacities are exceeded. The only formed matches are:  $(a, e)$  and possibly  $(a', e'')$  and  $(a'', e')$ . We know  $(a, e)$  is unique as we require  $e \notin \mu(a)$ . Additionally,  $e''$  and  $a''$  are, by definition, not in  $\mu(a')$  and  $\mu(e')$  respectively. Therefore, since  $\mu$  could not have duplicated matches, neither could  $\mu'$ . Both  $a$  and  $e$  lose and gain a match, and both  $a'$  and  $e'$  lose a match and possibly gain one match. Thus, their match sizes could not have increased, so they must not exceed their capacities. Finally,  $a''$  and  $e''$  might gain a match. This only happens if they are in  $N_\mu^A$  and  $N_\mu^E$  respectively, meaning they did not meet their quotas in  $\mu$ . Thus they could not exceed their quotas either.  $\square$

Next, in Proposition 2, we show that we only have to consider potential blocking tuples in order to determine if a matching is stable. Furthermore, we can equate stability with the non-existence of a potential blocking tuple with specific preference profiles.

For intuition about why we can ignore some blocking tuples, we briefly show that there is a limit on the effect cheaters can have on the rest of the matching. For instance, assume  $a \in A$  and  $e \in E$  would like to cheat. In this instance, they can only break off matches to one  $e' \in \mu(a)$  and  $a' \in \mu(e)$  respectively. Thus, as a result of the cheating, only  $e'$  and  $a'$  could have new unused capacity. Then  $e'$  and  $a'$  may decide to match with each other, or they may decide to match with other individuals  $a'' \in A \setminus \mu(e')$  and  $e'' \in E \setminus \mu(a')$  with unmatched capacity respectively. For any other agents involved in the blocking tuple  $\mathcal{T}$ , their ability to match with each other is not a *result* of  $a$  and  $e$  cheating.

Consider, for instance, some  $a^* \in A \cap \mathcal{T} \setminus \{a, a', a''\}$  and  $e^* \in E \cap \mathcal{T} \setminus \{e, e', e''\}$  that are in the tuple but not the aforementioned six affected agents. If  $(a^*, e^*)$  is formed during the second step of the process from Definition 3, then that's simply because the two had additional capacity and preferred to match with each other. This is a valid notion of instability, however, it can be more simply captured by the tuple  $(a^*, \gamma, \gamma, e^*, \gamma, \gamma)$  (recall that  $\gamma$  is the empty agent with

$\mathcal{E} = \{\gamma\}$ ), where all that happens is that  $a^*$  and  $e^*$  form a match.

*Proof of Proposition 2.* Consider a matching  $\mu$  for an instance of the DASM Problem and let  $\mathcal{T} = (a_1, \dots, a_k)$  be the blocking tuple for  $\mu$ . We show that if  $\mathcal{T}$  is not a potential blocking tuple, then it implies that there is some tuple  $\mathcal{T}_1$  with size  $|\mathcal{T}_1| < \mathcal{T}$  is also a blocking tuple for  $\mu$ . This would then prove the first part of our results, that  $\mu$  is unstable if and only if there exists a blocking tuple that is a potential blocking tuple.

Since  $\mathcal{T}$  is not a potential blocking tuple, there is at least one agent  $a^* \in \mathcal{T}$  who is not a cheater, is not cheated on, and does not match with an agent that is a cheater or cheated on. To see why, we consider multiple cases. First, if there are no cheaters, obviously no agents are cheaters or are cheated on, and therefore some agent in  $\mathcal{T}$  must satisfy this.

Second, if there is one cheater, say without loss of generality the cheater is  $a \in A$  who cheats on  $e' \in E$ , assume that all matches have at least one agent who is a cheater or who is cheated on. Since there is only one cheater ( $a$ ) and one who is cheated on ( $e'$ ), there can only be at most two such matches:  $(a, e)$  for some  $e \in E$  and  $(e', a'')$  for some  $a'' \in A$ . If  $(a, e)$  is a match, then since it must be new to be involved in the blocking tuple, then  $e \in E \setminus \mu(a)$ . Similarly, if  $(e', a'')$  is a match,  $a'' \in A \setminus \mu(E)$ . Additionally, since  $a''$  was not involved in cheating, it can only match with  $e'$  if it had unmatched quota. Therefore  $a'' \in N_\mu^A \setminus \mu(e')$ . There can be no other agents in  $\mathcal{T}$  who form matches. Thus either there is some agent  $a^* \in \mathcal{T}$  who does not match, and thus  $a^*$  can be removed and we still have a smaller blocking tuple  $\mathcal{T}_1$  (i.e.,  $a^*$  does not affect the final matching  $\mu''$  since it does nothing), or  $\mathcal{T} = (a, a'', e, e')$ . In the latter case,  $\mathcal{T}$  satisfies the conditions for a potential blocking tuple with  $a', e'' \in \mathcal{E}$ , which is a contradiction. Note that the argument is very similar if the cheater is  $e \in E$  who cheats on some  $a' \in A$ .

Finally, we consider when there are two cheaters. By a similar argument as before, the only matches that involve a cheater or one who is cheated on are the match  $(a, e)$  (which is required to be made in this case) for cheaters  $a \in A$  and  $e \in E \setminus \mu(a)$ , the match  $(a', e'')$  where  $a' \in \mu(e)$  was cheated on and  $e'' \in E \setminus \mu(a')$ , and the match  $(a'', e')$  where  $e' \in \mu(a)$  was cheated on and  $a'' \in A \setminus \mu(e')$ . Additionally, for  $e''$  and  $a''$  to be able to match with  $a'$  and  $e'$  respectively, they must have had unmatched quota after cheating occurred. Therefore, they were either cheated on or had unmatched quota to start, meaning  $e'' \in (N_\mu^E \cup \{e'\}) \setminus \mu(e')$  and  $a'' \in (N_\mu^A \cup \{e'\}) \setminus \mu(a')$ . Since  $\mathcal{T}$  is not a potential blocking tuple, there must be some other agent  $a^* \in \mathcal{T} \setminus \{a, a', a'', e, e', e''\}$ . Otherwise, we use the same argument before to show we can create a smaller tuple  $\mathcal{T}_1$  that is a blocking tuple for  $\mu$ . This concludes the first part of the proof.

At this point, we can assume  $\mathcal{T}$  is a potential blocking tuple that is also a blocking tuple. Consider  $\mu'$ , the swapped matching of  $\mu$  with respect to  $\mathcal{T}$ . Note that in the processed described in Definition 3,  $\mu'$  is equivalent to the final matching. It is not hard to check that each of the six preference conditions in Proposition 2 correspond to the final conditions for the blocking tuple from Definition 3. For instance,  $\mu \prec_a \mu'$  and  $\mu \prec_e \mu'$  simply means  $a$  and  $e$  prefer the final matching

to the starting matching. When  $a'$  is an agent (i.e.,  $a \notin \mathcal{E}$ ),  $\mu \prec_{a'} \mu \cup \{(a', e'')\}$  means  $a'$  prefers to match with  $e''$  over not in the context of the final matching. This is necessarily true by Definition 3, and the same argument holds for  $a''$ ,  $e'$ , and  $e''$ . This concludes the forward direction of the proof. The reverse direction of the proof follows by simply observing that a blocking tuple with these preferences is necessarily a blocking tuple, so if one exists, then  $\mu$  is clearly unstable.  $\square$

Finally, in the reserved maximal  $b$ -matching problem, it is fairly straightforward to show that the proposed Greedy algorithm achieves a maximal matching in  $O(|E|)$  time on edge set  $E$ .

*Proof of Proposition 3.* Let  $G = (V, E, q, S, r)$  be an instance of the reserved maximal  $b$ -matching problem and let  $\mu$  be the matching returned by Greedy. At the start of Greedy, we clearly have a reserved  $b$ -matching. For every edge  $e$ , Greedy only adds  $e$  if it does not break that the matching is a reserved  $b$ -matching. Thus it must always be a reserved  $b$ -matching throughout the algorithm.

Now we show  $\mu$  is maximal. Consider some edge  $e = (u, v) \notin \mu$ , and let  $\mu_e$  be the matching at the time  $e$  was considered. Since  $e$  was not added, its addition to  $\mu_e$  would make it no longer a reserved  $b$ -matching. Therefore, it must break a quota or reservation constraint. Say it breaks a quota constraint for  $v$  (without loss of generality). Then  $|\mu_e(v)| = q(v)$  since  $\mu_e$  is a reserved  $b$ -matching but adding  $e$  would break  $v$ 's quota. Since edges are only added throughout Greedy,  $|\mu(v)| \geq |\mu_e(v)| = q(v)$ , and since  $\mu$  is a reserved  $b$ -matching, it must be that  $|\mu(v)| = q(v)$ . Therefore, adding  $e$  to  $\mu$  would make it no longer a reserved  $b$ -matching.

Otherwise, adding  $e$  to  $\mu_e$  would have broken a reservation constraint for some  $S \in \mathcal{S}$ . Adding  $e$  to  $\mu_e$ , then, must fill the quota of at least one of  $S$ 's vertices. It clearly then must do this for one of its endpoints. If this happens to only one endpoint  $v \in S$  (without loss of generality), then adding  $e$  to  $\mu_e$  makes  $v$  meet its quota. Therefore  $|\mu_e(v)| = q(v) - 1$ . As before,  $|\mu(v)| \geq |\mu_e(v)| = q(v) - 1$ , so  $|\mu(v)| \in \{q(v), q(v) - 1\}$ . Additionally, since  $S$ 's reservation was broken by adding  $e$  which only affected  $v$ 's quota in  $S$ , then  $|S| - r(S)$  vertices in  $S' \subseteq S \setminus \{v\}$  (with  $|S'| = |S| - r(S)$ ) must have their quotas met in  $\mu_e$ . Since edges aren't removed, this is true in  $\mu$  as well. Since  $v \notin S'$ ,  $v$  can't have met its quota in  $\mu$ , else  $\mu$  would violate  $S$ 's reservation. Thus  $|\mu(v)| = q(v) - 1$ . Adding  $e$  to  $\mu$ , then, would make  $v$  meet its quota, thus breaking the reservation for  $S$ . Thus,  $e$  could not be added to  $\mu$ . In the final case, we consider if both  $u$  and  $v$  meet their quotas by adding  $e$  to  $\mu_e$ . The analysis is essentially the same. This concludes the proof.  $\square$

## C Survey Methods and Results (§3)

In this section, we provide real-world motivation for our valuation function in the DASM Problem. To do this, we conducted an online survey that presented the DASM Problem problems to participants. We find that the DASM Problem induces behavior that shows an employer (in our survey, a university) may be willing to trade the quality of their match

for that of their affiliate. We also find motivation for two polar versions of our valuation function: when  $\lambda = 1$  and  $\lambda = \epsilon$ .

Our survey protocol can be found in the Appendix C and follows the work of Dooley and Dickerson [2020]. We developed the survey protocol to answer our main research question: Do real-world participants follow the weighted valuation function and for which weights?

We hypothesize that individuals do follow our function, as we believe it is one of the most rational models for human behavior in the DASM setting. To explore this, we compute how often participants admit our tested weighted valuation function, considering both when  $\lambda = 1$  and  $\lambda = \epsilon$ , when presented with a specific DASM setting.

We conducted the survey through a crowdsourcing platform, Cint, which connected us with English-speaking participants located in the United States. Our institution's IRB reviewed our survey structure and data-collection methods and determined it exempt and did not necessitate an IRB approval. After screening for setting comprehension, 203 participants completed the survey. Ten of those completed the survey too quickly (less than five minutes) and we excluded those responses. Each of the remaining 193 responses are included in our analysis below and we paid Cint \$3.05 for their time. The median response time of these 193 responses was 23 minutes.

### C.1 Survey Design

The participants were first introduced to the standard matching problem with three agents on each side of the market. They were primed to identify themselves as one of the universities. There was a matching-related test designed to filter out participants who were not paying attention to the content of the survey. The test consisted of three text-based questions in which the participant matched the text to a visual depiction of the corresponding matching scenario based off a dichotomous preference profile. Next, the concept of affiliates was introduced in the same matching setting with the five possible matches that involved their university and their affiliate. We then randomly primed the participant to believe that it was in their best interest to prioritize the match of their affiliate<sup>4</sup>. Finally, we presented the participant with eight dichotomous scenarios and asked them to express their full preference over the possible matchings. These eight scenarios are fully detailed in the protocol in Appendix C.

### C.2 Results

The main survey results are depicted in Table 1. To test if participants agreed with the valuation function, we first calculated the probability of uniform random responses resulting in a preference profile that follows the function for both parameters. We then compared this to the observed probability and found that with  $p = 0.05$  on a right-tailed alternative hypothesis of a binomial test that the increase in observed adherence to the valuation function for both parameterizations is significant.

Additionally, of all participants, the median number of scenarios where they completely adhered to the  $\lambda = 1$  valuation

<sup>4</sup>Random priming, while not a main focus of our experiment, tests how differences in beliefs about one's own gain from the match of an affiliate would lead to different matching strategies

function is 2. The same is true for the  $\lambda = \epsilon$  valuation function. When  $\lambda = 1$ , the median values between the primed and unprimed groups individually are 3 and 2 respectively. When  $\lambda = \epsilon$ , the medians are 2 and 1 respectively.

Furthermore, 157 of the participants used the valuation function with  $\lambda = \epsilon$  at least once and 164 used  $\lambda = 1$  at least once (excluding scenario 5). These results indicate that participants generally adhered to both valuation functions, but did not do so consistently throughout the survey. However, since their adherence was significantly higher than random, this suggests that there is structure in how the participants chose to adhere to the valuation functions. We pose for future research to design a survey instrument which investigates why a participant would choose whether or not to adhere to a particular valuation function.

We also observe that there is light evidence that our priming method was effective in inducing participants to follow the valuation function. Since this was not one of our central research questions with the survey (*does our priming method induce more deference to the affiliates' match*) we only mention that with further work, we could explore exactly how to prime the participants better. The purpose in performing the priming was to simulate the behavior of an admission faculty member. While this population is challenging to survey, our survey instrument does suggest that when considering the self-interest in your affiliate candidate's match, an agent may be more likely to follow either of the valuation function parameterizations.

### C.3 Complete Protocol

This section includes the entire protocol used for the survey in Section 3

#### Faculty Hiring Program

The design of this survey is aimed at understanding how you make decisions with different competing priorities. You will be exploring this concept in the setting of a hiring market for new faculty professors. The Survey will have two parts: (1) familiarization with faculty hiring, and (2) answering questions about your preferences. We begin with the familiarization part now.

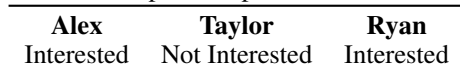
Consider a hiring market such as this one with three applicants (Ryan, Alex, and Taylor), and three universities (Bear Mountain, Littlewood, and West Shores).



Imagine that **you are Bear Mountain University**, and you performed an evaluation of the applicants. You decided that you liked Alex more than you liked Taylor, and you liked Taylor more than you liked Ryan.

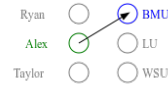
Top Tier Candidates	Middle Tier Candidates	Bottom Tier Candidates
Alex	Taylor	Ryan

You could depict that preference as:

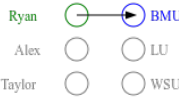


Your preference over your possible student matches could then look like:

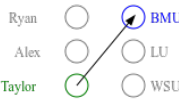
#### First Choice



#### Second Choice

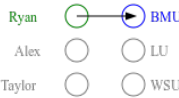


#### Third Choice

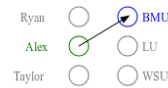


But since you like Ryan too, you could also have the preferences:

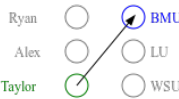
#### First Choice



#### Second Choice



#### Third Choice



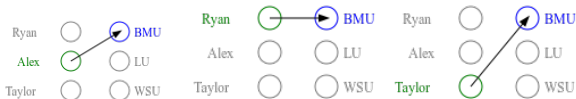
To test your comprehension of the previous setting, can you now do the matchings yourself? These are intuitive and should be easy to complete.

Assume **you are Bear Mountain University**. Assume this time that after you review the applicants, you are interested in being matched with the candidates as follows:

Alex	Taylor	Ryan
Interested	Not Interested	Interested

Then what is your ranking of the following options?

Assuming you believe the above, rank these outcomes from your most preferred (1) to your least preferred (3).

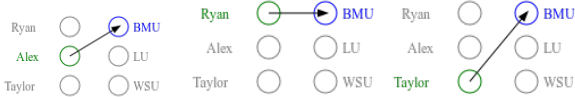


Assume **you are Bear Mountain University**. Assume this time that after you review the applicants, you place them in these tiers:

Alex	Taylor	Ryan
Not Interested	Not Interested	Interested

Then what is your ranking of the following options?

Assuming you believe the above, rank these outcomes from your most preferred (1) to your least preferred (3).

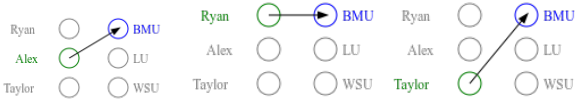


Assume you are Bear Mountain University. Assume this time that after you review the applicants, you place them in these tiers:

Alex	Taylor	Ryan
Not Interested	Interested	Not Interested

Then what is your ranking of the following options?

Assuming you believe the above, rank these outcomes from your most preferred (1) to your least preferred (3).

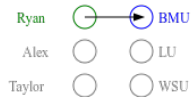


Awesome! Now onto the second part of the Survey. Since you understand the basic faculty hiring setting, let us introduce another layer of complexity.

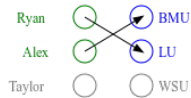
In faculty hiring, the applicants are affiliated with a university based off where they earned their PhD. What this means is that universities also care about where their student gets a job.

Assume that you are Bear Mountain University and your student is Ryan. You then have the following five options of matchings:

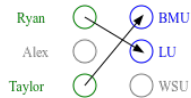
You are matched with Ryan.



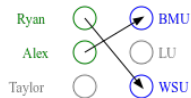
You are matched with Alex; Ryan is matched with Littlewood University.



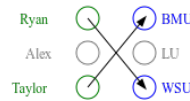
You are matched with Taylor; Ryan is matched with Littlewood University.



You are matched with Alex; Ryan is matched with West Shores University.



You are matched with Taylor; Ryan is matched with West Shores University.



In the remainder of the survey, we will ask you to express your preferences over these five options under different settings of which applicants you like best and what you think of the different universities.

Randomly assigned either of the two following prompts:

For the remainder of the survey, you will get to decide how you would like to balance your own interest in being matched with the best candidates possible, and where Ryan should be matched. There is no right answer – it is up to you about how you balance these two things.

or

In faculty hiring, it is common to want your affiliate to be placed at a good university. This is often because you as a university will be perceived as a better university if your students get jobs at top tier schools.

So, keep in mind if Ryan is placed at a university you are interested in, then your university will be viewed better and could hire better candidates in the future. While you want Ryan to be matched with a good school, you must balance this priority with your competing priority that you want a good candidate. There is no right answer – it is up to you about how you balance these two things.

We will ask you to express your preferences to 8 scenarios. When you are ready, please continue to Part 2.

Display this question 8 times with the interests as expressed in the enumerated list in Section 3.

Assume that you are Bear Mountain University and Ryan is your student.

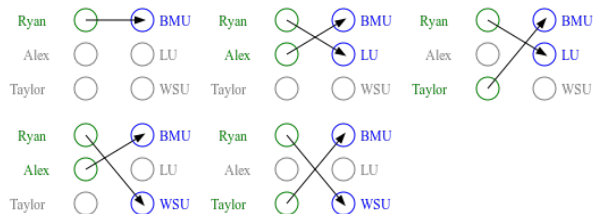
Assume that you have evaluated the candidates and you decide that you are interested in hiring Alex, but you are not interested in hiring Ryan and Taylor.

Alex	Taylor	Ryan (your student)
Interested	Not Interested	Not Interested

Assume that you have the following beliefs about the schools, based off of the school's ranking. You are interested in matching Ryan with Littlewood (LU) and your school (BMU), but you are not interested in Ryan being matched with West Shores (WSU).

LU	WSU	BMU (your university)
Interested	Not Interested	Interested

Assuming you believe the above, rank these outcomes from your most preferred (1) to your least preferred (5).



Can you describe the procedure you used to rank these? Why did you use this procedure?

## D Main Algorithm Proofs and Pseudocode (§4)

In this section, we present the proofs for our theoretical work. This includes proofs regarding the reserved maximal  $b$ -matching problem, `PriorityMatch` and `SmartPriorityMatch`. At the end, we present the pseudocode for the algorithm.

### D.1 The Reserved Maximal $b$ -Matching Problem

We start by proposing a simple and efficient algorithm for solving the reserved maximal  $b$ -matching problem. This shows that `SmartPriorityMatch` can be implemented efficiently.

Let `Greedy` be a greedy algorithm for the reserved maximal  $b$ -matching problem, where we consider edges in an arbitrary order and greedily add them to the matching as long as they don't break any reservation or capacity constraints.

**Proposition 3.** *`Greedy` solves reserved maximal  $b$ -matching in  $O(|E|)$  time for  $E$  edges.*

*Proof.* Let  $G = (V, E, q, \mathcal{S}, r)$  be an instance of the reserved maximal  $b$ -matching problem and let  $\mu$  be the matching returned by `Greedy`. At the start of `Greedy`, we clearly have a reserved  $b$ -matching. For every edge  $e$ , `Greedy` only adds  $e$  if it does not break that the matching is a reserved  $b$ -matching. Thus it must always be a reserved  $b$ -matching throughout the algorithm.

Now we show  $\mu$  is maximal. Consider some edge  $e = (u, v) \notin \mu$ , and let  $\mu_e$  be the matching at the time  $e$  was considered. Since  $e$  was not added, its addition to  $\mu_e$  would make it no longer a reserved  $b$ -matching. Therefore, it must break a quota or reservation constraint. Say it breaks a quota constraint for  $v$  (without loss of generality). Then  $|\mu_e(v)| = q(v)$  since  $\mu_e$  is a reserved  $b$ -matching but adding  $e$  would break  $v$ 's quota. Since edges are only added throughout `Greedy`,  $|\mu(v)| \geq |\mu_e(v)| = q(v)$ , and since  $\mu$  is a reserved  $b$ -matching, it must be that  $|\mu(v)| = q(v)$ . Therefore, adding  $e$  to  $\mu$  would make it no longer a reserved  $b$ -matching.

Otherwise, adding  $e$  to  $\mu_e$  would have broken a reservation constraint for some  $S \in \mathcal{S}$ . Adding  $e$  to  $\mu_e$ , then, must fill the quota of at least one of  $S$ 's vertices. It clearly then must do this for one of its endpoints. If this happens to only one endpoint  $v \in S$  (without loss of generality), then adding  $e$  to  $\mu_e$  makes  $v$  meet its quota. Therefore  $|\mu_e(v)| = q(v) - 1$ . As before,  $|\mu(v)| \geq |\mu_e(v)| = q(v) - 1$ , so  $|\mu(v)| \in \{q(v), q(v) - 1\}$ . Additionally, since  $S$ 's reservation was broken by adding  $e$  which only affected  $v$ 's quota in  $S$ , then  $|S| - r(S)$  vertices in  $S' \subseteq S \setminus \{v\}$  (with  $|S'| = |S| - r(S)$ ) must have their quotas met in  $\mu_e$ . Since edges aren't removed, this is true in  $\mu$  as well. Since  $v \notin S'$ ,  $v$  can't have met its quota in  $\mu$ , else  $\mu$  would violate  $S$ 's reservation. Thus  $|\mu(v)| = q(v) - 1$ . Adding  $e$  to  $\mu$ , then, would make  $v$  meet its quota, thus breaking the reservation for  $S$ . Thus,  $e$  could not be added to  $\mu$ . In the final case, we consider if both  $u$  and  $v$  meet their quotas by adding  $e$  to  $\mu_e$ . The analysis is essentially the same. This concludes the proof.  $\square$

## D.2 PriorityMatch Proofs

In this section, we address all proofs regarding `PriorityMatch`. First, we introduce a new lemma that will simply show a useful property that we use throughout these proof. The high level idea is that if matching  $\mu'$  is a swapped matching of  $\mu$ , agents only care about the part of their matches (and possibly their affiliate's matches) that change to decide which matching they like better. Specifically, we give conditions for an  $a \in A$  and  $e \in E$  where  $a \notin \mu(e)$  but  $a \in \mu'(e)$  would *not* strongly prefer the new match to the old one. If this can be shown for all possible ways to do a pairwise swap to match  $a$  and  $e$ , then they cannot form a blocking tuple. This is crucial to our proof.

**Lemma 4.** *Consider the DASM Problem. Let  $\mathcal{T} = (a, a', a'', e, e', e'')$  be a potential blocking tuple for a matching  $\mu$ , and  $\mu'$  be the swapped matching of  $\mu$  with respect to  $\mathcal{T}$ . Then both of the following hold:*

1. *If  $\text{pr}_a(e) \leq \text{pr}_a(e')$ , then  $a$  cannot prefer  $\mu'$  to  $\mu$ .*
2. *Let  $I_a, I_{a'}$ , and  $I_{a''}$  be the respective indicators that  $a, a'$ , and  $a''$  are in  $\text{aff}(e)$ . Let  $J_{a'', a'}$  be the indicator that  $a'' \neq a'$ . If*

$$\begin{aligned} & \text{pr}_e^e(a') + \lambda I_a \text{pr}_e^a(e') + \lambda I_{a'} \text{pr}_e^{a'}(e) \\ & \geq \text{pr}_e^e(a) + \lambda I_a \text{pr}_e^a(e) + \lambda I_{a'} \text{pr}_e^{a'}(e'') \\ & \quad + \lambda I_{a''} J_{a'', a'} \text{pr}_e^{a''}(e'), \end{aligned}$$

*then  $e$  cannot prefer  $\mu'$  to  $\mu$ .*

*Proof.* For 1:

$$\begin{aligned} \text{val}_a(\mu') &= \text{pr}_a(e) + \sum_{e^* \in \mu'(a) \setminus e} \text{pr}_a(e^*) \\ &\leq \text{pr}_a(e') + \sum_{e^* \in \mu(a) \setminus e'} \text{pr}_a(e^*) \quad (1) \\ &= \text{val}_a(\mu), \end{aligned}$$

where line 1 comes from the fact that the two summations sum over the same matches of  $a$ , and  $\text{pr}_a(e) \leq \text{pr}_a(e')$  (as given). Thus,  $a$  does not strongly prefer  $\mu'$  to  $\mu$ . This completes the first part of the Lemma. We will now show 2, for

$b_1, \dots, b_{|\text{aff}(a)|} \in \text{aff}(a)$ :

$$\begin{aligned}
\text{val}_e(\mu') &= \text{pr}_e^e(a) + \sum_{a^* \in \mu'(e) \setminus \{a\}} \text{pr}_e^e(a^*) \\
&\quad + \lambda I_a \left( \text{pr}_e^a(e) + \sum_{e^* \in \mu'(a) \setminus \{e\}} \text{pr}_e^a(e^*) \right) \\
&\quad + \lambda I_{a'} \left( \text{pr}_e^{a'}(e'') + \sum_{e^* \in \mu'(a') \setminus \{e''\}} \text{pr}_e^{a'}(e^*) \right) \\
&\quad + \lambda I_{a''} J_{a'', a'} \left( \text{pr}_e^{a''}(e') + \sum_{e^* \in \mu'(a'') \setminus \{e'\}} \text{pr}_e^{a''}(e^*) \right) \\
&\quad + \lambda \sum_{a^* \in \text{aff}(e) \setminus \{a, a', a''\}} \sum_{e^* \in \mu'(a^*)} \text{pr}_e^{a^*}(e^*) \\
&\leq \text{pr}_e^e(a') + \sum_{a^* \in \mu(e) \setminus \{a'\}} \text{pr}_e^e(a^*) \\
&\quad + \lambda I_a \left( \text{pr}_e^a(e') + \sum_{e^* \in \mu(a) \setminus \{e'\}} \text{pr}_e^a(e^*) \right) \\
&\quad + \lambda I_{a'} \left( \text{pr}_e^{a'}(e) + \sum_{e^* \in \mu(a') \setminus \{e\}} \text{pr}_e^{a'}(e^*) \right) \\
&\quad + \lambda I_{a''} J_{a'', a'} \left( \sum_{e^* \in \mu(a'')} \text{pr}_e^{a'}(e^*) \right) \\
&\quad + \lambda \sum_{a^* \in \text{aff}(e) \setminus \{a, a', a''\}} \sum_{e^* \in \mu(a^*)} \text{pr}_e^{a^*}(e^*) \quad (2) \\
&= \text{val}_e(\mu)
\end{aligned}$$

where line 2 comes from the fact that all the summations are over the same matches, and otherwise the terms we pull out compose the provided inequality. When we pull out the summation for  $a''$ , note that this is only something we can pull out if  $a'' \neq a'$ , because otherwise we already pulled it out as the  $a'$  summation. This is why we multiply it by  $J_{a'', a'}$ .  $\square$

We now introduce another useful lemma that implies that in any blocking tuple with respect to the matching found by PriorityMatch,  $a$  cannot be matched. Equivalently:

**Lemma 5.** *Let  $\mu$  be the resulting matching from PriorityMatch. Then any potential blocking tuple  $\mathcal{T} = (a, a', a'', e, e', e'')$  that blocks  $\mu$  must satisfy  $e', a'' \in \mathcal{E}$  and  $a', e'' \notin \mathcal{E}$ .*

*Proof.* Fix a blocking tuple  $\mathcal{T} = (a, a', a'', e, e', e'')$ . Let  $\mu'$  be the swapped matching of  $\mu$  with respect to  $\mathcal{T}$ . Assume for contradiction that  $e' \notin \mathcal{E}$ . That means  $e' \in \mu(a)$  by the definition of the blocking tuple. Notice that every edge  $(a^*, e^*)$  in every subgraph in PriorityMatch satisfies  $\text{pr}_{a^*}^{e^*}(e^*) = 1$ . Since all matches are selected from these edge and  $(a, e')$  was matched by PriorityMatch, this implies  $\text{pr}_a^e(e') = 1$ . By Lemma 4,  $a$  cannot prefer  $\mu'$  to  $\mu$ . This contradicts that  $\mathcal{T}$  is

a blocking tuple. This proves  $e' \in \mathcal{E}$ . By the definition of the blocking tuple, we then know  $a'' \in \mathcal{E}$ .

Next, consider the case when  $a' \in \mathcal{E}$  or  $e'' \in \mathcal{E}$ . By the definition of the potential blocking tuple, one implies the other, thus  $a', e'' \in \mathcal{E}$ . This means  $a$  and  $e$  both are not matched to capacity and they simply use this to match with each other. Since each must have an interest in each other to form a blocking tuple, (i.e.,  $\text{pr}_a^e(e) = 1$  and  $\text{pr}_e^e(a) = 1$  or  $\text{pr}_e^a(e) = 1$ ), that means  $(a, e)$  is an edge in one of the PriorityMatch graphs. Since they never reached capacity, they must have then matched during the maximal matching in that graph. This contradicts that  $e \notin \mu(a)$ .  $\square$

Combining these two lemmas yields the following useful lemma:

**Lemma 6.** *Consider the DASM Problem. Let  $\mathcal{T} = (a, a', a'', e, e', e'')$  be a potential blocking tuple for a matching  $\mu$ , and  $\mu'$  be the swapped matching of  $\mu$  with respect to  $\mathcal{T}$ . Let  $I_a$  and  $I_{a'}$  be the respective indicators that  $a$  and  $a'$  are in  $\text{aff}(e)$ . If*

$$\text{pr}_e^e(a') + \lambda I_{a'} \text{pr}_e^{a'}(e) \geq \text{pr}_e^e(a) + \lambda I_a \text{pr}_e^a(e) + \lambda I_{a'} \text{pr}_e^{a'}(e''),$$

then  $e$  cannot prefer  $\mu'$  to  $\mu$ .

*Proof.* Let  $I_{a''}$  be the indicator that  $a'' \in \text{aff}(a)$  and  $J_{a'', a'}$  be the indicator that  $a'' \neq a'$ . By Lemma 5,  $I_{a''} = 0$  and  $\text{pr}_e^e(e') = 0$  since  $e', a'' \in \mathcal{E}$ . The two following equations hold:

$$\begin{aligned}
&\text{pr}_e^e(a') + \lambda I_{a'} \text{pr}_e^{a'}(e) \\
&= \text{pr}_e^e(a') + \lambda I_a \text{pr}_e^a(e') + \lambda I_{a'} \text{pr}_e^{a'}(e), \\
&\text{pr}_e^e(a) + \lambda I_a \text{pr}_e^a(e) + \lambda I_{a'} \text{pr}_e^{a'}(e'') \\
&= \text{pr}_e^e(a) + \lambda I_a \text{pr}_e^a(e) + \lambda I_{a'} \text{pr}_e^{a'}(e'') + \lambda I_{a''} J_{a'', a'} \text{pr}_e^{a''}(e').
\end{aligned}$$

Substituting these into each side for the second inequality in Lemma 4 gives the desired result.  $\square$

Note that this is a more useful version of the second inequality of Lemma 4. Since we will need a version of both inequalities of Lemma 4, we will now canonically refer to Lemma 4 to refer to its first inequality, and Lemma 6 for the simpler version of the second inequality. We now prove the lemma defining when PriorityMatch works.

*Proof of Lemma 2.* Obviously, as the quotas always reflect the maximum remaining space any agent has for matches, this creates a valid matching. We now show this is stable.

Consider  $\mu$ , the output of PriorityMatch. Assume that it satisfies the preconditions of this lemma: for any potential blocking tuple  $\mathcal{T} = (a, a', a'', e, e', e'')$  such that  $a, a' \in \text{aff}(e)$  and  $a$  prefers the swapped matching of  $\mu$  with respect to  $\mathcal{T}$ , then:

$$\text{pr}_e^e(a') + \lambda \text{pr}_e^{a'}(e) \geq \text{pr}_e^e(a) + \lambda \text{pr}_e^a(e) + \lambda \text{pr}_e^{a'}(e''). \quad (1)$$

By Lemma 5, we can ignore the  $e'$  and  $a''$  parameters as  $e', a'' \in \mathcal{E}$ , so they are not involved in the blocking tuple. Thus our tuple is effectively reduced to  $\mathcal{T} = (a, a', e, e'')$ .



Assume for contradiction that  $\mathcal{T}$  is a blocking tuple. Let  $\mu'$  be the swapped matching of  $\mu$  with respect to  $\mathcal{T}$ . Note that to use Inequality 1, it is sufficient to show that  $a, a' \in \text{aff}(e)$  and  $\mu' \succ_a \mu$  (i.e., the precondition of the lemma). The latter is satisfied because we assume  $\mathcal{T}$  is a blocking tuple. Additionally, if we were to satisfy this, then  $I_a = I_{a'} = 1$ , which means  $\text{pr}_e^e(a') + \lambda \text{pr}_e^{a'}(e) = \text{pr}_e^e(a') + \lambda I_{a'} \text{pr}_e^{a'}(e)$  and  $\text{pr}_e^e(a) + \lambda \text{pr}_e^a(e) + \lambda \text{pr}_e^{a'}(e'') = \text{pr}_e^e(a) + \lambda I_a \text{pr}_e^a(e) + \lambda I_{a'} \text{pr}_e^{a'}(e'')$ . Plugging these into Inequality 1, we get:

$$\text{pr}_e^e(a') + \lambda I_{a'} \text{pr}_e^{a'}(e) \geq \text{pr}_e^e(a) + \lambda I_a \text{pr}_e^a(e) + \lambda I_{a'} \text{pr}_e^{a'}(e''),$$

This satisfies Lemma 6, which implies  $e$  cannot prefer  $\mu'$  to  $\mu$ . Therefore,  $a$  and  $a'$  cannot both be in  $\text{aff}(e)$  simultaneously.

Given that  $e', a'' \in \mathcal{E}$  and  $a', e'' \notin \mathcal{E}$  by Lemma 5, it must be the case, then, that  $a \in N_\mu^A$  is simply filling unmet capacity while  $e$  has met its capacity and is therefore dropping its match with some  $a' \in A$  in order to match with  $a$ . Then  $a'$  may or may not match with another employer (depending on if  $e'' \in \mathcal{E}$  or  $e'' \in E$ ). We can then do a case by case analysis based off the fact that  $a$  and  $a'$  cannot both be in  $\text{aff}(e)$  (note that we say  $\mu_i$  is the maximal matching found by PriorityMatch in  $G_i$ ):

1. If  $a \in \text{aff}(e)$ : Then  $I_a = 1$  and  $a' \notin \text{aff}(e)$  so  $I_{a'} = 0$ . Since  $a' \notin \text{aff}(e)$ ,  $(a', e)$  can only have been in  $G_1$ . Since it was matched by PriorityMatch, it must have been in  $G_1$ . Therefore,  $\text{pr}_e^e(a') = 1$ . Thus, on the lefthand side of Lemma 6:

$$\text{pr}_e^e(a') + \lambda I_{a'} \text{pr}_e^{a'}(e) = 1$$

On the righthand side:

$$\text{pr}_e^e(a) + \lambda I_a \text{pr}_e^a(e) + \lambda I_{a'} \text{pr}_e^{a'}(e'') = \text{pr}_e^e(a) + \lambda \text{pr}_e^a(e)$$

Since  $\mathcal{T}$  is a blocking tuple by assumption, then by Lemma 6 we must have  $\text{pr}_e^e(a) + \lambda \text{pr}_e^a(e) > 1$ . This can only happen if  $\text{pr}_e^e(a) = \text{pr}_e^a(e) = 1$ . Since Lemma 4 implies  $\text{pr}_a(e) = 1$ ,  $(e, a)$  must have been an edge in  $G_0$ . Note also that since  $a' \notin \text{aff}(e)$ ,  $(e, a') \notin G_0$ . Thus  $a$  did not meet its quota in  $G_0$  (it never did) and  $e$  must have had remaining quota after  $\mu_0$ . This contradicts the maximality of  $\mu_0$ .

2. Else if  $a' \in \text{aff}(e)$ : Then  $I_{a'} = 1$  and  $a \notin \text{aff}(e)$  so  $I_a = 0$ . We start by simplifying the left and righthand side of Lemma 6:

$$\begin{aligned} \text{pr}_e^e(a') + \lambda I_{a'} \text{pr}_e^{a'}(e) &= \text{pr}_e^e(a') + \lambda \text{pr}_e^{a'}(e) \\ \text{pr}_e^e(a) + \lambda I_a \text{pr}_e^a(e) + \lambda I_{a'} \text{pr}_e^{a'}(e'') &= \text{pr}_e^e(a) + \lambda \text{pr}_e^{a'}(e'') \end{aligned}$$

For  $\mathcal{T}$  to be a blocking tuple, Lemma 6 implies:

$$\text{pr}_e^e(a') + \lambda \text{pr}_e^{a'}(e) < \text{pr}_e^e(a) + \lambda \text{pr}_e^{a'}(e'')$$

The lefthand side can be 0,  $\lambda$ , 1, or  $1 + \lambda$ . It obviously cannot be 0, else  $\text{pr}_e^e(a') = \text{pr}_e^{a'}(e) = 0$ , so  $(e, a')$  could

not have been an edge in any graph, and therefore could not have been selected by PriorityMatch. It also cannot be  $1 + \lambda$ , since this is the maximum value of the righthand side.

Consider if  $\text{pr}_e^e(a') + \lambda \text{pr}_e^{a'}(e) \in \{1, \lambda\}$ . Then  $\text{pr}_e^{a'}(e) = 0$  and  $\text{pr}_e^e(a') = 1$  or the reverse, so  $(e, a')$  was an edge in  $G_2$  or  $G_3$  respectively, implying that  $e$  had unmatched quota until at least either  $\mu_2$  or  $\mu_3$  occurred respectively (i.e., it had unmatched quota in  $G_2$  or  $G_3$ ). Since  $a$  always had unmatched quota during PriorityMatch, it cannot have appeared in any graph prior to  $G_2$  or  $G_3$  respectively, else PriorityMatch would have matched  $e$  and  $a$ . In the first case, which occurs when the lefthand side is 1, we know it cannot be the case that  $\text{pr}_e^e(a) = \text{pr}_e^{a'}(e) = 1$  (since that would put  $(e, a')$  in  $G_0$ ), thus the righthand side is bound above by 1 so the inequality cannot hold. In the second case, which occurs when the lefthand side is  $\lambda$ , we know  $\text{pr}_e^e(a) = 0$  (otherwise  $(e, a')$  would be in  $G_0$  or  $G_2$ ), thus the righthand side is bound above by  $\lambda$  so the inequality cannot hold. In either case, this is a contradiction.

3. Else:  $a, a' \notin \text{aff}(e)$ , so  $I_a = I_{a'} = 0$ . Again, we simplify the left and righthand side of Lemma 6:

$$\begin{aligned} \text{pr}_e^e(a') + \lambda I_{a'} \text{pr}_e^{a'}(e) &= \text{pr}_e^e(a') \\ \text{pr}_e^e(a) + \lambda I_a \text{pr}_e^a(e) + \lambda I_{a'} \text{pr}_e^{a'}(e'') &= \text{pr}_e^e(a) \end{aligned}$$

Since PriorityMatch matched  $(e, a')$ ,  $(e, a')$  must have been in some graph. For this to happen when  $a' \notin \text{aff}(e)$ , it must be the case that  $\text{pr}_e^e(a') = 1$ . Thus the lefthand side becomes 1, which is also an upper bound on the righthand side. Thus Lemma 6 holds, contradicting that  $\mathcal{T}$  is a potential blocking tuple.

Thus we have contradicted the possibility of having a blocking tuple, so our algorithm produces a stable marriage.  $\square$

Finally, we prove the weakness of PriorityMatch in the prioritized setting.

*Proof of Lemma 1.* Consider a market with two agents on each side:  $A = \{a_1, a_2\}$  and  $E = \{e_1, e_2\}$ . We let  $\text{aff}(e_1) = \{a_1, a_2\}$  and  $\text{aff}(e_2) = \emptyset$ . All possible preferences are set to one *except*  $\text{pr}_{e_1}^{a_2}(e_2) = 0$ ,  $\text{pr}_{e_2}^{e_2}(a_2) = 0$ , and  $\text{pr}_{a_2}(e_2) = 0$ . All quotas are 1. This means that  $G_0$  contains edges  $(a_1, e_1)$  and  $(a_2, e_1)$ . Since  $e_1$  has a quota of 1, it must select only one edge. In an arbitrary maximal matching, it might select  $(a_1, e_1)$ . All other graphs, at this point, will either be empty, or only have edges with at least one 0-quota endpoint. Thus  $a_2$  and  $e_2$  will not be matched (though, we could imagine just arbitrarily matching the rest at the end and our argument will hold). However, note that  $a_2$  likes  $e_1$  and dislikes  $e_2$ . Similarly,  $e_1$  would prefer for its affiliate  $a_1$  to be matched with  $e_2$  than  $a_2$  be matched with  $e_2$ . Clearly,  $a_2$  and  $e_1$  would prefer to be matched and allow  $a_1$  and  $e_2$  to be matched.  $\square$

### D.3 SmartPriorityMatch Proofs

Next, we must show that SmartPriorityMatch is effectively an implementation of PriorityMatch with a smarter algorithm

for the matchings. In order to do this, we need to show that each matching could have been generated by `PriorityMatch` given the prior matches. We show this one step at a time. Note there are slight differences between the graphs for `PriorityMatch` and `SmartPriorityMatch` in the quotas of the graphs and reservations. To clarify, for instance, the quotas in the first graph for each algorithm, we use  $q_0$  for `PriorityMatch` and  $q'_0$  for `SmartPriorityMatch`.

**Lemma 7.** *Let  $G_0 = (V_0, E_0, q_0)$  be the first subgraph considered by `PriorityMatch` with capacities  $q_0$ . Let  $\mu_0$  be the matching `SmartPriorityMatch` finds on  $G_0$ . Then  $\mu_0$  is a maximal matching on  $G_0$ .*

*Proof.* Let  $G_1 = (V_1, E_1, q_1)$  be the second subgraph considered by `PriorityMatch`. Before `SmartPriorityMatch` finds  $\mu_0$ , it runs a reserved maximal matching  $\mu_1$  on  $(V_1, E_1, q'_1, \mathcal{S}, r)$ , where  $q'_1$  are the capacities used by `SmartPriorityMatch` on  $V_1$ ,  $\mathcal{S}$  are the affiliations, and  $r$  are the reservations. Next, it finds some maximal matching  $\mu_0$  on  $(V_0, E_0, q'_0)$ , where  $q'_0$  are the capacities used by `SmartPriorityMatch` on  $V_0$ . Since we use the same edge set and  $q_0(v)' = q_0(v) - |\mu_1(v)| \leq q_0(v)$  for each  $v \in V$ ,  $\mu_0$  is a valid matching on  $G_0$ .

Assume for contradiction there is some  $(e, a) \in E_0 \setminus \mu_0$  such that  $\mu_0 \cup \{(e, a)\}$  is a valid matching in  $G_0$ . Consider the topology of  $(V_0, E_0)$ . It is a set of disjoint stars connecting each employer (the center) to a subset of its affiliates. In  $G_0$ , we then know  $e$  is going to match to some subset of its neighborhood. Since the stars are all disjoint and all star leaves have capacity at least one, then in a maximal matching,  $e$  can and must match to any  $\min(|N_0(e)|, q_0(e))$ -sized subset of its neighbors during `PriorityMatch`.

Since  $\mu_0 \cup \{(e, a)\}$  is a valid matching in  $G_0$ , that means there is some set of matches  $S$  that involve  $e$  with  $(e, a) \in S$  such that  $\mu_0(e) \cup S$  is maximal (i.e., you cannot add any more matches to  $e$  without breaking some agent's quota). By our argument from before,  $|\mu_0(e) \cup S| = \min(|N_0(e)|, q_0(e))$ , therefore  $|\mu_0(e)| < \min(|N_0(e)|, q_0(e))$ . Since the structure of graph  $(V_0, E_0, q'_0)$  is the same as  $(V_0, E_0, q_0)$ ,  $\mu_0(e)$  must abide by the same properties to be maximal, notably that  $|\mu_0| = \min(|N'_0(e)|, q'_0(e))$  where  $N'_0(e) = \{a \in N_0(e) : q'_0(a) > 0\}$ . Therefore,  $\min(|N_0(e)|, q_0(e)) > \min(|N'_0(e)|, q'_0(e))$ .

We start by considering  $|N'_0(e)|$ . It is equivalent to the number of agents  $a \in N_0(e)$  who did not match to their quota in  $\mu_1$ . Since  $\mu_1$  is a reserved maximal b-matching with  $N_0(e) \in \mathcal{S}$  with reservation  $r(N_0(e)) = \min(|N_0(e)|, q_0(e))$ , this number is at least  $\min(|N_0(e)|, q_0(e))$ . Therefore,  $|N'_0(e)| \geq \min(|N_0(e)|, q_0(e))$ .

Thus for  $\min(|N_0(e)|, q_0(e)) > \min(|N'_0(e)|, q'_0(e))$  to hold, it must be the case that  $q'_0(e) < \min(|N_0(e)|, q_0(e))$ . Note that we set  $q'_0(e) = q_0(e) - |\mu_1(e)|$ . Thus  $q_0(e) - |\mu_1(e)| < \min(|N_0(e)|, q_0(e))$ , and so  $q_0(e) < \min(|N_0(e)|, q_0(e)) + |\mu_1(e)|$ . Additionally, we know that we set  $q'_1(e) = q_0(e) - \min(|N_0(e)|, q_0(e))$ , and since  $|\mu_1(e)| \leq q'_1(e)$ , we finally get that  $q_0(e) \leq q_0(e)$ , which is a contradiction.

Therefore, by contradiction,  $\mu_0$  is a maximal matching on  $G_0$ .  $\square$

Given this, we now show that the second matching  $\mu_1$  could be equivalent between `PriorityMatch` and `SmartPriorityMatch`.

**Lemma 8.** *Let  $G_0 = (V_0, E_0, q_0)$  and  $G_1 = (V_1, E_1, q_1)$  be the first and second subgraphs considered by `PriorityMatch` with capacities  $q_0$  and  $q_1$ . Assume an implementation of `PriorityMatch` and `SmartPriorityMatch` result in the same matching on  $G_0$ , call it  $\mu_0$ . If  $\mu_1$  is the matching found by `SmartPriorityMatch` on  $G_1$ , then  $\mu_1$  is a maximal matching on  $G_1$ .*

*Proof.* Consider the same notation as introduced in Lemma 7 and fix some  $e \in E$ . Recall in Lemma 7 we showed that  $|\mu_0(e)| = \min(|N_0(e)|, q_0(e))$ . Therefore,  $e$ 's quota in  $G_1$  for `PriorityMatch` is  $q_1(e) = q_0(e) - |\mu_0(e)| = q_0(e) - \min(|N_0(e)|, q_0(e))$ . In `SmartPriorityMatch`, its quota is  $q'_1(e) = q_0 - \min(|N_0(e)|, q_0(e))$ . Therefore, for all  $e \in E$ ,  $q_1(e) = q'_1(e)$ .

Now we will simply show that it is both a valid matching and it is maximal in  $G_1$ . Assume for contradiction it is not a valid matching in  $G_1$ . That means it must match some  $a \in A$  (as we know all  $e \in E$  has the same capacity in both graphs) above its capacity. Note that  $q'_1(a) = q_0(a)$ , and  $q_1(a) \geq q'_1(a) - 1$ , because since  $a$  could only be the leaf of a star component in  $G_0$ ,  $|\mu_0(a)| \leq 1$ . Therefore,  $a$ 's capacity could only have been exceeded by 1, and this only occurs when  $|\mu_0(a)| = 1$  and `SmartPriorityMatch` constructs  $\mu_1$  such that  $|\mu_1(a)| = q'_0(a)$ . For this to happen, then, `SmartPriorityMatch` first matches  $a$  up to its capacity in  $\mu_1$ . Then  $a$ 's capacity is reduced to 0, so it cannot match  $a$  in  $\mu_0$ . This contradicts that  $|\mu_0(a)| = 1$ . Thus,  $\mu_1$  is a valid matching on  $G_1$ .

Assume for contradiction that  $\mu_1$  is not maximal. This implies there is some  $(e, a) \in G_1 \setminus \mu_1$  where both  $e$  and  $a$  are not matched up to their capacity in  $G_1$ . Note, however, since  $\mu_1$  is maximal according to the reservation, either  $e$  is matched to capacity,  $a$  is matched to capacity, or  $a$  is reserved. If  $e$  is matched to capacity in the reserved matching, then it must also be matched to capacity in  $G_1$  since its capacity is the same in both. This is a contradiction. Otherwise,  $a$  is matched to capacity in the reserved matching or it is reserved. If  $a$  is matched to capacity, notice that since  $q'_1(a) = q_0(a)$ , thus  $|\mu_1(a)| = q_0(a) \geq q_1(a)$ . Thus,  $a$  must be (at least) matched to capacity in  $G_1$ . Finally, we consider when  $a$  is reserved. Let  $e = \text{aff}^{-1}(a)$ . For  $a$  to have remaining capacity and a remaining unmatched neighbor with capacity and be forced to not match, it could have only had 1 remaining capacity ( $q'_1(a) - |\mu_1(a)| = 1$ ) and there must have been exactly  $r(N_0(e)) = \min(|N_0(e)|, q_0(e))$  agents in  $N_0(e)$  that were not matched to capacity by  $\mu_1$ . Since  $|\mu_0(e)| = \min(|N_0(e)|, q_0(e))$  and all other  $a' \in N_0(e)$  must have had  $q'_0(a') = 0$ , these agents must precisely make up  $\mu_0(e)$ . Thus,  $|\mu_0(a)| = 1$ . Therefore,  $|\mu_0(a)| + |\mu_1(a)| = 1 + q'_1(a) - 1 = q'_1(a) = q_0(a)$ . Thus, in `PriorityMatch`,  $a$  has met its capacity in  $G_1$ . This is a contradiction.

Thus by contradiction,  $\mu_1$  is maximal on  $G_1$ .  $\square$

Now we can immediately prove Lemma 3.

*Proof of Lemma 3.* Simply combine Lemmas 7 and 8, and note that after  $\mu_0$  and  $\mu_1$  are found, `SmartPriorityMatch`

acts identically to `PriorityMatch`. This is sufficient to show `SmartPriorityMatch` is a valid implementation of `PriorityMatch`.  $\square$

Now we must prove `SMARTPRIORITYMATCH` exhibits additional properties to `PRIORITYMATCH`. We start by showing it satisfies the preconditions for Lemma 2.

**Lemma 9.** *Let  $\mu$  be the matching obtained by `SmartPriorityMatch`. Consider a potential blocking tuple  $\mathcal{T} = (a, a', a'', e, e', e'')$  such that  $a, a' \in \text{aff}(e)$ . Let  $\mu'$  be the swapped matching of  $\mu$  with respect to  $\mathcal{T}$ . Then if  $a$  prefers  $\mu'$  to  $\mu$ :*

$$\text{pr}_e^e(a') + \lambda \text{pr}_e^a(e') + \lambda \text{pr}_e^{a'}(e) \geq \text{pr}_e^e(a) + \lambda \text{pr}_e^a(e) + \lambda \text{pr}_e^{a'}(e'').$$

*Proof.* For the beginning of the proof, we will be viewing the order of events with respect to `PriorityMatch`. To that end,  $\mu_0$  was formed first, then  $\mu_1$  and the rest.

Fix our tuple and matchings  $\mu$  and  $\mu'$  and assume  $a, a' \in \text{aff}(e)$  and  $a$  prefers  $\mu'$  to  $\mu$ . Since  $a$  prefers  $\mu'$  to  $\mu$ , we know by Lemma 4 that  $\text{pr}_a(e) > \text{pr}_a(e')$ , which means  $\text{pr}_a(e) = 1$  and  $\text{pr}_a(e') = 0$ . Since  $a$  cannot match to something it does not like in `SmartPriorityMatch`, it must be the case that  $e' \in \mathcal{E}$  (and thus  $a'' \in \mathcal{E}$ ), and so  $a$  never matched to its quota.

Since  $a' \in \text{aff}(e)$ , it must exist in the tuple. By the potential blocking tuple definition,  $(a', e) \in \mu$ . Thus it must exist in some graph. Since  $a' \in \text{aff}(e)$ , it must have been  $G_0, G_2$ , or  $G_3$ . If  $(e, a)$  existed in a graph  $G_i$ , since they did not match but  $a$  never reached its capacity, that means  $e$  must have reached its capacity in or before  $\mu_i$ . Thus,  $(a', e)$  must have appeared at latest in graph  $G_i$ . It is not hard to see since  $\lambda \leq 1$  that this implies that:

$$\text{pr}_e^e(a') + \lambda \text{pr}_e^{a'}(e) \geq \text{pr}_e^e(a) + \lambda \text{pr}_e^a(e)$$

Thus, to prove the lemma, it is sufficient to show that  $\text{pr}_e^{a'}(e'') = 0$ . Assume for contradiction it is 1. For this to be true,  $e'' \notin \mathcal{E}$ . Additionally, by the definition of a blocking tuple,  $(a', e'') \notin \mu$ ,  $\text{pr}_{a'}(e'') = 1$ , and (since  $a' \notin \text{aff}(e'')$ ) because it is in  $\text{aff}(e)$   $\text{pr}_{e''}^{a'}(a') = 1$ . Therefore,  $(a', e'')$  appeared in  $G_1$ . For them to not match, that means  $a'$  must not have quota after  $G_1$ , so all its matches must have been in  $G_0$  and  $G_1$ . Therefore  $(a', e)$  must have occurred in  $G_0$ .

We will now view matchings in the order that occurs in `SmartPriorityMatch`, so  $\mu_1$  occurs first, then  $\mu_0$  and the rest. Recall from Lemma 7 that  $|\mu_0(e)| = \min(|N_0(e)|, q_0(e)) = r(N_0(e))$ . Therefore, the set of agents  $\mu_0(e)$  must have not been matched to quota when  $\mu_1$  was made (before  $\mu_0$  was made). Thus,  $\mu_0(e)$  is sufficient to satisfy the reservation on  $N_0(e)$  for the matching  $\mu_1(e)$ . Since  $(a, e)$  was not matched,  $a \notin \mu_0(e)$  even though  $(a, e) \in G_0$ , meaning  $a \in N_0(e)$ . Additionally,  $a$  was never matched to capacity. Therefore, there are at least  $r(N_0(e)) + 1$  agents in  $N_0(e)$  that did not meet their capacity in  $\mu_1$ . Additionally, since  $|\mu_0(a')| = 1$ ,  $|\mu_1(a')| \leq q_0(a') - 1$ . Since  $\mu_1$  occurred first and the reservation constraint had not been met and  $e''$  had capacity (since it never matched to capacity),  $(a', e'')$  would have matched in  $\mu_1$ . This is a contradiction. This concludes our proof.  $\square$

Now we proceed with Theorem 1.

*Proof of Theorem 1.* Lemma 3 shows that `SmartPriorityMatch` is a specific implementation of `PriorityMatch`. Additionally, Lemmas 9 shows that `SmartPriorityMatch` satisfies the preconditions of Lemma 2. Therefore, by Lemma 2, `SmartPriorityMatch` finds a stable matching.  $\square$

#### D.4 SmartPriorityMatch Pseudocode

Here we present the pseudocode for `SmartPriorityMatch`. This can be seen in Algorithm 1.

---

##### Algorithm 1 SmartPriorityMatch

---

**Input:** Sets  $A$  and  $E$  of agents, affiliate function  $\text{aff} : e \rightarrow \mathcal{P}(A)$ , quota function  $q : A \cup E \rightarrow \mathbb{N}$ , preference functions  $\forall a \in A \text{ pr}_a : E \rightarrow \{0, 1\}$ ,  $\forall e \in E \text{ pr}_e^e : A \rightarrow \{0, 1\}$ ,  $\forall e \in E, a \in \text{aff}(e) \text{ pr}_e^a : E \rightarrow \{0, 1\}$ , and a string  $val$  to designate the valuation function

**Output:** A stable matching  $\mu$

- 1:  $V \leftarrow E \cup A$
  - 2:  $E_0 \leftarrow \{(a, e) \in A \times E : a \in \text{aff}(e), \text{pr}_a(e) = \text{pr}_e^e(a) = \text{pr}_e^a(e) = 1\}$
  - 3:  $\forall e \in E, N_0(e) = \{a \in A : (e, a) \in E_0\}$
  - 4:  $E_1 \leftarrow \{(a, e) \in A \times E : a \notin \text{aff}(e), \text{pr}_a(e) = \text{pr}_e^e(a) = 1\}$
  - 5:  $\forall e \in E, q'_1(e) \leftarrow q(e) - \min(|N_0(e)|, q(e))$
  - 6:  $\forall a \in A, q'_1(a) \leftarrow q(a)$
  - 7:  $\mathcal{S} = \{N_0(e) : e \in E\}$
  - 8:  $\forall e \in E \text{ r}(N_0(e)) = \min(|N_0(e)|, q(e))$
  - 9:  $\mu_1 \leftarrow \text{ReservedMaximalMatching}(V, E_1, q'_1, \mathcal{S}, r)$
  - 10:  $\forall a \in E \cup A, q'_0(a) \leftarrow q(a) - |\mu_1(a)|$
  - 11:  $\mu_0 \leftarrow \text{MaximalMatching}(V, E_0, q'_0)$
  - 12:  $E_2 \leftarrow \{(a, e) \in A \times E : \text{pr}_a(e) = 1, \text{pr}_e^e(a) \neq \text{pr}_e^a(e)\}$
  - 13:  $\forall a \in E \cup A, q_2(a) \leftarrow q(a) - |\mu_0(a)| - |\mu_1(a)|$
  - 14:  $\mu_2 \leftarrow \text{MaximalMatching}(A \cup E, E_2, q_2)$
  - 15:  $E_3 \leftarrow \{(a, e) \in A \times E : \text{pr}_a(e) = \text{pr}_e^e(a) = 1, \text{pr}_e^a(a) = 0\}$
  - 16:  $\forall a \in E \cup A, q_3(a) \leftarrow q_2(a) - |\mu_2(a)|$
  - 17:  $\mu_3 \leftarrow \text{MaximalMatching}(A \cup E, E_3, q_3)$
  - 18: **return**  $\mu_0 \cup \mu_1 \cup \mu_2 \cup \mu_3$ .
- 

## E ILP Formulation and Proofs (§5)

In this section, we formulate our problem as an integer linear program (ILP). As this is a rather standard and straightforward solution, and ILP solvers are known to be efficient in practice, this is a good baseline to compare our algorithm to. Let  $z_{e,a}$  for all  $e \in E$  and  $a \in A$  denote  $(e, a)$  is matched if  $z_{e,a} = 1$  and  $(e, a)$  is unmatched if  $z_{e,a} = 0$ . Our basic constraints are as follows:

$$\forall e \in E, a \in A : z_{e,a} \in \{0, 1\} \quad (1)$$

$$\forall e \in E : \sum_{a \in A} z_{e,a} \leq q(e) \quad (2)$$

$$\forall a \in A : \sum_{e \in E} z_{e,a} \leq q(a) \quad (3)$$

These constraints simply ensure all edges are matched or not and the number of matches containing an agent does not ex-

ceed that agent's capacity. Note that this is sufficient to ensure that we have a valid matching. Now we need to consider stability. To do this concisely, we introduce a function  $\text{coeff} : \mathcal{B} \rightarrow \mathbb{N}$ , where  $\mathcal{B}$  is the set of tuples  $\mathcal{T} = (a, a', a'', e, e', e'')$  that satisfy all conditions for being a blocking tuple except those that depend on  $\mu$  (i.e., if we introduce the appropriate  $\mu$ ,  $\mathcal{T}$  is a blocking tuple). Note that our definition of  $\mathcal{B}$  determines the weight selection for our valuation function, as it determines which potential blocking tuples could actually be blocking tuples based off of preferences. In this program, we will use many indicators. To refrain from introducing many new variables, we let  $\mathbb{I}_p$  for some boolean  $p$  be 1 if  $p$  is true, and 0 otherwise. For example,  $\mathbb{I}_{a', e' \in \mathcal{E}}$  is 1 if  $a'$  and  $e'$  are both in  $\mathcal{E}$ , and otherwise it is 0. Then we can define  $\text{coeff}$  as follows:

$$\begin{aligned} \text{coeff}(a, a', a'', e, e', e'') \\ = & q(e)q(a)\mathbb{I}_{a', e' \in \mathcal{E}} + q(e)\mathbb{I}_{a' \in \mathcal{E}, e' \notin \mathcal{E}} + q(a)\mathbb{I}_{a' \notin \mathcal{E}, e' \in \mathcal{E}} \\ & + \mathbb{I}_{a', e' \notin \mathcal{E}, a'', e'' \in \mathcal{E}} + q(e'')q(a'')\mathbb{I}_{a'', e'' \notin \mathcal{E} \cup \{a', e'\}} \\ & + q(e'')\mathbb{I}_{e'' \notin \mathcal{E}, a'' \in \mathcal{E}} + q(a'')\mathbb{I}_{e'' \in \mathcal{E}, a'' \notin \mathcal{E}} \end{aligned}$$

This can also be thought of a conditional, where we return  $q(e)q(a)$ ,  $q(e)$ ,  $q(a)$ , 1,  $q(e'')q(a'')$ ,  $q(e'')$ , or  $q(a'')$  depending on which elements in the tuple are in  $\mathcal{E}$  or not. Note that these are all constants: it does not involve variables from the ILP. Then our constraints to ensure stability are as follows, where:

$$\begin{aligned} \forall \mathcal{T} = (a, a', a'', e, e', e'') \in \mathcal{B} : \\ \text{coeff}(\mathcal{T})z_{e,a} + \mathbb{I}_{a' \notin \mathcal{E}} \text{coeff}(\mathcal{T})(1 - z_{e,a'}) \\ + \mathbb{I}_{e' \notin \mathcal{E}} \text{coeff}(\mathcal{T})(1 - z_{e',a}) + \mathbb{I}_{a'' \notin \mathcal{E}} \text{coeff}(\mathcal{T})z_{e',a''} \\ + \mathbb{I}_{e'' \notin \mathcal{E}} \text{coeff}(\mathcal{T})z_{e'',a'} + \frac{\text{coeff}(\mathcal{T})}{q(a)} \mathbb{I}_{e' \in \mathcal{E}} \sum_{e^* \in E} z_{e^*,a} \\ + \frac{\text{coeff}(\mathcal{T})}{q(e)} \mathbb{I}_{a' \in \mathcal{E}} \sum_{a^* \in A} z_{e,a^*} + \frac{\text{coeff}(\mathcal{T})}{q(a'')} \mathbb{I}_{a'' \notin \mathcal{E}} \sum_{e^* \in E} z_{e^*,a''} \\ + \frac{\text{coeff}(\mathcal{T})}{q(e'')} \mathbb{I}_{e'' \notin \mathcal{E}} \sum_{a^* \in A} z_{e'',a^*} \\ \leq \text{coeff}(\mathcal{T}) \quad (4) \end{aligned}$$

While these constraints may seem construed, they are derived directly from the definition of a blocking tuple, and account for all possible cases of a blocking tuple. Therefore, this is the most direct translation of the DASM Problem into an ILP.

**Theorem 2.** *The ILP defined by (1), (2), (3), and (4) solves the DASM Problem.*

We prove Theorem 2 by breaking down the construction of the ILP.

*Proof of Theorem 2.* We start by constructing the ILP from the ground up, and it will be easy to check (albeit, time-consuming) that this ILP is just a broken down version of the ILP in question. Let  $z_{e,a}$  for all  $e \in E$  and  $a \in A$  denote  $(e, a)$  is matched if  $z_{e,a} = 1$  and  $(e, a)$  is unmatched if  $z_{e,a} = 0$ . Our basic constraints are as follows:

$$\begin{aligned} \forall e \in E, a \in A : z_{e,a} \in \{0, 1\} \\ \forall e \in E : \sum_{a \in A} z_{e,a} \leq q(e) \\ \forall a \in A : \sum_{e \in E} z_{e,a} \leq q(a) \end{aligned}$$

These constraints simply ensure all edges are matched or not, and the number of matches containing an agent does not exceed its capacity. Note that this is sufficient to ensure that we have a valid matching. Next, we must ensure stability. We will consider a number of potential blocking tuples. We break it down into all the different ways matches can be broken down and reformed for a swapped matching.

First: consider when some  $a \in A$  and  $e \in E$  are simply undermatched. Then, without breaking matches, they are allowed to match to each other. They will only do this if they get something out of the match. We must ensure that either they are matched together, or one has reached capacity. For notation, let  $I_a^e$  be 1 if  $a \in \text{aff}(e)$  and 0 otherwise. We can guarantee the result with the following constraint:

$$\begin{aligned} \forall a \in A, e \in E \text{ s.t. } \text{pr}_a(e) = 1 \wedge \text{pr}_e^a(a) + I_a^e \text{pr}_e^a(e) \geq 1 : \\ (q(a) \cdot q(e))z_{e,a} + q(a) \sum_{a^* \in A} z_{e,a^*} + q(e) \sum_{e^* \in E} z_{e^*,a} \\ \geq q(a) \cdot q(e) \end{aligned}$$

Second: consider when  $e$  is undermatched, but  $a$  drops its match with some  $e' \in \mu(a)$ . These three together will only form a blocking tuple if  $a$  prefers  $e$  to  $e'$  and  $e$  gets something out of the match. We must ensure that  $(a, e)$  is matched or  $(a, e')$  is not matched or  $e$  is at capacity:

$$\begin{aligned} \forall a \in A, e \in E, e' \in E \setminus \{e\} \\ \text{s.t. } \text{pr}_a(e) > \text{pr}_a(e') \wedge \text{pr}_e^a(a) + I_a^e \text{pr}_e^a(e) \geq I_a^e \text{pr}_e^a(e') : \\ q(e)z_{e,a} + q(e)(1 - z_{e',a}) + \sum_{a^* \in A} z_{e,a^*} \geq q(e) \end{aligned}$$

Third: consider when  $a$  is undermatched, but  $e$  drops its match with some  $a' \in \mu(e)$ . These three together will only form a blocking tuple if  $e$  prefers a match from  $a$  to  $e$  than  $a'$  to  $e$  and  $a$  gets something out of the match. We must ensure that  $(a, e)$  is matched or  $(a', e)$  is not matched or  $a$  is at capacity:

$$\begin{aligned} \forall a \in A, e \in E, a' \in A \setminus \{a\} \\ \text{s.t. } \text{pr}_a(e) = 1 \\ \wedge \text{pr}_e^a(a) + I_a^e \text{pr}_e^a(e) \geq \text{pr}_e^a(a') + I_{a'}^e \text{pr}_e^a(e') : \\ q(a)z_{e,a} + q(a)(1 - z_{e,a'}) + \sum_{e^* \in E} z_{e^*,a} \geq q(a) \end{aligned}$$

Fourth: consider when  $a$  and  $e$  drop matches  $e'$  and  $a'$  re-

spectively to match with each other, but neither  $e'$  nor  $a'$  decide to rematch. This is only notable when both  $a$  and  $e$  prefer being matched together. In this case, we must ensure  $(a, e)$  is matched or  $(a, e')$  is not matched or  $(a', e)$  is not matched.

$$\forall a \in A, e \in E, a' \in A \setminus \{a\}, e' \in E \setminus \{e\}$$

$$\text{s.t. } \text{pr}_a(e) > \text{pr}_a(e')$$

$$\wedge \text{pr}_e^e(a) + I_a^e \text{pr}_e^a(e) \geq \text{pr}_e^e(a') + I_a^e \text{pr}_e^a(e') + I_{a'}^e \text{pr}_e^{a'}(e) :$$

$$z_{e,a} + (1 - z_{e,a'}) + (1 - z_{e',a}) \geq 1$$

Fifth: consider when  $a$  and  $e$  drop matches  $e'$  and  $a'$  respectively to match with each other, and  $a'$  rematches with some  $e''$  that is undermatched and  $e'$  does not rematch. This is only notable when both  $a$  and  $e$  prefer being matched together and both  $a'$  and  $e''$  gain from being matched together. We need to ensure that  $(a, e)$  is matched,  $(a, e')$  is not matched,  $(a', e)$  is not matched,  $(a', e'')$  is matched, or  $e''$  is matched to capacity:

$$\forall a \in A, e \in E, a' \in A \setminus \{a\}, e' \in E \setminus \{e\}, e'' \in E \setminus \{e, e'\}$$

$$\text{s.t. } \text{pr}_a(e) > \text{pr}_a(e')$$

$$\wedge \text{pr}_e^e(a) + I_a^e \text{pr}_e^a(e) + I_{a'}^e \text{pr}_e^{a'}(e'')$$

$$\geq \text{pr}_e^e(a') + I_a^e \text{pr}_e^a(e') + I_{a'}^e \text{pr}_e^{a'}(e)$$

$$\wedge \text{pr}_{a'}(e'') = 1 \wedge \text{pr}_{e''}^{e''}(a') + I_{a'}^{e''} \text{pr}_{e''}^{a'}(e'') \geq 1 :$$

$$q(e'')z_{e,a} + q(e'')(1 - z_{e,a'}) + q(e'')(1 - z_{e',a})$$

$$+ q(e'')z_{e'',a'} + \sum_{a^* \in A} z_{e'',a^*} \geq q(e'')$$

Sixth: consider when  $a$  and  $e$  drop matches  $e'$  and  $a'$  respectively to match with each other, and  $e'$  rematches with some  $a''$  that is undermatched and  $a'$  does not rematch. This is only notable when both  $a$  and  $e$  prefer being matched together and both  $e'$  and  $a''$  gain from being matched together. We need to ensure that  $(a, e)$  is matched,  $(a, e')$  is not matched,  $(a', e)$  is not matched,  $(a'', e')$  is matched, or  $a''$  is matched to capacity:

$$\forall a \in A, e \in E, a' \in A \setminus \{a\}, e' \in E \setminus \{e\}, a'' \in A \setminus \{a, a'\}$$

$$\text{s.t. } \text{pr}_a(e) > \text{pr}_a(e')$$

$$\wedge \text{pr}_e^e(a) + I_a^e \text{pr}_e^a(e) + I_{a''}^e \text{pr}_e^{a''}(e')$$

$$\geq \text{pr}_e^e(a') + I_a^e \text{pr}_e^a(e') + I_{a'}^e \text{pr}_e^{a'}(e)$$

$$\wedge \text{pr}_{a''}(e') = 1 \wedge \text{pr}_{e'}^{e'}(a'') + I_{a''}^{e'} \text{pr}_{e'}^{a''}(e') \geq 1 :$$

$$q(a'')z_{e,a} + q(a'')(1 - z_{e,a'}) + q(a'')(1 - z_{e',a})$$

$$+ q(a'')z_{e',a''} + \sum_{e^* \in E} z_{e^*,a''} \geq q(a'')$$

Seventh: consider when  $a$  and  $e$  drop matches  $e'$  and  $a'$  respectively to match with each other, and  $a'$  and  $e'$  rematch with some  $e''$  and  $a''$  respectively that are either both undermatched or  $a'' = a'$  and  $e'' = e'$ . This is only notable when

both  $a$  and  $e$  prefer being matched together, both  $a'$  and  $e''$  gain from being matched together, and both  $e'$  and  $a''$  gain from being matched together. We need to ensure that  $(a, e)$  is matched,  $(a, e')$  is not matched,  $(a', e)$  is not matched,  $(a', e'')$  is matched,  $(a'', e')$  is matched, or if  $a'' \neq a'$  and  $e'' \neq e'$ , then either  $a''$  or  $e''$  is matched to capacity (recall  $J_{a'',a'} = 1$  if and only if  $a'' \neq a'$ ):

$$\forall a \in A, e \in E, a' \in A \setminus \{a\}, e' \in E \setminus \{e\}, a'' \in A \setminus \{a, a'\}$$

$$\text{s.t. } \text{pr}_a(e) > \text{pr}_a(e')$$

$$\wedge \text{pr}_e^e(a) + I_a^e \text{pr}_e^a(e) + I_{a''}^e \text{pr}_e^{a''}(e') \geq I_a^e \text{pr}_e^a(e') + I_{a'}^e \text{pr}_e^{a'}(e)$$

$$\wedge \text{pr}_{a'}(e'') = 1 \wedge \text{pr}_{e''}^{e''}(a') + I_{a'}^{e''} \text{pr}_{e''}^{a'}(e'') \geq 1$$

$$\wedge \text{pr}_{a''}(e') = 1 \wedge \text{pr}_{e'}^{e'}(a'') + I_{a''}^{e'} \text{pr}_{e'}^{a''}(e') \geq 1 :$$

$$(q(a'') \cdot q(e''))z_{e,a} + (q(a'') \cdot q(e''))(1 - z_{e,a'})$$

$$+ (q(a'') \cdot q(e''))(1 - z_{e',a}) + (q(a'') \cdot q(e''))z_{e',a''}$$

$$+ (q(a'') \cdot q(e''))z_{e'',a'}$$

$$+ J_{a'',a'} \left( q(e'') \sum_{e^* \in E} z_{e^*,a''} + q(a'') \sum_{a^* \in A} z_{e'',a^*} \right)$$

$$\geq (q(a'') \cdot q(e''))$$

This encodes all the cases for the existence of a blocking tuple. Therefore, solutions to the ILP directly correspond to solutions to the dichotomous affiliate stable matching problem. Note that the ILP of interest is actually equivalent to this, one simply needs to go through each type of blocking tuple and check the inequalities.  $\square$