# 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 faculty 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 $\mathsf{pr}_a : E \to \{0,1\}$ (resp. $\mathsf{pr}_e^e : A \to \{0,1\}$, 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 $\mathsf{pr}_u(v) = 1$ (or $\mathsf{pr}_u^u(v) = 1$ 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 $\mu : A \cup E \to 2^{A \cup E}$ 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 $\mathsf{aff} : E \to 2^A$ 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 $\mathsf{aff}(e_1) = \{a_1\}$, $\mathsf{aff}(e_2) = \{a_2, a_3\}$, and $\mathsf{aff}(e_3) = \emptyset$. Note that 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 $a \in \mathsf{aff}(e)$, $e$ has preferences $\mathsf{pr}_e^a : E \to \{0,1\}$. 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:*

$$\mathsf{val}_e(\mu) = \sum_{a^* \in \mu(e)} \mathsf{pr}_e^e(a^*) + \lambda \sum_{a_i \in \mathsf{aff}(e)} \sum_{e^* \in \mu(a_i)} \mathsf{pr}_e^{a_i}(e^*)$$

$$\mathsf{val}_a(\mu) = \sum_{e^* \in \mu(a)} \mathsf{pr}_a(e^*).$$

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

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. It may be agreed upon by employers or set by a centralized system in order to avoid potential abuse of this parameter. 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. If $\lambda = \epsilon$ for small $\epsilon > 0$, employers care about their matches first and 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: $\mathsf{val}_{e_2}(\mu) = 3$ (since it likes one of its matches and both of its affiliates' matches) and $\mathsf{val}_{e_2}(\mu') = 2$ (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: $\mathsf{val}_{e_2}(\mu) = 1 + 2\epsilon$ and $\mathsf{val}_{e_2}(\mu') = 2$. 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 $a' \in \mu(a)$.*

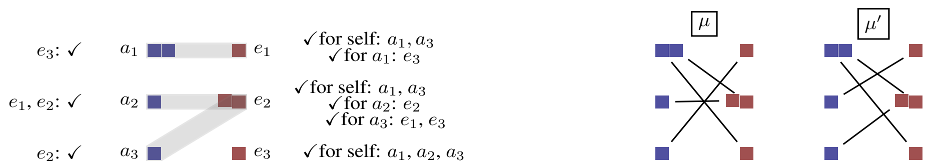


Figure 1: An example matching problem. On the left are affiliations and preferences for each agent. The capacity of each agent is represented with squares. For instance, $e_2$ has $\mathsf{aff}(e_2) = \{a_2, a_3\}$, $q(e_2) = 2$, and it approves of $a_1$ and $a_3$ for itself, $e_2$ for affiliate $a_2$, 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 in the standard stable marriage problem, we would like to ensure that no two agents[1] $a \in A$ 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 $a' \in \mathsf{aff}(e)$, 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 $\mathcal{T} = (a_1, \ldots, a_k)$ 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 match in the final matching. Thus, non-cheating agents are not performing calculated activity; they are simply reacting.

Interestingly, we only need to consider certain blocking tuples of size at most six to determine instability. See Proposition 2 and the full version of our paper 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 $a'$.

[1]In future work, this could be generalized to *coalitions*, or larger sets of agents.

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 our full paper 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 our full paper 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 $\mathcal{T}$. When $\lambda = 1$, 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 and is a blocking tuple. When $\lambda = \epsilon$, since $e_2$ does not prefer $\mu'$ to $\mu$, this is not a blocking tuple.

| | $\lambda = 1$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Scenario | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| *Random* | 10% | 5% | 5% | 3% | 100% | 10% | 10% | 3% |
| *Observed Unprimed* | **41%** | 31% | 30% | **56%** | 100% | 42% | **25%** | 30% |
| *Observed Primed* | **41%** | **44%** | **35%** | 55% | 100% | **51%** | 22% | **45%** |

| | $\lambda = \epsilon$ | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Scenario | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| *Random* | 3% | 1% | 1% | 3% | 20% | 3% | 3% | 1% |
| *Observed Unprimed* | 14% | 15% | 18% | **56%** | 30% | 17% | **16%** | 17% |
| *Observed Primed* | **18%** | **34%** | **20%** | 55% | **37%** | **28%** | 10% | **34%** |

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.

## 3 Evidence from a Human Experiment

This survey evaluates the applicability of our valuation function proposed for the DASM Problem. More details of methods and results can be found in the full version of our paper.

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 our full paper. 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. Inputs are assumed to be a $A$ of applicants and $E$ of employers, where each agent reports all appropriate approval lists (i.e., binary vectors). Let $n = |A|$ and $m = |E|$. All proofs and pseudocode can be found in the full version of this paper.

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

SmartPriorityMatch puts a "priority level" on each applicant-employer pair. For instance, a pair $(a, e) \in A \times E$ where $a \in \mathsf{aff}(e)$ and each agent is maximally interested in the match ($\mathsf{pr}_e^e(a) = \mathsf{pr}_e^a(e) = \mathsf{pr}_a(e) = 1$) 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 $a \in \mathsf{aff}(e)$ if they have maximum interest for the match: $\mathsf{pr}^e_e(a) = 1$, $\mathsf{pr}^a_e(e) = 1$, and $\mathsf{pr}_a(e) = 1$.

$G_1$- Edges between an $e \in E$ and $a \in A \setminus \mathsf{aff}(e)$ if they are interested in each other: $\mathsf{pr}^e_e(a) = 1$ and $\mathsf{pr}_a(e) = 1$.

$G_2$- Edges between an $e \in E$ and $a \in \mathsf{aff}(e)$ if they are interested in each other for their own match: $\mathsf{pr}^e_e(a) = 1$ and $\mathsf{pr}_a(e) = 1$.

$G_3$- Edges between an $e \in E$ and $a \in \mathsf{aff}(e)$ if $e$ is interested in itself for $a$ and $a$ is interested in $e$: $\mathsf{pr}^a_e(e) = 1$ and $\mathsf{pr}_a(e) = 1$.

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.*

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 another employer $e'$ in $G_1$. If $e$ likes the match $(e', a_1)$ and not $(e', a_2)$, then 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 $a, a' \in \mathsf{aff}(e)$ and $a$ prefers the swapped matching of $\mu$ with respect to $\mathcal{T}$, then:*

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

To achieve this, our maximal $b$-matchings must depend on lower-priority graphs. Instead of running the 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 $r : \mathcal{S} \to \mathbb{N}$ 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 our full paper for a simple greedy solution to this problem.

Our algorithm starts with a reserved matching on $G_1$, where reservations ensure we can still find a maximal matching on $G_0$ afterwards. Since each $a \in A$ may only be adjacent to $\mathsf{aff}^{-1}(a)$ in $G_0$, $G_0$ is a set of disjoint stars with centers $e \in E$. This 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 just 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 $r(N_0(e)) = \min(|N_0(e)|, q_0(e))$.

We now describe our algorithm. First, run a reserved maximal $b$-matching on $G_1$. Next, run the standard PriorityMatch process on $G_0$, $G_2$, and $G_3$. For more details, see the full version of our paper. The four resulting matchings are disjoint, and we can show 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.*

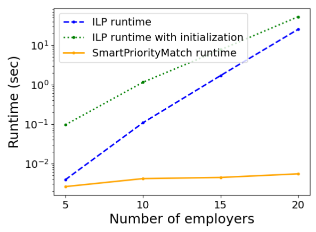
Finally, we can show that SmartPriorityMatch satisfies the conditions posed in Lemma 2. This concludes Theorem 1. We briefly note that this algorithm solves the problem for any weight $\lambda \in [0, 1]$,[2] however the algorithm itself does not depend on $\lambda$. Thus, there must exist a matching 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

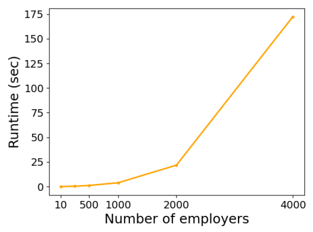
This section provides experimental validation for the polynomial-time scalability of SmartPriorityMatch, as analyzed in Theorem 1. To the best of our knowledge, our model is new, so there is no direct benchmark from the literature. Because of this, following the path of others (e.g., recently Cooper 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 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 our full paper. 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 the full version of

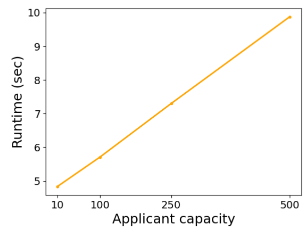
[2]We additionally note that with a slight modification to edge priority, our algorithm could work for $\lambda \in [0, \infty)$.



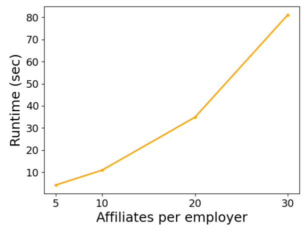
(a) Varying $m$ (small). Fix $\frac{n}{m} = 2$, $q = 3$, and $t = 0.5$.



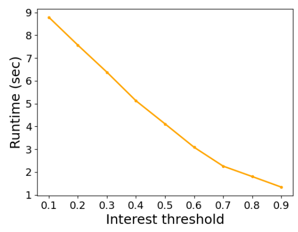
(b) Varying $m$ (large). Fix $\frac{n}{m} = 5$, $q = 5$, and $t = 0.5$.



(c) Varying $q$. Fix $m = 1000$, $\frac{n}{m} = 5$, and $t = 0.5$.



(d) Varying $n/m$. Fix $m = 1000$, $q = 5$, and $t = 0.5$.



(e) Varying $t$. Fix $m = 1000$, $q = 5$, and $n/m = 5$.

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

our paper. We use similar runtime experiments used by Tziavelis et al. [2019] adapted to the DASM setting. 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.

## Acknowledgements

We thank Haris Aziz and Duncan McElfresh for comments on earlier versions of this work. This research was supported in part by NSF CAREER Award IIS-1846237, NSF D-ISN Award #2039862, NSF Award CCF-1852352, NIH R01 Award NLM-013039-01, NIST MSE Award #20126334, DARPA GARD #HR00112020007, DoD WHS Award #HQ003420F0035, NSF BIGDATA Grant #1546108, NSF SPX Grant #1822738, NSF CCF grant #2114269, ARCS Endowment Award, and Ann G. Wylie Fellowship.

## 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.

[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, 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 Giannakopoulos, Katerina Doka, Nectarios Koziris, and Panagiotis Karras. Equitable stable matchings in quadratic time. In *NeurIPS*, 2019.

[Tziavelis *et al.*, 2020] Nikolaos Tziavelis, Ioannis Giannakopoulos, Rune Quist Johansen, Katerina Doka, Nectarios Koziris, and Panagiotis Karras. Fair procedures for fair stable marriage outcomes. In *AAAI*, 2020.