Matching with Partners and Projects [*]

Antonio Nicolò[†], Arunava Sen[‡], Sonal Yadav[§]

November 29, 2017

Abstract

We study a model that is a hybrid of the classical one-sided and two sided matching models. Agents are matched in pairs in order to undertake a project and have preferences over both the partner and the project they are assigned to. Each agent partitions the set of partners into friends and outsiders, and the set of of possible projects, into good and bad ones (dichotomous preferences). The overall preference ordering on partner, project pairs is separable. Friendship is mutual and preferences over projects among friends are correlated. We propose an algorithm, the minimum demand priority algorithm that generates stable assignments in this domain, satisfies a restricted notion of Pareto efficiency called friendship efficiency and is strategy-proof. Finally we show that stable assignments may not exist if any one of assumptions on the preferences is relaxed.

JEL classification: C78; D47; D71

Keywords: Matching; Stability; Strategy-proofness; Two-sided matching; One-sided matching

[*]We would like to thank Ahmet Alkan, Yann Bramoullé, Lars Ehlers, Jean-Jacques Herings, Michele Lombardi, Jordi Massó, Debasis Mishra, Herve Moulin, Hans Peters, Juan Sebastián Pereya, Ton Storcken, Fedor Sandomirskiy, Francesco De Sinopoli, Attila Tasnádi, William Thomson, Alexander Westkamp, Claudio Zoli and especially Ravindra B. Bapat, Marco LiCalzi and Thayer Morill for their comments and suggestions. We would also like to thank Siddharth Chatterjee and Aditya Vikram for carefully reading the paper and their comments. This paper has also benefited from the comments of seminar participants at Algiers, Alicante, Bath, Budapest, Delhi, Glasgow, Lisbon, Maastricht, Nizhny Novgorod, Saint Petersburg, Venice, Verona and York. This paper was earlier circulated with the title "Matching with Homophily".

[†]Department of Economics, University of Padova, via del Santo 33, 35123 Padova, Italy and School of Social Sciences, The University of Manchester, Manchester M13 9PL, United Kingdom.

[‡]Economics and Planning Unit, Indian Statistical Institute, 7 S.J.S Sansanwal Marg, New Delhi-110016, India.

[§]Department of Economics, University of Padova, via del Santo 33, 35123 Padova, Italy.

1 Introduction

In many situations agents are matched in teams in order to perform a task. Agents have preferences over the task that they are asked to perform as well as over the partners that they are assigned to work with. Consequently, forming stable teams is important - it ensures that agents do not have opportunities to abandon their assignments and do better for themselves.

A centralized authority matches agents in pairs and assigns them a task. We are interested in mechanisms that satisfy stability and provide incentives to agents to truthfully reveal their preferences. This problem shares some features with two-sided matching models like the marriage market since agents have preferences over their potential partners. However it also has common features with one-sided matching models like the house allocation problem, because a task has to be assigned to each pair of agents. In this sense our model is a hybrid version of the two classical models. It is easy to construct examples to show that stable allocations no longer exist in this setting. The contribution of our paper is then twofold: we propose a novel model of matching and we identify a suitable domain restriction where stable allocations exist. In the tradition of matching literature, we propose an algorithm that identifies stable allocations in this domain, provides incentives to agents to truthfully reveal their preferences and satisfies a restricted notion of efficiency. Our paper can be thought of as an attempt to extend the matching literature to team formation focussing on the case of matching in pairs in a well-defined but restricted preference domain.

Our model has several applications. It is a natural variant of the roommate problem where agents have preferences over potential roommates *and* available rooms. Another example is pair programming. Pair programming is a software development technique where two programmers work together at one work station. One individual, the driver writes the code while, the other individual, the observer, reviews each line of the code as it is typed in. The two programmers switch roles frequently. A website for pair programming is a centralized mechanism that matches programmers in pairs, and assigns them a job.

Yet another example is the problem of assigning term papers or projects to students. In several academic institutions, undergraduate or Master's students are required to take "Project" courses. Students have to undertake some independent (non-coursework) research in these courses. Due to the large number of students, they are often assigned in pairs (and sometimes in larger groups) to particular research topics. Students also benefit from learning from their peers. Grades in these courses depend in very large measure on a joint report prepared by a student team. As a result students have preferences over their assigned partner (often based on whether they "get on" with him/her and on their assessments of their abilities) and the project.

In our model there are a set of agents and a set of projects, and the number of projects is large enough so that a project can be assigned to each pair of agents. Each agent has a preference ordering over partners and projects. We make several assumptions about this preference ordering. Preferences are *separable* over partners and projects; preferences over each component are *dichotomous* i.e. alternatives in each component are partitioned into good and bad sets.[1] The set of possible partners is partitioned into friends (good partners) and outsiders (bad ones), and the set of projects into good and bad projects. Therefore every partner, project pair can be placed in one of four indifference classes, depending on whether the partner and the project are good or bad. Friendship is mutual and transitive, so that the set of agents can be partitioned into groups of friends. Finally, preferences of friends are correlated (*homophily*). Specifically, we assume a strong alignment in the preference for projects among friends: for any pair of friends, the set of good projects of one of them is weakly contained in the set of good projects of the other one. However, two friends need not have the same set of good projects. One agent can be fussier than the other and like only a subset of the projects liked by the other agent.

We are interested in finding stable allocations in this preference domain. Stability requires that no set of agents can block an assignment, and in our setting we admit three possible ways of blocking an assignment. First, two agents can leave their current partners and form a new team to perform a project which is left unassigned, if this makes both agents strictly better off. Second, a group of agents can block an assignment via a position swap: agents swap places amongst themselves to strictly improve. Blocking via an unassigned project or via position swaps leads to a change in both partner and project for the agents who are involved. Third, pairs of agents may swap projects amongst themselves and strictly improve; therefore agents who only swap projects keep their partners. We propose an algorithm, the minimum demand priority algorithm (MDPA) that generates stable allocations and provides agents incentives to truthfully reveal their preferences. It also satisfies a restricted notion of Pareto efficiency called friendship efficiency, meaning that the proposed allocation cannot be Pareto dominated by any allocation obtained reallocating partners and projects among friends (in our domain we allow for indifferences so that stability does not guarantee efficiency).

Our mechanism works as follows. We arbitrarily order groups of friends and we allocate projects to each group, one by one. In the first step we remove one agent from each group of friends if the size of this group is an odd number leaving the group unchanged otherwise. We then consider available projects with positive demand, that is projects which are regarded as good by at least one agent. Within each group, we allocate projects sequentially starting from those that have the minimum demand. Appropriately designed tie-breaking rules specify the pair of agents to whom a project is assigned when more than two agents demand it, and also the way to complete a pair when only one agent considers a project good. Once all projects with positive demand are assigned, we pair the remaining friends and allocate them an arbitrary project. We repeat the procedure for each group of friends. Since we have

[1] The assumption of dichotomous preferences is quite common in the matching literature. See for instance Bogomolnaia and Moulin (2004), Roth et al. (2005) etc.

removed one agent from each group of friends with odd cardinality, there may exist a set of agents who are still not paired and who are not friends among themselves. We arbitrarily order these agents and form as many pairs as possible who have a common good project. For the remaining pairs we assign projects that are good for at least one agent in the pair, whenever this is possible.

An important feature of the MDPA is that within a group, projects are first allocated to less fussy agents. As a result, the more fussy agents may be rationed out of good projects and be allocated bad projects. In other words, the algorithm is designed to penalize fussy agents relative to the less fussy ones. This aspect of the construction is helpful for both stability and strategy-proofness. It is hard for fussy disgruntled agents to find a good project to block with. Less fussy agents also have more ways to block and manipulate in case they receive bad projects; thus making assignments using the minimum demand principle makes sense both for stability and strategy-proofness.

It may seem equally logical to first allocate projects to more fussy agents in an effort to maximize the number of agents who receive good projects in an assignment. This would involve allocating projects via maximal demands where we sequentially allocate projects with higher demand before those with lower demand. However we show by means of an example that allocating projects within a group in this way provides incentives to less fussy agents to pretend to be more fussy in order to get a good project. We provide other examples that indicate that the MDPA is a salient procedure.

Finally, we demonstrate that our assumptions regarding homophily and dichotomous preferences are crucial for the existence of stable assignments. For instance, in the absence of homophily, stable assignments fail to exist when we keep all the other assumptions. If we depart from the dichotomous domain, impossibility results occur even if strong restrictions such as single peakedness are imposed.

1.1 Existing Literature

To the best of our knowledge, the model that we consider is distinct from others in the literature. However there are existing models that bear some resemblance to ours. We first briefly discuss some of these models and point out the essential differences. Finally we survey the sociological network formation literature which provides evidence on the existence of homophily.

Our model extends the classical roommate problem (see Roth and Sotomayor (1992) for a discussion) by allowing agents to have preferences over roommates *and* rooms. This fact renders the two models completely different - a more detailed account of the differences can be found in Section 8.2 in Nicolò et al. (2017).

A more closely related model is the "stable activities" (or SA) variant of the roommate problem studied in Cechlárová and Fleiner (2005). In the SA model different kinds of partnerships are possible between every pair of agents. A partnership between agents can be thought of as a common activity such as playing chess, going to the movies together and so on. Each agent has preferences over partner, activity pairs in which respect it is similar to our model. However activities can be replicated arbitrarily in the SA model, i.e. several pairs can be assigned the same activity. In our model, on the other hand, a project can be assigned to only one pair if it is assigned at all. This makes the two problems completely different both in terms of formulation and results. For instance, we have a notion of blocking via project swaps that is meaningless in the SA problem. The main result of the Cechlárová and Fleiner (2005) paper is that there is an equivalence between the SA problem and the roommates problem. This is emphatically not the case in our model as we point out in Nicolò et al. (2017). Importantly, it may not be possible to extend a stable roommate assignment to a stable assignment of partners and projects in our setting.

Sethuraman and Smilgins (2016) extend the classical two sided marriage problem by including a set of objects. There are two disjoint sets of agents (men and women) and each woman (man) has a preference over possible man (woman), object pairs. The cardinality of the three sets is assumed to be equal. A matching consists of (man, woman, object) triples. They show by means of examples that stable matchings may not exist when both agents sharing an object have a "right" over it, i.e., an agent can evict her partner from the object and find a new partner to strictly improve (identical to our notion of blocking via position swaps). The paper goes on to assume that only one set of agents (either men or women) own the objects. The unified agent-object pair can be considered as a single "agent". A blocking pair involves a man, a woman and an object that is owned by one of the two. A matching is M-stable (or W-stable) if there does not exist a blocking coalition when men (women) own the objects. They show that an M-stable and a W-stable matching exist in any matching market. The basic model is similar to ours but differs in three aspects. First in our model, there is no man-woman distinction and pairs have to be formed from the same set of agents. Second, the number of projects can be greater than the number of pairs of agents. Consequently, another notion of blocking arises in our model - blocking via unassigned projects. Finally, the most important distinction between the two papers is their respective responses to the underlying impossibility result. Sethuraman and Smilgins (2016) assign property rights to agents on one side of the market reducing it to a bi-partite matching model. We do not assign property rights over the projects and instead, we impose strong but plausible (in our opinion) restrictions on preferences and demonstrate the existence of assignments satisfying several desirable properties.

Combe (2017) also considers a model where two distinct sets of agents have to be matched in pairs to a common set of objects. The paper follows an approach similar to that of Sethuraman and Smilgins (2016). It introduces a notion of objects ownership and shows that stable matchings exist when ownership is given to agents in the same set.[2]

Pycia (2012) proves a very general existence result on coalition stability. However, his model does not cover ours because of the presence of objects in our model. Since objects do not have preferences over agents they are assigned to, our notions of blocking do not have a counterpart in his setting. In particular, if objects are interpreted as agents, deviating coalitions that involve objects cannot strictly improve. The difference between the models is manifested in the results as well. According to the main result of Pycia (2012), a coalition structure is stable if and only if agents' preferences are "pairwise-aligned" [3] i.e. every pair of agents rank coalitions that contain both of them, in the same way. It is easy to check that our preference restrictions do not satisfy pairwise alignment, yet stable assignments (in our sense) are shown to exist.

A special case of the question investigated in Pycia (2012) is the existence of stable threesome matchings. Alkan (1988) considers societies where there are three sets of agents (men, women and children) and a matching consists of distinct triples, each formed by a man, a woman and a child. Agents have preferences over the pairs they are matched with; thus a blocking coalition is a triple, where each member in the coalition strictly improves. Alkan (1988) shows that stable matchings fail to exist, even when the preferences are restricted to be separable. Biró and McDermid (2010) considers a three-sided model with cyclic preferences, where men care only about women, women only about children and children only about men. As discussed in the earlier paragraph, our model cannot be interpreted as a special case of these models.

In a slightly different spirit, Raghavan (2014) considers an allocation problem where agents have to be assigned in pairs to objects. An object is either assigned to a unique pair of agents or not assigned at all. However, unlike in the roommate problem and ours, agents only have preferences over objects and not on potential partners. The paper examines allocation rules from the perspective of strategy-proofness and efficiency.

Our model is also related distantly to the those on the existence of stable matchings in two-sided matching models where couples are looking for jobs in the same labour market. A convenient example is the one where doctors are seeking internships at hospitals and some of the doctors are couples. (See Klaus and Klijn (2005), Klaus and Klijn (2007), Klaus et al. (2007) and Khare and Roy (2016)). In these models, the identity of the couples is predetermined. Every agent has an individual preference on the hospitals and a joint preference on the possible hospital duples. Hospitals also have preferences over the students. It is clear that our model is fundamentally different from these models - in our case, the formation of couples is endogenous while there is no counterpart of the hospital agent.

We now make a few remarks to motivate our assumption on the preference similarities

[2]Burkett et al. (2016)analyze two versions of the random serial dictatorship mechanism in a model where pairs of roommates have to be assigned to rooms and agents care about their roommates and their rooms.

[3]Certain richness assumptions on preferences are made as well.

of agents who are "friends". Sociological literature provides ample evidence of the existence of homophily, that is, the tendency of individuals to connect and form close ties with other individuals who are similar to them. Lazarsfeld et al. (1954) distinguished two types of homophily: status homophily, in which similarity is based on informal, formal, or ascribed status, and value homophily, which is based on values, attitudes, and beliefs. These two forms of homophily depend on each other, because individuals tend to have ties with whom they share similar experiences, like school, sports, hobbies, or with those who live in the same neighborhood or with those who belong to the same age group. Hence, we not only tend to build closer ties with individuals which are similar to us[4], but we also become more similar to our friends over time (Bauman and Ennett, 1996). All this amounts to an observed similarity between tastes and friendships, with friends sharing the same preferences for music (Selfhout et al., 2009), food and so on. More relevant for our paper, Ruef et al. (2003) have shown that homophily and network constraints based on strong ties have the most pronounced effect on the composition of entrepreneurial founding teams.

The remainder of the paper is organized as follows. In Section 2 we present the model. Section 3 describes the MDPA algorithm and Section 4 presents its normative properties. In Section 5 we present alternative procedures that satisfy some but not all the normative properties of the MDPA algorithms. Section 6 shows that stable assignments may not exist if either the homophily or the dichotomous domain assumptions is dispensed with. Section 7 concludes. The Appendix provides the proofs of the main theorems.

2 The Model

There is a finite set of agents $N = \{1, 2, \ldots, i, j, \ldots, n\}$ and a finite set of alternatives (projects), $A = \{a, b, c, d \ldots\}$. We assume that $|n| = 2m$ for some integer $m \geq 2$ and that $|A| \geq m$.

An *assignment* σ is a collection of triples (i, j, a) with the interpretation that agent pair (i, j) is assigned project a. In order to ensure feasibility, we require each agent to be paired with one other agent and with one project. In addition, every project is assigned to exactly one pair or left unassigned. We require all agents to be assigned a partner and a project. We shall say that the triple (i, j, a) is an element of σ if the pair (i, j) is assigned to the project a. Abusing notation, we shall also let $\sigma(i) = (j, a)$ if (i, j, a) is an element of σ. Feasibility imposes some obvious restrictions on σ which we do not elaborate for convenience. Finally let u^σ denote the set of unassigned projects in the assignment σ i.e. it is the set of projects a such that there does not exist agents i and j with $(i, j, a) \in \sigma$.

Let Σ denote the set of all feasible assignments.

[4]See Cohen (1977), Kandel (1978), Verbrugge (1983), McPherson et al. (2001), Golub and Jackson (2012).

2.1 Preferences

Each agent i has a preference ordering[5] $\succ_i$ over partner, project pairs (j, a) where $j \neq i$. We impose several restrictions on $\succ_i$ which we believe are natural in our model. In Sections 6.1 and 6.2 we show that without some of these restrictions satisfactory assignments do not exist.

Every agent i has a set of good partners (friends) $P^i \subseteq N \setminus \{i\}$ and good projects $G_i \subseteq A$. We allow for the possibility of $P^i, G^i = \emptyset$. We denote the complement sets of G^i and P^i by $A \setminus G^i$ and $N \setminus P^i$ respectively. These sets will be referred to as bad projects and bad partners respectively. The ordering $\succ_i$ satisfies the following properties:

(i) If j, k belong to the same element of the partition $\{P^i, N \setminus P^i\}$ and a, b belong to the same element of the partition $\{G^i, A \setminus G^i\}$, then $(j, a) \sim_i (k, b)$.

(ii) If $a \in G^i$, $b \notin G^i$, then $(k, a) \succ_i (k, b)$ for all $k \neq i$.

(iii) If $k \in P^i$, $l \notin P^i$, then $(k, a) \succ_i (l, a)$ for all $a \in A$.

These restrictions imply $\succ_i$ is *separable* over partners and projects. In addition, preferences over each component (partners and projects) are *dichotomous* i.e. alternatives in each component are partitioned into good and bad sets.

According to these preferences all partner, project pairs can be placed in one of four indifference classes: (I) both partner and project are good, (II) only the partner is good (III) only the project is good and (IV) neither partner nor project is good. We shall refer to the classes (I), (II), (III) and (IV) as (G, G), (G, B), (B, G) and (B, B) respectively. In general, we adopt the convention that the first component in the ordered pairs refers to the type of partner and the second to the type of project.

Separability implies (G, G) is the most preferred equivalence class and (B, B), the worst. Classes (G, B) and (B, G) can be ranked either way. We refer to the case where (G, B) is ranked above (B, G) as a *partner dominant* preference. In this case agent i prefers a good partner, bad project pair over a bad partner, good project pair.

Similarly, a *project dominant* preference is one where (B, G) is ranked above (G, B) i.e. agent i prefers a bad partner, good project pair over a good partner, bad project pair.

We assume the following (i) if $j \in P^i$ then $i \in P^j$ and (ii) if $i \in P^j$ and $j \in P^k$, then $i \in P^k$. Thus the friendship relationship is mutual (if i is j's friend, then j is i's friend) and transitive (if i is j's friend and k is i's friend, then k is i's friend).

These assumptions induce a partition on the set of agents N where each element in the partition is a set of agents who are all friends. We shall refer to each element of this partition

[5] An ordering is a binary relation which is complete, reflexive and transitive.

as a *friendship component*. These components are labelled $\{F_1, F_2, \ldots, F_L\}$. Note that the number of components which have an odd number of agents must be even.

Finally we assume *homophily* in the preferences for agents in the same friendship component. Specifically if i, j are friends, then either $G^i \subseteq G^j$ or $G^j \subseteq G^i$ holds. Homophily induces a strong alignment in the preference for projects among friends. In other words, there cannot be a pair of projects a, b and a pair of friends i, j such that i likes a but not b, while j likes b but not a.

We assume that the preference of agent i, $\succ_i$ satisfies all the conditions above. Thus $\succ_i$ is associated with a set of good partners and projects which we denote by $P^i(\succ_i)$ and $G^i(\succ_i)$ respectively.

2.2 Blocking and Stability

We consider three kinds of blocking in our model: (i) blocking via unassigned projects (ii) blocking via position swaps and (iii) blocking via a project swap.

Definition 1 *The pair of agents k, l blocks assignment σ via an unassigned project at profile $\succ$ if there exists $(k, i, a), (l, j, b) \in \sigma$ and $c \in u^\sigma$ such that $(k, l, c) \succ_k (k, i, a)$ and $(k, l, c) \succ_l (l, j, b)$.*

A pair of agents blocks in this sense if they can abandon their current partners and choose an unassigned project in the assignment which strictly improves their welfare from the initial situation.

Definition 2 *The agents $i_1, \ldots, i_k$ block σ by position swaps if there exist agents $j_1, \ldots, j_k$, projects $a_1, \ldots, a_k$ and a permutation $\mu : \{1, \ldots, k\} \to \{1, \ldots, k\}$ such that*

1. *$(i_s, j_s, a_s) \in \sigma$, $s = 1, \ldots, k$ and*
2. *$(j_{\mu(s)}, a_{\mu(s)}) \succ_{i_s} (j_s, a_s)$ for all $s = 1, \ldots, k$.*

According to this notion, the agents $i_1, \ldots, i_k$ swap places amongst themselves thereby acquiring new partners and projects. This change makes each of these agents strictly better off. This is a strong requirement. A weaker and perhaps more reasonable condition would require "the abandoned" partners to give consent for the swap. In Definition 2 above this would require the additional conditions $(i_s, a_{\mu(s)}) \succsim_{j_{\mu(s)}} (i_{\mu(s)}, a_{\mu(s)})$, $s \in \{1, \ldots, k\}$.[6] However we retain the stronger notion since we are able to show the existence of assignments which are immune to the stronger notion of blocking.

[6] Morrill (2010) investigates a similar question in the roommate model.

F_1	F_2
1 $\{a\}$	3 $\{b\}$
2 $\{a\}$	4 $\{b\}$

Table 1: Examples 1 and 2

Definition **3** *The pairs* (i_s, j_s), $s = 1, \ldots, k$ *blocks* σ *if there exist projects* $a_1, \ldots, a_k$ *and a permutation* $\mu : \{1, \ldots, k\} \to \{1, \ldots, k\}$ *such that*

1. $(i_s, j_s, a_s) \in \sigma$, $s = 1, \ldots, k$ *and*
2. $(j_s, a_{\mu(s)}) \succ_{i_s} (j_s, a_s)$ *and* $(i_s, a_{\mu(s)}) \succ_{j_s} (i_s, a_s)$.

In this notion, pairs of agents can swap their assigned projects amongst themselves and strictly improve. Note that both agents in a pair are required to strictly improve.

Examples 1, 2 and 3 show that the three notions of blocking are independent.

Example **1** Let $N = \{1, 2, 3, 4\}$ and $A = \{a, b, c, d\}$. The friendship components are $F_1 = \{1, 2\}$ and $F_2 = \{3, 4\}$. Table 1 describes the good project sets for the agents.

Let $\succ$ be the preference profile where $\succ_i$ is partner dominant for all i. Consider the assignment $\sigma = \{(1, 2, b), (3, 4, a)\}$. It is immune to blocking via an unassigned project since the unassigned projects c and d are bad for all agents.

It is immune to blocking via position swaps since the preference of each agent is partner dominant. For instance, consider agents 1 and 3. A position swap between 1 and 3 results in the assignment $\{(3, 2, b), (1, 4, a)\}$ which makes both 1 and 3 worse off.

Suppose the pairs $(1, 2)$ and $(3, 4)$ exchange projects. All four agents now strictly improve to the (G, G) class. Therefore σ can be blocked via a project swap.

Example **2** Let $N = \{1, 2, 3, 4\}$ and $A = \{a, b, c\}$. The friendship components are $F_1 = \{1, 2\}$ and $F_2 = \{3, 4\}$. Table 1 continues to specify the set of good projects for the agents.

Let $\succ$ be the preference profile where $\succ_i$ is partner dominant for all i. Consider the assignment $\sigma = \{(1, 2, c), (3, 4, b)\}$.

It can be blocked by agents $(1, 2)$ via the unassigned project c which improves their welfare from the (G, B) to the (G, G) class.

As in Example 1 the assignment cannot be blocked via a position swap because all the agents have partner dominant preferences. Finally note that 3 and 4 have a good project in the assignment σ and a project switch will make them worse off.

Example **3** Let $N = \{1, \ldots, 6\}$ and $A = \{x, y, e, f\}$. The friendship components are $F_1 = \{1, 2, 3\}$, $F_2 = \{4, 5\}$ and $F_3 = \{6\}$. Table 2 describes the good project sets for the agents.

F_1	F_2	F_3
1 $\{x\}$	4 $\{e, f\}$	6 $\{x\}$
2 $\{x, y\}$	5 $\{e, f\}$	
3 $\{x, y\}$		

Table 2: Example 3

F_1	F_2
1 $\{x\}$	3 $\{y\}$
2 $\{x, y\}$	4 $\{x, y\}$

Table 3: Example 4

Let $\succ$ be the preference profile where $\succ_1$ is project dominant and $\succ_i$ is partner dominant for all $i \in N \setminus \{1\}$. Consider the assignment $\sigma = \{(1, 2, y), (4, 5, e), (3, 6, x)\}$. The only agents who can strictly improve to a higher indifference class are 1 and 3.

The only unassigned project is f. Since f is not a good project for 1 and 3. The assignment σ cannot be blocked via an unassigned project.

There do not exist four agents who can strictly improve from σ. Therefore σ cannot be blocked via a project swap.

However σ is blocked by agents 1 and 3 by a position swap. The assignment after the swap is $\{(3, 2, y), (1, 6, x)\}$. Agent 1 strictly improves as x is a good project for 1 and $\succ_1$ is project dominant. Agent 3 strictly improves because she gets a good partner and still gets a good project.

We require an assignment to be immune to all three kinds of blocking.

DEFINITION 4 *The assignment σ is stable at $\succ$ if it cannot be blocked by unassigned projects, position swaps or project swaps.*

Consider two preference profiles which are induced by the same good partner and good project sets for all agents. The following example show that an assignment which is stable at one preference profile need not be stable at the other.

EXAMPLE 4 Let $N = \{1, 2, 3, 4\}$ and $A = \{x, y, z\}$. The friendship components are $F_1 = \{1, 2\}$ and $F_2 = \{3, 4\}$. The set of acceptable projects are summarized in Table 3.

Let $\succ$ be the preference profile where $\succ_i$ is partner dominant for all i. The assignment $\sigma = \{(1, 2, y), (3, 4, x)\}$.

Agents 2 and 4 are in the (G, G) equivalence class. Thus agents 1 and 3 are the only agents who can strictly improve. Since z is the only unassigned project and z is not a good project for 1 and 3, σ cannot be blocked via an unassigned project. Also $\succ_1$ and $\succ_3$ are partner dominant, so that agents 1 and 3 will not swap positions with each other. This will make both agents worse off. The assignment cannot be blocked via a project swap because it is not possible to make all four agents strictly better off. Thus σ is stable at $\succ$.

Let $\succ'$ be the preference profile where $\succ'_i$ is project dominant for all i with the same friendship components and the good project sets as specified in Table 3. Suppose 1 and 3 swap positions. Since x and y are good projects for 1 and 3 respectively, σ is blocked via a position swap. Hence σ is not stable at $\succ'$.

Example 4 has an interesting property: there exists an assignment that is stable *irrespective* of whether an agent's preferences are partner or project dominant. Consider the assignment $\sigma' = \{(1, 2, x), (3, 4, y)\}$. All agents are in the (G, G) indifference class and thus no agent can strictly improve. Thus σ' is stable at any preference profile which is consistent with the friendship components and good project sets specified in the example.[7]

This observation motivates a stronger notion of stability and a general question.

The profiles $\succ$ and $\succ'$ are *good set equivalent* if $P^i(\succ_i) = P^i(\succ'_i)$ and $G^i(\succ_i) = G^i(\succ'_i)$ for all i.

Definition 5 *The assignment σ is robustly stable at profile $\succ$ if it is stable at all profiles $\succ'$ that are good set equivalent to $\succ$.*

Robust stability is clearly a desirable property if it can be satisfied. An assignment that satisfies it depends only on the structure of the friendship components and the set of good projects of agents. We have shown that such assignments exist in Example 4. We show below that this is no accident: such assignments always exist and our algorithm (described below) generates one.

3 The Minimum Demand Priority Algorithm

We describe an algorithm to generate an assignment which we refer to as the *Minimum Demand Priority Algorithm* (MDPA).

Let $\succ^N$ and $\succ^A$ be fixed orderings of the sets N and A respectively. If agent $i \succ^N j$, then agent i has priority over j.

Fix an arbitrary profile $\succ$. This profile induces friendship components $\{F_1, \ldots, F_L\}$ which are ordered as $F_1, F_2, \ldots, F_L$.

[7] An example which illustrates this property when the preferences of all agents are project dominant can be found in Nicolò et al. (2017) (Example 5 in Section 2.2).

Step 0: In each component F_q where $|F_q|$ is odd, remove the agent with the lowest priority in F_q and add this agent to the *Residual set* R. If $|F_q|$ is even, then no changes are made and the original component is retained. The adjusted friendship components are $\{\tilde{F}_1, \ldots, \tilde{F}_L\}$. Each adjusted component has even cardinality.

We shall make assignments in the components $\tilde{F}_1, \tilde{F}_2, \ldots, \tilde{F}_L$ in sequence. These will be labelled Steps 1 to L respectively. Each step comprises of an initial step, several intermediate steps and a termination step. For component $\tilde{F}_q$, the initial step will be denoted by Step $q.0$, intermediate steps by Step $q.1, q.2, \ldots$ and the termination step by Step $q.T$. At the start of the generic step $q.s$ where $q \in \{1, \ldots, L\}$ and $s \in \{0, \ldots, T\}$, the algorithm is provided three inputs: (i) the set of available projects denoted by $A(q.s)$, (ii) the set of unassigned agents $N(q.s)$ in component $\tilde{F}_q$ and (iii) the set of waiting agent, project pairs $W(q.s)$. For every step $q.s$, $|W(q.s)|$ is either 0 or 1.

Step 1.0: Here $A(1.0) = A$, $N(1.0) = \tilde{F}_1$ and $W(1.1) = \emptyset$. The demand for every available project $a \in A(1.0)$ is given by $D(a; (1.0)) = \#\{i \in N(1.0) : a \in G^i(\succ_i)\}$. The set of agents who demand project a is given by $S(a; (1.0)) = \{i \in N(1.0) : a \in G^i(\succ_i)\}$.

Remove all projects with zero demand. Consider the project (projects) with the least demand. In case there is more than one such project, pick the project which is ranked highest according to $\succ^A$. Denote the project with the lowest demand (after tie breaking) by a. There are two cases to consider.

1. $D(a; (1.0)) = 1$. Then no assignments are made at Step 1.0 and $W(1.1) = \{(i, a)\}$ where $S(a; (1.0)) = \{i\}$. Also sets $A(1.1) = A(1.0) \setminus \{a\}$, $N(1.1) = N(1.0) \setminus \{i\}$ and proceed to Step 1.1.

2. $D(a; (1.0)) \geq 2$. Assign a to the pair of agents with the highest and second highest priority in $S(a; (1.0))$. Now sets $A(1.1) = A(1.0) \setminus \{a\}$, $N(1.1) = N(1.0) \setminus \{i, j\}$ where i, j are the agents who have just been assigned a in this step and $W(1.1) = \emptyset$. Proceed to Step 1.1.

Step 1.1: Step 1.1 repeats Step 1.0 with the sets $A(1.1)$, $N(1.1)$ and $W(1.1)$ (determined at the end of Step 1.0) but with an important difference.

As in Step 1.0, consider the project with the least non zero demand with appropriate tie breaking. Suppose this project is b. Once again there are three possibilities.

1. $D(b; (1.1)) = 1$ and $W(1.1) \neq \emptyset$. Let $(i, a) \in W$ and $S(b; (1.1)) = \{j\}$. Then the pair (i, j) is assigned b. Also sets $A(1.2) = [A(1.1) \cup \{a\}] \setminus \{b\}$, $N(1.2) = N(1.1) \setminus \{j\}$, $W(1.2) = \emptyset$ and proceed to Step 1.2.

2. $D(b;(1.1)) = 1$ and $W(1.1) = \emptyset$. Then no assignments are made at Step 1.1 and $W(1.2) = \{(j,b)\}$ where $S(b;(1.0)) = \{j\}$. Also sets $A(1.2) = A(1.1) \setminus \{b\}$, $N(1.2) = N(1.1) \setminus \{j\}$ and proceed to Step 1.2.

3. $D(b;(1.1)) \geq 2$. Assign b to the pair of agents with the highest and second highest priority in $S(b;(1.1))$. Now sets $A(1.2) = A(1.1) \setminus \{b\}$, $N(1.2) = N(1.1) \setminus \{i,j\}$ where i,j are the agents who have just been assigned a in this step and $W(1.2) = W(1.1)$. Proceed to Step 1.2.

By construction at any stage, $W(1.s)$ is either null or consists of one element.

Step 1.2 repeats Step 1.1 with the appropriate sets. Proceeding in this way, there exists a step $1.T$ where $D(a;(1.T) = 0$ for every $a \in A(1.T)$. This is called the termination step. Here three cases can arise.

I. $N(1.T) = \emptyset$ and $W(1.T) = \emptyset$. This means that all agents in $\tilde{F}_1$ have been assigned a partner and a project.

II. $|N(1.T)|$ is even. Note that $W(1.T) = \emptyset$. Arrange the agents in $N(1.T)$ using $\succ^N$. Form pairs of consecutive agents proceeding in sequence. Assign projects to pairs in sequence using $\succ^A$ from $A(1.T)$.

III. $|N(1.T)|$ is odd. Then $W(1.T) \neq \emptyset$. Let $W(1.T) = \{(i,a)\}$. Arrange the agents in $N(1.T)$ using $\succ^N$. Suppose j is the highest priority agent in this set. Form the triple (i,j,a). Observe that $|N(1.T) \setminus \{j\}|$ is even. For the remaining agents, partners and projects are assigned as in (II) above.

This completes the assignment for all agents in the component $\tilde{F}_1$. Let the set of projects assigned to pairs in $\tilde{F}_1$ be $\Delta(\tilde{F}_1)$. We repeat the procedure for $\tilde{F}_2 =$ with $A(2.0) = A \setminus \Delta(\tilde{F}_1)$, $N(2.0) = \tilde{F}_2$ and $W(2.0) = \emptyset$. This completes the assignment for the agents in component $\tilde{F}_2$. Proceeding in this manner, at the end of Step L we have assigned partners and projects to all agents in $\tilde{F}_1 \cup \tilde{F}_2 \ldots \cup \tilde{F}_L$.

Step $L+1$: In this step, projects are assigned to agents in the set R. The set of available projects is $A(R) = A(\tilde{F}_L) \setminus \Delta(\tilde{F}_L)$. Without loss of generality (and by suitable relabelling of agents), let $R = \{1, 2, \ldots, 2r\}$ for $r \geq 0$. We make assignments for the agents in R in Steps $L+1.1$ through $L+1.r^*$. At each of these steps $L+1.k$, a set of agents R_k is obtained. The procedure terminates in Step $L+1.r^*$ where $\cup_{k=1}^{r^*} R_k = R$.

Step $(L+1.1)$: In this step, we match agent 1 with the agent with the lowest index $t > 1$ in R such that $G^1(\succ_1) \cap G^t(\succ_t) \cap A(R)$ is non empty. This pair is assigned a project a

in $G^1(\succ_1) \cap G^t(\succ_t) \cap A(R)$ (ties are once again broken according to $\succ^A$). In this case, $R_1 = \{i, t\}$. If no such agent t exists, then agent 1 is unmatched at this step and $R_1 = \{1\}$.

The set of projects assigned in this step is denoted by $\Delta_R(1)$ where $\Delta_R(1) = \emptyset$ if $R_1 = \{1\}$ and $\{a\}$ if $R_1 = \{1, t\}$ and a is assigned to $(1, t)$.

The set of agents remaining for the next step is $R \setminus R_1$. The set of projects available for the next step is $A(R) \setminus \Delta_R(1)$.

Step $(L + 1.2)$: We repeat Step $(L + 1.1)$ with agent 1 being replaced by the agent with the smallest index in $R \setminus R_1$ and the set of projects $A(R)$ replaced by $A(R) \setminus \Delta_R(1)$. This generates possibly another assignment. At the end of this step, we obtain R_2 and the set of available projects $A(R) \setminus \cup_{k=1}^{2} \Delta_R(k)$.

Proceeding in this manner, there will exist a Step $(L+1.r^*)$ where $\cup_{j=1}^{r^*} R_j = R$. The set of projects available for assignment in Step $(L + 1.r^*)$ is $A(R) \setminus \cup_{k=1}^{r^*} \Delta_R(k)$.

There are two possibilities.

1. All agents in R have been assigned a partner and a project. In this case the algorithm terminates.

2. Suppose 1 does not hold. Let $\bar{R}$ be the set of agents who have not been assigned a partner and a project. Note that $\bar{R}$ must be even in cardinality. Proceed to Step $L+2$.

Step $L + 2$: In this step, $A(R) \setminus \cup_{k=1}^{r^*} \Delta_R(k)$ is the set of available projects. Partition $\bar{R}$ into $\{\bar{R}^1, \bar{R}^2\}$ where $\bar{R}^1$ consists of agents who have no good projects among the set of available projects and $\bar{R}^2$ are the remaining agents.

Order the agents in $\bar{R}^1$ and $\bar{R}^2$ according to $\succ^N$. Assign the highest priority agent in $\bar{R}^1$ with the highest priority agent in $\bar{R}^2$ and an available project in the good set of the agent in $\bar{R}^2$, the agent with the second highest priority in $\bar{R}^1$ with the agent with the second highest priority in $\bar{R}^2$ and so on.

1. If $|\bar{R}^1| = |\bar{R}^2|$, then the procedure will terminate with all agents in R being assigned a project and partner.

2. If $|\bar{R}^1| > |\bar{R}^2|$, then an even number of agents in $\bar{R}^1$ will be left unassigned. They are now paired consecutively and are assigned a project from the available set according to $\succ^A$.

3. If $|\bar{R}^1| < |\bar{R}^2|$, then an even number of agents in $\bar{R}^2$ are left unassigned. They are paired consecutively and assigned a project from the good set of the higher priority agent (ties are broken according to $\succ^A$).

This completes the description of the algorithm. We illustrate it with an example.

F_1	F_2	F_3	F_4
4 $\{x, y, z, w, q\}$	12 $\{a, b, c, d\}$	**23** $\{f\}$	**24** $\{e\}$
2 $\{x, y, z, w\}$	13 $\{a, b, c\}$		
6 $\{x, y, z, w\}$	14 $\{a, b, c\}$		
3 $\{x, y, z\}$	15 $\{a, b\}$		
7 $\{x, y\}$	16 $\{a, b\}$		
8 $\{x, y\}$	17 $\{a, b\}$		
1 $\{x\}$	18 $\{a\}$		
5 $\{x\}$	19 $\{a\}$		
9 $\{x\}$	20 $\{a\}$		
10 $\{x\}$	21 $\{a\}$		
11 $\{x, y, z, w, q, e\}$	**22** $\{a\}$		

Table 4: Example 5

Example **5** Let $N = \{1, 2, \dots, 24\}$ and $A = \{a, b, \dots, z\}$. Consider a preference profile which induces the components: $F_1 = \{1, \dots, 11\}$, $F_2 = \{12, \dots, 22\}$, $F_3 = \{23\}$ and $F_4 = \{24\}$. Table 4 summarizes the good project sets for all agents. The priority order of agents is $1 \succ^N 2 \succ^N 3 \dots 24$ and the priority order of projects is $z \succ^A y \dots \succ^A a$.

In Step 0, the adjustment of the components results in $R = \{11, 22, 23, 24\}$. The adjusted components are $\tilde{F}_1 = F_1 \setminus \{11\}$, $\tilde{F}_2 = F_2 \setminus \{22\}$. The MDPA assigns projects to $\tilde{F}_1$, $\tilde{F}_2$, R in sequence.

Step 1: We assign projects to agent pairs in $\tilde{F}_1$. Table 5 illustrates the demand for the available projects, the waiting set and the assignment made at every substep of step 1.

In Step 1.0, q is the least demanded project (with $D(q; 1.0) = 1$). So no assignment is made in Step 1.0. Also $W(1.1) = \{(4, q)\}$, $A(1.1) = A(1.0) \setminus \{q\}$ and $N(1.1) = N(1.0) \setminus \{4\}$. In Step 1.1, the project with the least demand is w (with $D(w; (1.1)) = 2$. Now the MDPA assigns w to the pair $(2, 6)$. The set of agents and projects for the next step are updated by removing agents $2, 6$ and project w. Also $W(1.2) = W(1.1) = \{(4, q)\}$. In Step 1.2, the project with least demand is z with demand 1 and $W(1.2) = \{4, q)\}$. So the triple $(4, 3, z)$ is formed in this step. The set of available projects in Step 1.3 is $[A(1.2) \cup \{q\}] \setminus \{z\}$. Also $W(1.3) = \emptyset$. In Step 1.3, project y is the least demanded with $D(y; (1.3)) = 2$. So the MDPA forms $(7, 8, y)$. Since $W(1.3) = \emptyset$, we have $W(1.4) = \emptyset$. In Step 1.5, the least demanded project is x with $D(x; (1.4)) = 4$. We have $S(x; 1.4) = \{1, 5, 9, 10\}$. Since 1 and 5 are the agents with the highest and second highest priority in $S(x; (1.4))$ according to $\succ^N$, the MDPA forms $(1, 5, x)$. Finally Step 1.5 is the termination step for the component $\tilde{F}_1$ as the demand for each available project in this step is zero. Note $W(1.5) = \emptyset$ and agents $9, 10$ are unassigned. The pair $(9, 10)$ is assigned the highest ranked available project according

Step $1.q$	x	y	z	w	q	$W(1.q)$	Assignment
1.0	10	6	4	3	1	$\{(4,q)\}$	No assignment
1.1	9	5	3	2	NA	$\{(4,q)\}$	$(2,6,w)$
1.2	7	3	1	NA	NA	$\{(4,q)\}$	$(4,3,z)$
1.3	6	2	NA	NA	NA	$\emptyset$	$(7,8,y)$
1.4	4	NA	NA	NA	NA	$\emptyset$	$(1,5,x)$
1.5	NA	NA	NA	NA	NA	$\emptyset$	$(9,10,v)$

Table 5: Step 1 in Example 5

Step $2.q$	a	b	c	d	$W(2.q)$	Assignment
2.0	10	6	3	1	$\{(12,d)\}$	No assignment
2.1	9	5	2	NA	$\{(12,d)\}$	$(13,14,c)$
2.2	7	3	NA	NA	$\{(12,d)\}$	$(15,16,b)$
2.3	5	NA	NA	NA	$\{(12,d)\}$	$(17,18,a)$
2.4	NA	NA	NA	NA	$\{(12,d)\}$	$(12,19,d),(20,21,u)$

Table 6: Step 2 in Example 5

to $\succ^A$. So $(9,10,v) \in \sigma$.

Step 2: We make assignments to agent pairs in $\tilde{F}_2$. Table 6 provides the demand for the available projects, the waiting set and the assignment made in every substep of Step 2.

In Step 2.0, the project with the least demand is d, with $D(d;(2.0)) = 1$. Thus no assignment is made and $W(2.1) = \{(12,d)\}$. Note that in any Step $2.q$, $q \in \{1,2,3\}$, the least demanded project, say a has demand $D(a;(2.q)) \geq 2$: this project is assigned to the highest and the second highest priority agents in $S(a;(2.q))$. So $W(2.q) = \{(12,d)\}$. For instance, the triple $(15,16,b)$ is formed by the MDPA in Step 2.2. The termination step for component $\tilde{F}_2$ is Step 2.4: $(12,d) \in W(2.4)$ and $19, 20, 21$ are unassigned. Since $19 \succ^N 20 \succ^N 21$, we have $(12,19,d)$ and $(20,21,u)$.

Step 3: Partner, project assignments are made for the agents in $R = \{11,22,23,24\}$. Note that $G^{11} \cap A(R) = \{q,e\}$, $G^{22} \cap A(R) = \emptyset$, $G^{23} \cap A(R) = \{f\}$ and $G^{24} \cap A(R) = \{e\}$. Also $11 \succ^N 22 \succ^N 23 \succ^N 24$. In Step 3.1, $(11,24,e)$ is formed by the MDPA. This is because 11 is the highest priority agent in this step and has a common good available project with agent 24. Note that $G^{11} \cap G^j \cap A(R) = \emptyset$ for $j \in \{22,23\}$. So $R_1 = \{11,24\}$ and $\Delta_R(1) = \{e\}$. In Step 3.2, the set of remaining agents is $R \setminus R_1 = \{22,23\}$. The set of available projects is $A(R) \setminus \Delta_R(1)$. Agent 22 is the higher priority agent in $R \setminus R_1$. Since $G^{22} \cap G^{23} \cap A(R) = \emptyset$, no assignment is made in this step. Also $R_2 = \{22\}$ and $\Delta_R(2) = \emptyset$. In Step 3.3, we again have $R_3 = \{23\}$ and $\Delta_R(3) = \emptyset$. Step 3.4 is the termination substep for R, where agents $22, 23$ are unassigned. The MDPA proceeds to Step 4.

Step 4: We have $\bar{R}^1 = \{22\}$ and $\bar{R}^2 = \{23\}$. The MDPA forms $(22, 23, f)$. The algorithm terminates at Step 4.

4 Properties of the MDPA

In this section we show that the MDPA satisfies several important properties.

4.1 Stability

We have already defined and discussed the notion of stability. We show below the MDPA generates a robustly stable assignment.

Theorem **1** *The MDPA algorithm generates a robustly stable assignment at every $\succ$. Consequently a robustly stable assignment exists at every profile.*

The proof of Theorem 1 is provided in the Appendix.

4.2 Strategy-Proofness

In this section we investigate the strategic properties of the MDPA. We assume that friendship is commonly observable. Since friendship is mutual and transitive, no agent can *individually* manipulate and misreport her set of good partners or friends. We assume that the set of good projects for an agent is private information and can be misreported by an agent if she believes this could be advantageous. However the assumption of homophily introduces some complications in the standard model as it imposes a restriction on preference *profiles*. Therefore individual announcements of good sets may lead to profile announcements that are inconsistent with homophily. Below, we propose a model that satisfactorily deals with this issue.

For any friendship component F_q, there is a commonly known linear order $\succ_q^O$ over the set of all projects with the following interpretation: for any $x, y \in A$, if $x \succ_q^O y$ then all agents in F_q who like x also like y. The private information of an agent is a *threshold* project x. All projects y such that $x \succ_q^O y$ are good. We believe this is a natural assumption that also ensures that any announced profile of threshold projects is consistent with homophily.

We make a minor modification to the MDPA for convenience. We assume that while making assignments in any component the order $\succ_q^O$ is used to break ties when there are several projects with identical minimum demand. This amounts to a version of the MDPA

F_1	F_2
1 $\{x, y, z\}$	3 $\{z\}$
2 $\{x, y, z\}$	4 $\{z\}$

Table 7: Example 6

where a different order on A is used for each component and for the set R; robust stability is not affected by this modification.

We now describe a general mechanism in this setting. Each agent $i \in F_q$, $q \in \{1, \dots, L\}$ announces a preference ordering $\succ_i$ that contains two pieces of information: (i)a set of good projects, $G^i(\succ_i)$ and (ii) whether her preferences are partner or project dominant. It is assumed that the set $G^i(\succ_i)$ is consistent with $\succ_q^O$ i.e. if $x \in G^i(\succ_i)$ and $x \succ_q^O y$, then $y \in G^i(\succ_i)$. Let Γ_i denote the set of all possible announcements of agent i. Recall that Σ is the set of all feasible assignments.

An assignment rule is a map σ, $\sigma : \times_{i \in N} \Gamma_i \to \Sigma$.

Definition 6 *An assignment rule σ is strategy-proof if there does not exist $\succ, \succ'_i \in \Gamma_i$ and $\succ_{-i} \in \times_{j \neq i} \Gamma_i$ such that $\sigma(\succ'_i, \succ_{-i}) \succ_i \sigma(\succ_i, \succ_{-i})$.*

The notion of strategy-proofness is standard: an agent cannot strictly improve by misreporting her preferences for any possible announcements of the preferences of other agents. Our main result in this section is the following.

Theorem **2** *The MDPA algorithm is strategy-proof.*

The proof of Theorem 2 can be found in the Appendix.

4.3 Efficiency

An obvious question is whether the MDPA generates an assignment that is Pareto efficient i.e. is it possible to strictly improve the well being of at least one agent without making anyone worse off? It is not, as the following example demonstrates.

Example **6** Let $N = \{1, 2, 3, 4\}$ and $A = \{x, y, z\}$. The friendship components are $F_1 = \{1, 2\}$ and $F_2 = \{3, 4\}$. Table 7 describes the good project sets for the agents. Let the priority order of agents be $1 \succ^N 2 \succ^N 3 \succ^N 4$ and $z \succ^A y \succ^A x$. The components are assigned in the sequence F_1 followed by F_2. Note that $R = \emptyset$. The MDPA generates $(1, 2, z), (3, 4, x)$. But the Pareto efficient assignment is either $\{(1, 2, x), (3, 4, z)\}$ or $\{(1, 2, y), (3, 4, z)\}$.

There is however a weaker sense in which the MDPA is efficient which we describe below.

Consider a component F_q with $|F_q|$ even, a set of projects A' and a preference profile $\succ$ for agents in F_q.

A *component assignment rule* is a rule that pairs agents in F_q and assigns a unique and distinct project to each pair from A'. [8]

The *happiness index* attained by a component assignment rule is the number of agents who are assigned a good project.

A component assignment rule π satisfies *component efficiency or friendship efficiency* if there does not exist a component assignment rule whose happiness index exceeds that of π. Note that when we compare one component assignment rule to another, *we are assigning projects from A' to the agents in F_q.*

The MDPA specifies a component assignment rule, which assigns projects to the agents in a component via minimum demand. The MDPA satisfies component efficiency, in the sense that there does not exist another component assignment rule whose happiness index exceeds that of the MDPA.

We state this result formally below.

Theorem **3** *Consider an even component F_q, a set of projects A' and a preference profile $\succ$. The MDPA satisfies friendship efficiency i.e. there does not exist a component assignment rule whose happiness index exceeds that of the assignment generated by the MDPA.*

The proof of Theorem 3 is contained in the Appendix. Below we illustrate the argument involved in the proof with reference to Example 5 that we have considered earlier.

Consider component $\tilde{F}_2 = \{12, 13, \ldots, 21\}$. The set of projects available for assignment to $\tilde{F}_2$ is $A \setminus \{w, z, y, x, v\}$. The happiness level of the MDPA for this component is 7: agents 19, 20 and 21 are not assigned good projects. Observe that agents 15 to 21 (7 agents) like only two projects a and b. These two projects can be assigned to at most four agents. Therefore at least three agents must remain with bad projects for any component assignment rule.

The MDPA could be modified to marginally improve efficiency; however these modifications would not guarantee Pareto efficiency. We leave the question about the existence of robustly stable, strategy-proof and Pareto efficient mechanisms for future research.[9]

[8]Note that the component assignment can depend on the preferences of the agents in F_q. However, we hold these preferences fixed for the analysis in this section of the paper.

[9]For any friendship component at the termination step, we may end up assigning projects to agents for whom all available projects are bad. The assigned projects can be good for some agents belonging to subsequent friendship components. Therefore, postponing this assignment of projects to the very end of the mechanism removes a source of inefficiency.

5 Other Component Assignment Procedures

We have seen that the MDPA makes assignments within a friendship component by sequentially allocating projects with lower demand before those with higher demand. This procedure satisfies strategy-proofness and friendship efficiency. We have remarked earlier that the procedure favours the less fussy agents to the more fussy agents. Below we provide several natural procedures for allocation within components that violate either friendship efficiency or strategy-proofness. Although we do not have a formal characterization result, we believe that the MDPA is a salient procedure for allocation within components.

5.1 A minimum demand endogeneous priority procedure

Let $\tilde{F}_q$ be a component with an even number of agents. An ordering of agents in the component is constructed based on their good sets; in particular i is ranked higher than j if $G^j \subseteq G^i$. In case $G^i = G^j$, ties are broken using a fixed order.

Agents ranked 1 and 2 in this order are now paired with each other. They are assigned a project that is in the good set of both the agents and is a minimally demanded project amongst agents in the component. If there is more than one such project, then ties are broken according to a fixed order on projects. Agents ranked 3 and 4 are then paired together with a project using the same principle i.e. with a project that is a good project for both agents amongst the set of minimally demanded available projects. Similarly agents ranked $5, 6, 7, 8$, etc. are paired together. If there does not exist a project that is good for both agents, assign a project that is good for at least one. Otherwise assign a project arbitrarily from the set of available projects.

In Example 5, for the component $\tilde{F}_2$, the mechanism generates the assignment $(12, 13, c)$, $(14, 15, b)$, $(16, 17, a)$, $(18, 19, x)$ and $(20, 21, y)$. The number of unhappy agents for this procedure is 4, while it is three for the MDPA. Therefore this mechanism is not friendship efficient. The superiority of the MDPA over this mechanism is due to the presence of waiting pairs in the MDPA.

This procedure is also not strategy-proof. For instance, assume that agent 18 has higher priority than 17 in the fixed ordering over agents (used to break ties when agents have the same good sets). Then agent 18 by misreporting $\{a, b\}$ as her set of good projects will be assigned a in this mechanism.

5.2 A maximum demand fixed priority mechanism

This mechanism can be thought of as a dual to the MDPA. Let S and A' denote the set of agents in $\tilde{F}_q$ and the set of available projects for the component. Let $S_1, \ldots, S_T$ and

$A_1, \dots, A_T$ denote the partitions of S and A' constructed in the proof of Theorem 3. Specifically A_1 is the set of projects which are good for all agents in S; S_1 is the set of agents whose available good project set is exactly A_1; A_2 is the set of projects which are good for all agents in $S \setminus S_1$; S_2 is the set of agents whose available good project set is exactly $A_1 \cup A_2$ etc. Thus S_1 is the set of most fussy agents in the sense that they like only projects in A_1, S_2 is the set of the next most fussy agents who like only projects in $A_1 \cup A_2$ etc. In the MDPA, assignment begins with the set (A_T, S_T). If some agents cannot be assigned a good project, they are then added to agents in S_{T-1} and they jointly demand A_{T-1}. If there are surplus projects in A_T, they are discarded. The algorithm then progressively assigns projects in (A_{T-1}, S_{T-1}) to (A_1, S_1) where unassigned agents in each stage are transferred to the next stage while the surplus projects at every stage are discarded.

In the algorithm under consideration, assignment begins with projects in A_1 and agents in S_1. If $|S_1| > 2|A_1|$, then these surplus agents are made to wait while the remaining $2|A_1|$ agents are paired and assigned projects using fixed priority and tie breaking rules.[10] If $|S_1| \leq 2|A_1|$, then $\frac{|S_1|}{2}$ pairs are formed and assigned projects according to a fixed tie breaking rule. Note that if $|S_1|$ is odd, then there is a remaining agent who can be assigned a project from A_1, but no partner from $|S_1|$. In this case the remaining agent is assigned a partner from S_2. Two adjustments are made for Step 2:(i) If there is an agent who has been pulled out from S_2 to be matched with an agent in S_1, this agent is removed from S_2.[11] and (ii) the projects available for assignment in Step 2 is $A_2 \cup R_1$, where R_1 denotes the surplus projects in Step 1 after the assignments have been made in Step 1.[12] Then these projects (projects in A_2 and the surplus projects from A_1) are assigned to remaining agents in S_2 in the same manner as before if S_2 is non empty. In case S_2 is empty, proceed to S_3 and add these projects to A_3. Again projects are assigned to agents in S_3 as in the first Step.

Thus for any Step $2 \leq q \leq T$, we assign the available projects in $A_q \cup R_{q-1}$ to the remaining agents in S_q. This process continues untill the stage (A_T, S_T) is reached. There are three possibilities.

1. S_T is empty (i.e. no agents are remaining in S_T when Step T is reached) and all agents in F_q have been assigned a partner and a project. Then the procedure terminates here.

2. S_T is empty but there are some waiting agents.[13] The waiting agents are divided into disjoint pairs and each pair is assigned a project from the available set.

[10] Note that these surplus agents will not be assigned a good project in the procedure as all their good projects are contained in the set S_1.

[11] There is exactly one such agent, who is removed from S_2 to be matched with the remaining "odd" agent in S_1.

[12] Note that $R_1 = \emptyset$ if $|S_1| \geq 2|A_1|$.

[13] Note that these are agents who were surlpus in some Step $1 \leq q \leq T$. Also there will be an even number of waiting agents.

F_1
4 $\{x, y\}$
3 $\{x, y\}$
2 $\{x, y\}$
1 $\{x, y\}$
6 $\{x\}$
5 $\{x\}$

Table 8: Example 7

3. S_T is non empty. Assign projects to agents in S_T in the same manner as before. Note if $|S_T|$ is odd, then there will exist an odd number of waiting agents. Thus the remaining agent in S_T (who cannot find a partner within S_T) is assigned a partner from the waiting set of agents and a project from either $A_T \cup R_{T-1}$ (if $|A_T \cup R_{T-1}| \geq |S_T|$) or an arbitrary project from the set of available projects. The waiting agents are divided into disjoint pairs and each pair is assigned a project from the available set.

This algorithm is not strategy-proof, as shown by the next example.

Example **7** Let $N = \{1, 2, 3, 4, 5, 6\}$ and $A = \{x, y, z, q\}$. All agents belong to the same component. Table 8 provides the sets of good projects of all the agents.

The partitions of A and N are: $A_1 = \{x\}$, $S_1 = \{5, 6\}$ and $A_2 = \{y\}$, $S_2 = \{1, 2, 3, 4\}$. The mechanism generates the assignment $(5, 6, x)$, $(1, 2, y)$ and $(3, 4, z)$. Let $1 \succ^N 2 \succ^N 3 \succ^N 4 \succ^N 5 \succ^N 6$ be the fixed priority over the agents. Agent 3 can manipulate by announcing $\{x\}$ as her set of good projects. Then the mechanism will form $(3, 5, x)$ and agent 3 will be strictly better off.

5.3 A fixed priority algorithm

Let $\tilde{F}_q$ be a component with an even number of agents. Define a priority over the agents in $\tilde{F}_q$. Label agents in $\tilde{F}_q$ as $\{1, 2, \ldots, 2p-1, 2p\}$ according to the priority i.e. agent 1 has higher priority than 2 who has higher priority than 3 and so on. The agents 1 and 2 are paired together and assigned a project in the following manner. If there exists a project which is good for both agents and is available, this project is assigned to $(1, 2)$. In case there does not exist any available project which is good for both agents, assign a good project for the higher priority agent in the pair (if possible) or assign a good project for the lower priority agent in the pair. Otherwise assign an arbitrary project to the pair $(1, 2)$. Proceed to the next two agents, 3 and 4 and assign a project to the pair $(3, 4)$ using the same principle as

F_1
1 $\{x, y, z\}$
2 $\{x\}$
3 $\{x\}$
4 $\{x, y, z\}$
5 $\{x, y\}$
6 $\{x, y\}$

Table 9: Example 8

above. We continue to form pairs of consecutive agents and assigning them a project from the set of available projects.

This mechanism is strategy-proof. However it is not friendship efficient.

Example **8** Let $N = \{1, 2, 3, 4, 5, 6\}$ and $A = \{x, y, z, e, f, g\}$. All agents belong to the same friendship component F_1. The set of acceptable projects are: $G^1 = \{x, y, z\}$, $G^2 = G^3 = \{x\}$, $G^4 = \{x, y, z\}$ and $G^5 = G^6 = \{x, y\}$.

Suppose the priority over the agents $\succ^N$ is $1 \succ^N 2 \succ^N 3 \succ^N 4 \succ^N 5 \succ^N 6$. The FPA algorithm forms one of the following assignments,

1. $\{(1, 2, x), (3, 4, y), (5, 6, a)\}$ where $a \in \{z, e, f, g\}$.

2. $\{(1, 2, x), (3, 4, z), (5, 6, b)\}$ where $b \in \{y, e, f, g\}$.

We claim that none of these assignments satisfy friendship efficiency. In both assignments, there is at least one unhappy agent i.e. agent 3. Consider the MDPA assignment $(1, 4, z)$, $(5, 6, y)$ and $(2, 3, x)$ where the number of unhappy agents is zero.

6 Alternative Domains

In this section we show that stable assignments may not exist if either the homophily or the dichotomous preferences assumptions are dispensed with.

6.1 Dichotomous Domains without homophily

We retain all the assumptions on preferences in Section 2.1 except homophily. In Example 9, we show the non-existence of stable assignments for a partner dominant preference profile. Similarly Example 10 shows that non-existence of stable assignments for a project dominant preference profile.

F_1	F_2
1 $\{a, b\}$	4 $\{g\}$
2 $\{c, d\}$	
3 $\{e, f\}$	

Table 10: Example 9

F_1	F_2
1 $\{a, b\}$	4 $\{g, h\}$
2 $\{c, d\}$	5 $\{x, y\}$
3 $\{e, f\}$	6 $\{z, u\}$

Table 11: Proposition 3

Example **9** Let $N = \{1, 2, 3, 4\}$ and $A = \{a, b, c, d, e, f, g\}$. The friendship components are $F_1 = \{1, 2, 3\}$ and $F_2 = \{4\}$. The good sets of agents are described in Table 10. Note that homophily is violated in F_1.

Let $\succ$ be the preference profile where all agent preferences are partner dominant and the good partner, project sets are specified above.

In any assignment, one of the agents in F_1 must be matched with agent 4, while the remaining agents are matched with each other. Assume w.l.o.g that $(1, 2)$ and $(3, 4)$ are matched with each other. Since $\succ_3$ is partner dominant, agent 3 would like to break with agent 4 and be paired with either 1 or 2, even with a bad project. Since the good sets of agents 1 and 2 are disjoint, at least one of them must be receiving a bad project. Moreover this agent must also have a project in her good set which is unassigned. Agent 3 can then pair with this agent and this unassigned project to block the assignment. Hence a stable assignment does not exist.

Example **10** Let $N = \{1, 2, 3, 4, 5, 6\}$ and $A = \{a, b, c, d, e, f, g, h, x, y, z, u\}$. The friendship components are $F_1 = \{1, 2, 3\}$ and $F_2 = \{4, 5, 6\}$. Table 11 specifies the set of good projects for all agents. Homophily is violated for both components. Let $\succ$ be the preference profile where each agent's preference is project dominant and the good partner, project sets are specified above.

Since there are an odd number of agents in each component, at least one agent in F_1 is matched to an agent in F_2 in any assignment . Assume w.l.o.g that 3 and 6 are paired together. Since the good projects of agents 3 and 6 are disjoint, at least one of them is not getting a good project. Suppose it is agent 3.

In order for agent 3 not to form a blocking coalition with either agents 1 or 2 via an unassigned project either (i) or (ii) below must hold: (i) all good projects of 1 and 2 are assigned or (ii) agents 1 and 2 are both getting good projects. However there are four projects that are either good for agent 1 or good for agent 2, while the total number of assigned projects is three. Therefore (i) cannot hold.

Suppose (ii) holds. Then agents 1 and 2 must be paired with agents 4 and 5 in F_2. Moreover both agents 1 and 2 must be getting a good project. Therefore agents 4 and 5 are not getting good projects i.e. they are getting bad partners and bad projects. They will then block by pairing with each other with an unassigned project.

Thus a stable assignment does not exist.

6.2 Non Dichotomous Preferences

In this subsection, we illustrate the importance of the dichotomous domain assumption in our model. We assume that preferences over partner, project pairs continue to be separable i.e. "marginal" preferences over both components can be defined unambiguously. These component preferences are non-dichotomous but are well-behaved in the sense that they are single-peaked. Nevertheless stable assignments may fail to exist according to Proposition 4.

Let $>_o$ and $>_p$ be strict orderings over A and N respectively. Agent i has an anti-symmetric preference ordering over projects, $\succ_i^{proj}$ which is defined as follows: she has a unique best project denoted by $\tau(i)$ and for all $a, b \in A$,

$$\begin{matrix} b >_o a >_o \tau(i) \\ \tau(i) >_o b >_o a \end{matrix} \Rightarrow b \succ_i^{proj} a$$

Similarly agent i has an anti-symmetric preference order over partners denoted by $\succ_i^{part}$ which are defined as follows: she has a unique best partner denoted by $\pi(i)$ and for all $j, k \in N \setminus \{i\}$,

$$\begin{matrix} j >_p k >_p \pi(i) \\ \pi(i) >_p j >_p k \end{matrix} \Rightarrow j \succ_i^{part} k$$

These component preferences are combined lexicographically to define preferences over partner, project pairs. Agent i's preferences $\succ_i$ are project dominant if

$$(j, a) \succ_i (k, b) \text{ if either } [a \succ_i^{proj} b] \text{ or } [a = b \text{ and } j \succ_i^{part} k]$$

Similarly agent i's preferences $\succ_i$ are partner dominant if

$$(j, a) \succ_i (k, b) \text{ if either } [j \succ_i^{part} k] \text{ or } [j = k \text{ and } a \succ_i^{proj} b]$$

Let $D^1(N,A)$ and $D^2(N,A)$ denote the set of all such project dominant and partner dominant preferences respectively.

Note that these preferences are multi-dimensional single peaked as defined by Barberà et al. (1993).

Proposition 4 *There exist N and A for which stable assignments do not exist when preferences of all agents belong to $D^1(N,A)$. Similarly there exists N and A for which stable assignments do not exist when preferences of all agents belong to $D^2(N,A)$.*

The proof of Proposition 4 is constructive and is provided in the Appendix.

7 Conclusion

In this paper, we have investigated a class of matching models where agents have to be matched in pairs with a project. We provide a domain restriction on partner, project pairs that guarantees the existence of a robustly stable assignment. We provide an algorithm, the MDPA algorithm which generates a robustly stable assignment at every preference profile. It satisfies efficiency within every friendship component. In addition it is strategy-proof.

In future, we hope to extend our work to teams of general size. Our current results on pairs do not extend in a straightforward manner to the more general case. We also hope to investigate other notions of capturing homophily in project and team assignment models.

8 Appendix

8.1 Proofs

We provide the proofs of the major results in the paper.

Proof of Theorem 1:

Let $\succ$ be an arbitrary profile and σ be the assignment generated by the MDPA algorithm at $\succ$. We will show that σ cannot be blocked via an unassigned project, by a position swap or by a project swap at any $\succ'$ which is good set equivalent to $\succ$.

An important initial observation is that the algorithm relies only on the $P^i(\succ_i)$,$G^i(\succ_i)$ sets for all agents. Therefore if two profiles $\succ, \succ'$ induce the same good sets for all agents, the algorithm generates the same assignment.

Blocking via unassigned projects: Suppose σ is blocked via an unassigned project, i.e. there exist $(k,i,a),(l,j,b) \in \sigma$ and $b \in u^\sigma$ such that $(k,l,b) \succ'_k (k,i,a)$ and $(k,l,b) \succ'_l (l,j,c)$ where $\succ'$ is good set equivalent to $\succ$.

We consider the following exhaustive possibilities.

(i) The agents $k, l \in F_q$ for some q.

There are two subcases to consider.

(a) Agents k, l are paired together in σi.e. $(k, l, a) \in \sigma$. Since k and l strictly improve, we have $a \notin G^k(\succ'_k)$, $a \notin G^l(\succ'_l)$ and $b \in G^k(\succ'_k) \cap G^l(\succ'_l)$.[14] In the algorithm, it must have been the case that (k, l, a) was formed in Step $q.T$ (the termination step for the component $\tilde{F}_q$) Also the demand for each available project in this step must have been zero. Since $b \in u^\sigma$, b is available in Step $q.T$ and $D(b; (q.T)) = 2$ in Step $q.T$. This results in a contradiction.

(b) Agents k, l are not paired together in σi.e. $(k, i, a), (l, j, c) \in \sigma$ where $i \neq l$ and $j \neq k$. In the algorithm, there is atmost one residual agent from each friendship component. This implies that at leastone of the agents among k and l is paired with a friend. Assume w.l.o.g $k, i \in F_q$. Since k is strictly improving, we have $a \notin G^k(\succ'_k)$ and $b \in G^k(\succ'_k)$. In the algorithm, it must have been the case that agent k is assigned a partner and a project in the termination step for $\tilde{F}_q$ where the demand for each available project was zero. However b was available in this step by assumption. This is a contradiction.

(ii) Agents $k, l \notin F_q$ for any q.

Suppose $k \in F_q$ and $l \in F_p$. There are three possibilities to consider.

(a) $\succ'_k$ and $\succ'_l$ are both partner dominant. Since k and l are strictly improving, we must have $i \notin F_q$ and $j \notin F_p$. Thus k, i, l, j all belong to the residual set R. Also we have $a \notin G^k(\succ'_k)$, $b \in G^k(\succ'_k)$, $c \notin G^l(\succ'_l)$ and $b \in G^l(\succ'_l)$. Since k and l are being assigned bad projects in σ, we can conclude that (k, i, a) and (l, j, c) are being formed in Step $L + 2$. Suppose agent $k \succ^N l$. Since b is unassigned, there will exist a substep of Step $L + 1$, say $L + 1.r$ (with $r \leq r^*$) where k and l could have been paired together with project b which is good for both agents. Therefore we have a contradiction.

(b) $\succ'_k$ and $\succ'_l$ are both project dominant.
Suppose one of the pairs (k, i), (l, j) belongs to the same friendship component. W.l.o.g let $k, i \in F_q$ for some q. Since agent k strictly improves, we have $a \notin G^k(\succ'_k)$ and $b \in G^k(\succ'_k)$. We then have a contradiction exactly as in (i)(b).

Finally suppose neither k, i nor l, j belong to the same component. Thus k, i, l, j belong to the residual set. This reduces to (ii)(a) which we have already dealt with.

(c) One of $\succ'_k$ and $\succ'_l$ is partner dominant while the other is project dominant. Suppose $\succ'_k$ is partner dominant. Since agent k strictly improves, we conclude that i and k cannot belong to the same friendship component. Also $a \notin G^k(\succ'_k)$ and $b \in G^k(\succ'_k)$.

[14]This fact is independent of whether $\succ'_k$ is partner or project dominant.

There are two further possibilities. Suppose l, j do not belong to the same component. Then agents k, i, l, j are residual agents. Once again, we are back to (ii)(a). The remaining case is $l, j \in F_p$ for some p. Since l strictly improves, we have $c \notin G^l(\succ_l')$ and $b \in G^l(\succ_l')$. Note that this case is equivalent to (i)(b), where agent k is replaced by l and i is replaced by j.

Therefore σ cannot be blocked via an unassigned project.

Blocking via a position switch: Suppose σ can be blocked by a position swap and assume w.l.o.g (by suitable relabelling) that $(1, 2, a_1), (3, 4, a_2), \ldots, (2k-1, 2k, a_k) \in \sigma$ where the odd numbered agents are permuted by μ and strictly improve i.e. $\mu(1) = 3, \mu(3) = 5, \mu(s) = s+2$ (where s is odd), $\mu(2k-1) = 1$ and

$$\big(s+3, a_{\frac{s+3}{2}}\big) \succ_s \big(s+1, a_{\frac{s+1}{2}}\big) \text{ where } s \in \{1, 3, \ldots, 2k-1\}.$$

This is a cycle of length k. We can also assume w.l.o.g there does not exist a subcycle of σ where agents who are permuted strictly improve. Let $S = \{1, 3, \ldots, 2k-1\}$.

Assume $(s, s+1) \in F_q$ (where $s \in S$) for some q.[15] We claim that it cannot be the case that either $(s+2, s+3) \in F_p$ where $q < p$ or $(s+2, s+3) \in R$. Suppose (i, j) and (i', j') are two pairs in distinct friendship components and are assigned together in the MDPA. Suppose i takes the place of i' in the position swap. Then it cannot be the case that i strictly improves if the (i', j') belong to a friendship component where project assignment is made after that of the (i, j) component. Similarly i cannot improve if (i', j') belong to the residual set.

We now verify the claim. Let $(s+2, s+3) \in F_p$ and $q < p$. Since $\big(s+3, a_{\frac{s+3}{2}}\big) \succ_s \big(s+1, a_{\frac{s+1}{2}}\big)$, it cannot be the case that $\succ_s$ is partner dominant. Therefore $a_{\frac{s+1}{2}} \notin G^s$ and $a_{\frac{s+3}{2}} \in G^3$. According to the MDPA, $a_{\frac{s+3}{2}}$ is available when assignments are made in $\tilde{F}_q$ (in particular to agent s). Therefore s could not have got a bad project in σ. This results in a contradiction. By an identical argument, $(s+2, s+3) \notin R$.

Suppose there exists s (where $s \in S$) such that $(s, s+1) \in F_{q_s}$ for some q_s. By the argument in the earlier paragraph, $(s+2, s+3) \in F_{q_{s+2}}$ where $q_{s+2} \leq q_s$. Proceeding in this way, we obtain a sequence of q_{s+r}, $r \in \{0, 2, \ldots, 2k\}$ such that $q_{s+2k} = q_s$ and $q_{s+r} \leq q_{s+r+2}$ for all $r \in \{0, \ldots, 2k\}$. Clearly we must have $q_s = q_{s+r}$ for all r i.e. all agents in $\{1, 2, \ldots, 2k\}$ belong to the same component F_{q_s}.

We shall refer to this possibility as Case A. The only remaining possibility is $(s, s+1) \in R$ for all s, which we shall refer to as Case B. We will show that both Cases A and B lead to contradictions.

Case A: By assumption, $(4, a_1) \succ_1 (2, a_1)$, $(6, a_3) \succ_3 (4, a_2)$, $\ldots$, $(2, a_1) \succ_{2k-1} (k, a_k)$. Since all agents belong to the same component, the preference relations above imply $a_1 \notin G^1, a_2 \in G^1; a_2 \notin G^3, a_3 \in G^3; \ldots; a_k \notin G^{2k-1}, a_1 \in G^{2k-1}$. Since $a_2 \in G^1$ and $a_2 \notin G^3$, homophily

[15]We adopt the convention that $2k - 1 + 2 \equiv 1$.

implies $G^1 \supset G^3$.[16] By a similar argument, $G^3 \supset G^5$ etc. So $G^1 \supset G^3 \dots G^{2k-1} \supset G^1$ which is impossible.

Case B: All the agents are in R i.e. they belong to different friendship components.

Since all agents in the set S strictly improve by the swap, the assignments for these agents in σ are made in Step $L+2$.[17] Recall in Step $L+2$, $\bar{R}^1$ is the set of agents who have no good projects available and $\bar{R}^2$ are those that have. Since the agents in S are strictly improving, they must belong to $\bar{R}^2$. Moreover their partners in σ also belong to $\bar{R}^2$; otherwise some of the agents in S would not be able to strictly improve.

Now consider $(1, 2, a_1), (3, 4, a_2) \in \sigma$. We have argued that $a_1 \notin G^1$, $a_2 \in G^1$ and $a_2 \notin G^3$. By the MDPA, $4 \succ^N 3$ and $a_2 \in G^4$. But then $a_2 \in G^1 \cap G^4$ which is a contradiction to the assumption that 1 and 4 were not assigned a good project in Step $L+1$.

Blocking via a project swap: Suppose σ can be blocked by project swaps and assume w.l.o.g (by suitable relabelling) that $(1, 2, a_1), (3, 4, a_2), \dots, (2k-1, 2k, a_k) \in \sigma$ where all agents strictly improve by a swap of projects. Let

Assume $\mu(a_1) = a_2, \mu(a_2) = a_3, \dots, \mu(a_k) = a_1$. Furthermore these swaps are strictly improving for all agents i.e.

$$\left(s+1, a_{\frac{s-1}{2}}\right) \succ_s \left(s+1, a_{\frac{s+1}{2}}\right) \text{ and } \left(s, a_{\frac{s-1}{2}}\right) \succ_{s+1} \left(s, a_{\frac{s+1}{2}}\right)$$

where $s \in \{1, 3, \dots, 2k-1\}$ and $a_0 \equiv a_k$.

By using the same arguments as in case of the position swaps, we can conclude the following: if $(s, s+1)$ belong to some friendship component F_q then $(s-2, s-1)$ cannot be members of a friendship component which is assigned projects after F_q. Nor can $(s-1, s-2)$ belong to R. Once again using identical arguments to those used earlier, only one of two possibilties can arise: Case A' where all agents in $\{1, 2, \dots, 2k\}$ belong to the same component F_q and Case B' where all agents in $\{1, 2, \dots, 2k\}$ belong to distinct components.

Case A': Consider the agents in $\{1, 3, \dots, 2k-1\}$. By assumption, $(2, a_k) \succ_1 (2, a_1), (4, a_1) \succ_3 (4, a_2), (6, a_2) \succ_5 (6, a_3), \dots, (2k, a_{k-1}) \succ_{2k-1} (2k, a_k)$. Thus $a_1 \notin G^1$ and $a_1 \in G^3$. Homophily implies $G^1 \subset G^3$.[18] Similarly $a_2 \notin G^3$ and $a_2 \in G^5$ implies $G^3 \subset G^5$ by homophily. Proceeding in the same way, $G^1 \subset G^3 \subset G^5 \dots G^{2k-1}$. Hence $G^1 \subset G^{2k-1}$. However $a_k \notin G^{2k-1}$ and $a_k \in G^1$, we also have $G^{2k-1} \subset G^1$. We have a contradiction.

Case B': Since both agents in the pair $(s, s+1)$, $s \in \{1, 3, \dots, 2k-1\}$ are strictly improving, it must be the case that all the pairs are formed and assigned projects in Step $L+2$. However after swapping projects, both agents in every pair get a good project. Let s be the highest

16 Note that this is a strict superset.

17 In Step $L+1$, any pair assigned a project has the property that the project is good for both agents in the pair.

18 Note that this is a strict subset.

priority agent in $\{1, 2, \ldots, 2k\}$. All projects in $\{a_1, \ldots, a_k\}$ are available when agent s is being considered for a partner, project assignment in Step $L+1.q$ for some q. Since agent $s+1$ and all projects in $\{a_1, \ldots, a_k\}$ are available in Step $L+1.q$, the MDPA must assign a good project to s in Step $L+1.q$. This contradicts the assumption that s strictly improves.

This completes the proof of the result.

Proof of Theorem 2: Let i be an agent such that $i \in F_q$. Let $\succ_i$ be a preference ordering for agent i and $\succ$ be a preference profile. We will show that there does not exist $\succ_i'$ such that $\sigma(\succ_i', \succ_{-i}) \succ_i \sigma(\succ)$. Note that the outcome in the MDPA does not depend on whether $\succ_i$ is partner or project dominant. We therefore only need to show that i cannot benefit by misreporting $G^i(\succ_i)$. In view of the assumptions made on the good sets, this is equivalent to checking that i cannot benefit by changing her threshold project.

Suppose $(k, i, a) \in \sigma(\succ)$. There are two separate cases to consider: Case 1 where i is a residual agent and Case 2 where i is matched with a friend. We consider each case in turn.

Case 1: Note that a misreport by agent i will not change his residual status as the MDPA algorithm always picks the minimum agent in F_q according to $\succ^N$. Suppose there exists a misreport $\succ_i'$ such that i strictly improves. In both cases, i is not matched with a friend; therefore $a \notin G^i(\succ_i)$ while the misreport yields a project $b \in G^i(\succ_i)$.

Since $a \notin G^i(\succ_i)$, it must be the case that i is assigned in Step $L+2$ for the profile $\succ$. Suppose b is assigned in Step $L+1$ and $(i, k', b) \in \sigma(\succ_i', \succ_{-i})$. Assume $k' \succ^N i$. It must be the case that the pair (k', i) was considered in $\sigma(\succ)$ in some substep of $L+1$; otherwise the pair would not be considered in $(\succ_i', \succ_{-i})$. Since $(k', i, b) \in \sigma(\succ_i', \succ_{-i})$ is formed in Step $L+1$, we have $b \in G^k(\succ_k)$. Also $b \in G^i(\succ_i)$ as agent i strictly improves by misreporting. So $b \in G^i(\succ_i) \cap G^{k'}(\succ_k)$ and i would have received a good project in $\sigma(\succ)$ which is a contradiction. Assume that i has a higher priority than k'. Then the pair (i, k') would have been considered in some substep in $L+1$ and i would have received a good project in $\sigma(\succ)$. We therefore conclude that (k', i, b) could not have been formed in Step $L+1$ in $(\succ_i', \succ_{-i})$ i.e. b is assigned to (k', i) in some substep of $L+2$.

It is clear that the set of unassigned agents and available projects in Step $L+2$ is the same in $\succ$ and $(\succ_i', \succ_{-i})$. Moreover the sets $\bar{R}^1$ and $\bar{R}^2$ are the same in both cases. Since b is available in this step, $i \in \bar{R}^2$. If $k' \in \bar{R}^1$, then $(i, k', b) \in \sigma(\succ)$ which is a contradiction. Finally if $k' \in \bar{R}^2$, i and k must be consecutive in priority in $\bar{R}^2$ which implies that $(i, k', b) \in \sigma(\succ)$. This is a contradiction.

Case 2: In this case, $k, i \in F_q$. Agent i will continue to be matched to another agent in F_q for any misreport. Assume without loss of generality that the set of available projects in F_q is $\{a_1, a_2, \ldots, a_T\}$ and $a_T \succ_q^O a_{T-1} \succ_q^O \ldots \succ_q^O a_1$.

Consider an arbitrary good set profile $\{G^j(\succ_j)\}$, $j \in \tilde{F}_q$. As described earlier, it is determined by a profile of threshold projects, $\{a^j\}$, $j \in F_q$. This profile generates a demand vector $D(a_t)$, $t = 1, \ldots, T$. By assumption, the good project profile is homophily consistent and $D(a_1) \geq D(a_2) \geq \ldots \geq D(a_T)$. The profile also specifies the sets of agents who demand each project $S(a_1), S(a_2), \ldots S(a_T)$.

According to the MDPA and as a consequence of homophily (and the tie breaking assumption), projects are considered in the sequence $a_T, a_{T-1}, \ldots, a_2, a_1$. Abusing notation slightly, let $D^r(a_t)$ denote the demand for project a_t when a_r is considered by the algorithm i.e. a_r has the least demand and is chosen after tie breaking. Thus $D^T(a_t) = D(a_t)$ in the demand vector in the first step when project a_T is being considered. When a_r is being considered, all projects a_t, $t > r$ have already been considered. Note that these projects have not necessarily been allocated to some pair; however for the purpose of the algorithm we can regard $D^r(a_t) = 0$ whenever $t > r$. Let $S^r(a_t)$, $t \leq r$ denote the set of agents who have positive demand for project a_t.

Suppose i's threshold project in the profile $\{G^j(\succ_j)\}$, $j \in F_q$ is a_k, i.e. she has positive demand only for projects a_t where $t \geq k$. We begin with an important observation. Agent i can manipulate only if she does not receive a good project when being truthful. This implies that $D^t(a_t) \geq 3$ for all $t \leq k$. Moreover there are at least two agents in each set $S^t(a_t)$ who are ahead of i in the order $\succ^N$. Let a_t be a project with $t \leq k$. If i has been allocated a project or is waiting when a_t is being considered, she must be receiving or will receive an acceptable project. Therefore it must be the case that $i \in S^t(a_t)$. If $D^t(a_t) = 1$, agent i will be a waiting pair and by construction of the algorithm, must receive an acceptable project. If $D^t(a_t) = 2$, i gets a_t. Hence $D^t(a_t) \geq 3$ so that project a_t must be allocated when it is considered. There must therefore be two agents who are ahead of i in the numerical order in the set $S^t(a_t)$ who are allocated a_t.

Agent i can misreport in only one of two ways: (i) by announcing a threshold a_m where $T \geq m > k$, i.e by expanding the set of good projects and (ii) by announcing a threshold a_m where $1 \leq m < k$, i.e. by contracting the set of good projects.

We consider case (i) first. Let $\hat{D}^r(a_t)$, $t \leq r$ denote the demand of projects when project a_r is being considered, i.e. these are the demands in the various stages of MDPA, when MDPA is run on the profile where i misreports. Similarly $\hat{S}^r(a_t)$ denotes the set of agents who demand a_t at the misreported profile. We will track the project received by i in this profile and show that it cannot belong to the set $\{a_1, a_2, \ldots, a_k\}$.

Consider projects a_r where $r > m$. For such projects, $\hat{D}^r(a_r) = D^r(a_r)$ and $\hat{S}^r(a_r) = S^r(a_r)$. Hence these projects if allocated, are allocated to the same agents in the truthful and misreported profiles. In addition, any agent, project pair who is waiting in one of the profiles is also waiting in the other.

We now turn to the case where a_m is being considered in the misreported profile. By assumption, $i \in \hat{S}(a_m)$. Let $\hat{D}^m_{-i}(a_m) = \hat{D}^m(a_m) - 1$ and $\hat{S}^m_{-i}(a_m) = \hat{S}^m(a_m) \setminus \{i\}$. Note that $D^m(a_m) = \hat{D}^m_{-i}(a_m)$ and $S^m(a_m) = \hat{S}^m_{-i}(a_m)$ by virtue of the argument in the previous paragraph.

There are several cases to consider at this point. If $\hat{D}^m_{-i}(a_m) = 0$, (a_m, i) is the waiting pair at this step. If there already exists a waiting pair, then i is allocated a_m which is, by assumption, a bad project. Otherwise (a_m, i) is a waiting pair and will be assigned a project later. Note that $\hat{D}^{m-1}_{-i}(a_{m-1}) = D^{m-1}(a_{m-1})$ and $\hat{S}^{m-1}_{-i}(a_{m-1}) = S^{m-1}(a_{m-1})$.

Suppose $\hat{D}^m_{-i}(a_m) = 1$ or $\hat{D}^m_{-i}(a_m) \geq 2$ and i is one of the two highest priority agents in the set $\hat{S}^m(a_m)$. Then i is allocated a_m which is a bad project. Otherwise, since $S^m(a_m) = \hat{S}^m_{-i}(a_m)$, a_m will be allocated to the same two agents who were assigned a_m in the truthful profile. Once again, $\hat{D}^{m-1}_{-i}(a_{m-1}) = D^{m-1}(a_{m-1})$ and $\hat{S}^{m-1}_{-i}(a_{m-1}) = S^{m-1}(a_{m-1})$.

We can therefore conclude the following: if i is assigned a project at this step, it must be to a bad one; otherwise $\hat{D}^{m-1}_{-i}(a_{m-1}) = D^{m-1}(a_{m-1})$ and $\hat{S}^{m-1}_{-i}(a_{m-1}) = S^{m-1}(a_{m-1})$.

Suppose i is not assigned a project at Step $q.a_m$.[19] Consider Step $q.a_{m-1}$. Suppose $i \notin S(a_{m-1})$i.e. $m-1 > k$ and a_{m-1} is not a good project for i. If $\hat{D}^{m-1}_{-i}(a_{m-1}) = 0$, i remains waiting with a_m. If $\hat{D}^{m-1}_{-i}(a_{m-1}) = 1$, i is allocated a_{m-1} with the other demander for a_{m-1}. Since a_{m-1} is not a good project for i, misreporting is not beneficial. Similarly $\hat{D}^{m-1}_{-i}(a_{m-1}) \geq 2$ and i is one of the two highest priority agents in $\hat{S}^{m-1}(a_{m-1})$; i is assigned a_{m-1}. The only remaining case is when i is not among the two highest agents $\hat{S}^{m-1}(a_{m-1})$ according to the numerical order. In this case, $\hat{S}^{m-1}_{-i}(a_{m-1}) = S^{m-1}(a_{m-1})$ implies that the same pair of agents are allocated a_{m-1} in the misreported profile as in the truthful profile.

Summarizing, we have reached the same conclusion in stage a_{m-1} as in stage a_m. Suppose a_{m-1} is not acceptable project for i. If i is allocated a project in stage a_{m-1}, it must be to an unacceptable project. If i has not been allocated a project, it must be true that $\hat{D}^{m-2}_{-i}(a_{m-2}) = D^{m-2}(a_{m-2})$ and $\hat{S}^{m-2}_{-i}(a_{m-2}) = S^{m-2}(a_{m-2})$.

In fact, the same argument can be replicated for all stages a_{m-t}, $t = 0, \ldots, m-k$ till one reaches a_k where the project being considered by the algorithm in the misreported profile is an acceptable project of i. In particular, we can conclude that either i is allocated an unacceptable project or the algorithm reaches stage a_k with $\hat{D}^k_{-i}(a_k) = D^k(a_k)$ and $\hat{S}^k_{-i}(a_k) = S^k(a_k)$.

At Step $q.a_k$, we have already argued that $D^k(a_k) \geq 3$ and there are two agents other than i in $S^k(a_k)$ who are ahead of i in the priority order $\succ^N$. Therefore $\hat{D}^k_{-i}(a_k) \geq 2$. Moreover $\hat{S}^k_{-i}(a_k) = S^k(a_k)$ implies that the same pair of agents who were assigned a_k in the truthful profile are also allocated a_k in the misreported profile. The same argument can be applied repeatedly to show that if i has not been assigned at Step $q.a_{k+1}$, she cannot be assigned any of the projects $a_1, \ldots a_{k-1}, a_k$. The algorithm then assigns a bad project to i. This can happen in one of two ways. If i is waiting with a project a_l, $k < l \leq T$, after a_1 has been allocated, then i will be allocated a_l. Otherwise i will receive an unacceptable project after allocation of acceptable projects in F_q has been completed. In any case, manipulation does

[19]Step $q.a_m$ denotes the step in which project a_m is the least demanded project which is chosen after tie breaking.

not succeed.

Finally consider case (ii) where i misreports by contracting her good set. Following the earlier argument, it is clear that assignment proceeds in exactly the same manner in the misreported profile as in the truthful profile until Step $q.a_k$ is reached. This implies that the same agents who were assigned projects $a_1, \dots a_k$ are not assigned in Steps $q.a_{k+1}$ and earlier. Hence at each Step $q.a_t$, $t = k, k-1, \dots, 1$, $\hat{D}_{-i}(a_t) \geq 2$ and $\hat{S}_{-i}(a_t)$ contains two agents with higher priority than i. Therefore these agents are assigned $a_1, \dots, a_k$ and i does not receive a good project. This completes the proof.

Proof of Theorem 3: Consider a component $\tilde{F}_q$ and a preference profile $\succ$. Let A' be the set of projects available for assignment to $\tilde{F}_q$ in the MDPA. For convenience, we will denote $\tilde{F}_q$ by S. Also with a minor abuse of notation, we will write $G^i(\succ_i) \cap A'$ simply as $G^i(\succ_i)$. Construct pairs of sets $(S_1, A_1), \dots, (S_T, A_T)$ as follows:

(1.a) $A_1 = \{a \in A' : a \in G^i(\succ_i) \text{ for all } i \in S\}$.

(1.b) $S_1 = \{i \in S : G^i(\succ_i) = A_1\}$.

(2.a) $A_2 = \{a \in A' \setminus A_1 : a \in G^i(\succ_i) \text{ for all } i \in S \setminus S_1\}$.

(2.b) $S_2 = \{i \in S \setminus S_1 : G^i(\succ_i) = A_1 \cup A_2\}$.

$\vdots$

$\vdots$

(t.a) $A_t = \{i \in A \setminus (A_1 \cup \ldots \cup A_{t-1}) : a \in G^i(\succ_i) \text{ for all } i \in S \setminus (S_1 \ldots \cup S_{t-1})$.

(t.b) $S_t = \{i \in S \setminus (S_1 \cup \ldots \cup S_{t-1}) : G^i(\succ_i) = A_1 \cup \ldots \cup A_{t-1}\}$.

$\vdots$

The set A_1 consists of the available projects that are in the good sets of all agents in S. The set S_1 consists of those agents whose good project set is exactly the set A_1. Either $S_1 = S$ or there exist some agents who have good projects not belonging to A_1. Let A_2 be the set of projects which are in the good sets of all agents in $S \setminus S_1$. Similarly S_2 are the agents whose good projects is exactly the set $A_1 \cup A_2$. Note that $A_1 \cap A_2 = \emptyset$ and $S_1 \cap S_2 = \emptyset$. Proceeding in this manner, we obtain sets $(A_1, S_1), \dots, (A_T, S_T)$ where $A_1, \dots, A_T$ and $S_1, \dots, S_T$ are partitions of A' and S respectively. Notice that this procedure is well defined due to the assumption of homophily. The set S_T is the set of least fussy agents i.e. agents with the largest good sets (amongst available projects) while the set A_T consists of projects that are in the good set only of the agents in S_T. By homophily, agents in S_T like all the projects in

$\cup_{t=1}^{T} A_t$. In general, for any $i \in S_t$, $G^i(\succ_i) = \cup_{r=1}^{t} A_r$. Also the projects in $\cup_{r=t+1}^{T} A_r$ are not good projects for any agent $i \in S_t$.

The sets $(A_1, S_1), \dots, (A_T, S_T)$ can be used to calculate the happiness index of the MDPA. The algorithm begins by considering projects with the least demand amongst the available projects i.e. by assigning projects in A_T to agents in S_T. If $2|A_T| \geq |S_T|$, all agents in S_T are assigned projects in A_T using tie breaking. In this case, all agents in S_T get a good project. This includes the case where one of the agents in S_T is a waiting pair with one of the projects in A_T. Note that in the MDPA the waiting agent is guaranteed a good project; therefore in the computation of the happiness index we can assume without loss of generality that all agents in S_T are assigned a good project. On the other hand if $2|A_T| < |S_T|$, all projects in A_T are assigned to agents in S_T and the number of happy agents is $|S_T|$. In conclusion, the number of happy agents in Step T is $\min\{2|A_T|, |S_T|\}$. Let U_T denote the unhappy agents in this step. The number of unhappy agents in this step is $|U_T| = \max\{|S_T| - 2|A_T|, 0\}$.

In Step $T-1$, projects in A_{T-1} are allocated to agents in $\tilde{S}_{T-1}$ where $\tilde{S}_{T-1} = S_{T-1} \cup U_T$. Replicating the earlier arguments, the number of unhappy agents at the end of this step is $\max\{|\tilde{S}_{T-1}| - 2|A_{T-1}|, 0\}$. Moreover the number of unhappy agents in $\tilde{F}_q$ from the MDPA is $|U_1| = \max\{|\tilde{S}_1| - 2|A_1|, 0\}$. We will complete the proof by showing that no component allocation procedure can lead to a strictly smaller number of unhappy people.

Suppose the MDPA generates some unhappy agents. Let t^* be the highest value of t, $t \in \{1, \dots, T\}$ such that there exists an unhappy agent in S_t: in other words, agents in $\cup_{t=t^*+1}^{T} S_t$ are all assigned good projects. Using the arguments in the previous paragraph, it follows that

$$2|A_t| \leq |\tilde{S}_t| \text{ for all } t^* \leq t \leq 1 \text{ with } 2|A_{t^*}| < |\tilde{S}_{t^*}| \tag{1}$$

Hence the number of happy agents in stage t, $t \in \{1, \dots, t^*\}$ is $\min\{2|A_t|, |\tilde{S}_t|\}$. Since $\sum_{t=1}^{t^*} \tilde{S}_t = \sum_{t=1}^{t^*} S_t$, the total number of unhappy agents generated by the MDPA is $\sum_{t=1}^{t^*}[|S_t| - 2|A_t|]$.

Consider an arbitrary component allocation rule and further consider the allocation of projects to agents in $\cup_{t=1}^{t^*} S_t$. By construction, the good projects of agents in $\cup_{t=1}^{t^*} S_t$ must belong to the set $\cup_{t=1}^{t^*} A_t$. By Equation 1, $2\left|\cup_{t=1}^{t^*} A_t\right| < \left|\cup_{t=1}^{t^*} S_t\right|$. Therefore the set of unhappy agents amongst the set of agents $\cup_{t=1}^{t^*} S_t$ must at least be $\sum_{t=1}^{t^*}[|S_t| - 2|A_t|]$ which is also a lower bound for the total number of unhappy agents in $\tilde{F}_q$. But we have already argued that the MDPA achieves this bound for the number of unhappy agents. This completes the proof.

Proof of Proposition 4: We first consider the case when the preferences of all agents belong to $D^1(N, A)$. It will suffice to construct an example for this purpose. Let $N = \{1, 2, 3, 4, 5, 6\}$ and $A = \{a, b, c, d, e, f\}$. Let $a >_o b >_o c >_o d >_o e >_o f$ and $1 >_p 2 >_p 3 >_p 4 >_p 5 >_p 6$.

$\succ_1^{proj}$	$\succ_2^{proj}$	$\succ_3^{proj}$	$\succ_4^{proj}$	$\succ_5^{proj}$	$\succ_6^{proj}$
a	b	c	d	e	f
b	a	b	e	f	e
c	c	a	f	d	d
d	d	d	c	c	c
e	e	e	b	b	b
f	f	f	a	a	a

Table 12: $\succ_i^{proj}$

Table 12 specifies $\succ_i^{proj}$ for all agents i. Agents 2 and 3 are left-oriented i.e. any project which lies to the left of the best project of the agent is preferred to a project which lies to the right of the peak (according to $>_o$). Similarly agents 4 and 5 are right-oriented.

The preferences on partners for the agents are not important and therefore not specified.

Finally $\succ_i$ is project dominant for all ii.e. they belong to $D^1(N, A)$. We show below that a stable assignment does not exist at profile $\succ$.

Since there are six projects and six agents, exactly three projects must be assigned. Consider an arbitrary stable assignment.

We first show that at least one of the projects in $\{\tau(1), \tau(2)\}$ and one of the projects in $\{\tau(5), \tau(6)\}$ must be assigned.

Suppose neither $\tau(1)$ nor $\tau(2)$ are assigned. Then neither agents 1 nor 2 are getting their first and second ranked projects. Hence they can form a blocking coalition by choosing one of the unassigned projects $\tau(1)$ or $\tau(2)$. Note that this would give each agent either their first or second ranked project. By a similar argument, one of the projects in $\{\tau(5), \tau(6)\}$ must be assigned.

Suppose the third project assigned is the unassigned project in $\{\tau(1), \tau(2)\}$ i.e. the assigned projects are $\tau(1)$, $\tau(2)$, and either $\tau(5)$ or $\tau(6)$. Then at least one of the agents $i \in \{4, 5, 6\}$ is assigned to either $\tau(1)$ or $\tau(2)$. Observe also that agent 3 is not getting her best project since $\tau(3) \in u^\sigma$. Now $(i, 3, \tau(3))$ is a blocking coalition. This is true because single-peakedness implies agent i strictly prefers $\tau(3)$ to $\tau(1)$ or $\tau(2)$.

By a symmetric argument, it cannot be the case that the assigned projects are $\tau(5)$, $\tau(6)$ and either $\tau(1)$ or $\tau(2)$.

The only remaining possibility is that assigned projects are one of $\{\tau(1), \tau(2)\}$, one of $\{\tau(5), \tau(6)\}$ and one of $\{\tau(3), \tau(4)\}$. Assume w.l.o.g $\tau(4)$ is the assigned project.

Observe that either agent 1 or 2 is not getting her best project. Suppose 1 is the agent who is not getting her best project. There are four possibilities.

(i) Agent 1 is getting $\tau(2)$ and agent 3 is not i.e. agent 3 is getting a project in $\{\tau(4), \tau(5), \tau(6)\}$.

$\succ_1^{part}$	$\succ_2^{part}$	$\succ_3^{part}$	$\succ_4^{part}$
2	3	4	1
3	1	2	2
4	4	1	3

Table 13: $\succ_i^{part}$

Since 3 is left-oriented, 1 and 3 block via the unassigned project $\tau(1)$.

(ii) Agent 3 is getting $\tau(2)$ but agent 1 is not i.e. agent 1 is receiving a project from the set $\{\tau(4), \tau(5), \tau(6)\}$. Then agents 1 and 3 block via the unassigned project $\tau(3)$.

(iii) Neither agent 1 nor agent 3 is getting $\tau(2)$. Here agents 1 and 3 block via one of the unassigned projects $\tau(1)$ or $\tau(3)$.

(iv) Agents 1 and 3 are assigned $\tau(2)$. Hence agent 2 is assigned a project from the set $\{\tau(3), \tau(4), \tau(5), \tau(6)\}$. Since agent 2 is left-oriented, agents 1 and 2 block the assignment via the unassigned project $\tau(1)$.

The final case is when $\tau(1)$ is being assigned. Agents 2 and 3 are assigned a project in $\{\tau(1), \tau(4), \tau(5)\}$. Then agents 2 and 3 block via the project $\tau(2)$.

The case where $\tau(3)$ is the third assigned project can be dealt by a symmetric argument. This completes the proof of the claim.

The next example demonstrates non-existence of stable assignments when the preferences for all agents belong to $D^2(N, A)$.[20] Let $N = \{1, 2, 3, 4\}$ and $A = \{a, b, c\}$. Let $1 >_p 2 >_p 3 >_p 4$ and $a >_o b >_o c$. Table 13 specifies $\succ_i^{part}$ for all agents i. The preferences on projects are not important and thus not specified. Also $\succ_i$ is partner dominant for all ii.e. they belong to $D^2(N, A)$. We show below that a stable assignment does not exist at profile $\succ$.

Consider the assignment where agents $(1, 2)$ and $(3, 4)$ are matched with each other. Let $(1, 2, x), (3, 4, y) \in \sigma$ where $x, y \in A$. Agent 2 strictly prefers $(3, y)$ over $(1, x)$ for all $x, y \in A$. Similarly agent 4 strictly prefers $(1, x)$ over $(3, y)$ for all $x, y \in A$. Thus agents 2 and 4 will block σ via a position switch.

Next consider the assignment where agents $(1, 4)$ and $(2, 3)$ are matched with each other. Agents 1 and 3 will block the assignment via a position switch. This is because agent 1 strictly prefers any assignment where 2 is her partner over any assignment where 4 is her partner. Similarly agent 3 strictly prefers any assignment where 4 is her partner over any assignment where 2 is her partner.

[20]Non-existence in this case is closely related to the non-existence of stable assignments in the roommate problem.

Finally consider the assignment where agents $(1, 3)$ and $(2, 4)$ are matched with each other. Agents 3 and 4 block the assignment via a position switch. Agent 3 strictly improves as she prefers any assignment where 2 is her partner over any assignment where 1 is her partner. Also agent 4 prefers any assignment where 1 is her partner over any assignment where 2 is her partner. Therefore there does not exist a stable assignment.

References

Alkan, A. (1988): "Nonexistence of stable threesome matchings," *Mathematical social sciences*, 16, 207–209.

Barberà, S., F. Gul, and E. Stacchetti (1993): "Generalized median voter schemes and committees," *Journal of Economic Theory*, 61, 262–289.

Bauman, K. E. and S. T. Ennett (1996): "On the importance of peer influence for adolescent drug use: Commonly neglected considerations," *Addiction*, 91, 185–198.

Biró, P. and E. McDermid (2010): "Three-sided stable matchings with cyclic preferences," *Algorithmica*, 58, 5–18.

Bogomolnaia, A. and H. Moulin (2004): "Random matching under dichotomous preferences," *Econometrica*, 72, 257–279.

Burkett, J., F. X. Flanagan, and A. L. Griffith (2016): "Allocating Group Housing," .

Cechlárová, K. and T. Fleiner (2005): "On a generalization of the stable roommates problem," *ACM Transactions on Algorithms (TALG)*, 1, 143–156.

Cohen, J. M. (1977): "Sources of peer group homogeneity," *Sociology of Education*, 227–241.

Combe, J. (2017): "Matching with Ownership," *Available at SSRN: https://ssrn.com/abstract=3071879.*

Golub, B. and M. O. Jackson (2012): "How homophily affects the speed of learning and best-response dynamics," *The Quarterly Journal of Economics*, 127, 1287–1338.

Kandel, D. B. (1978): "Homophily, selection, and socialization in adolescent friendships," *American journal of Sociology*, 84, 427–436.

Khare, S. and S. Roy (2016): "Stability in Matching with Groups having Non-Responsive Preferences," .

Klaus, B. and F. Klijn (2005): “Stable matchings and preferences of couples,” *Journal of Economic Theory*, 121, 75–106.

——— (2007): “Paths to stability for matching markets with couples,” *Games and Economic Behavior*, 58, 154–171.

Klaus, B., F. Klijn, and J. Massó (2007): “Some things couples always wanted to know about stable matchings (but were afraid to ask),” *Review of Economic Design*, 11, 175–184.

Lazarsfeld, P. F., R. K. Merton, et al. (1954): “Friendship as a social process: A substantive and methodological analysis,” *Freedom and control in modern society*, 18, 18–66.

McPherson, M., L. Smith-Lovin, and J. M. Cook (2001): “Birds of a feather: Homophily in social networks,” *Annual review of sociology*, 27, 415–444.

Morrill, T. (2010): “The roommates problem revisited,” *Journal of Economic Theory*, 145, 1739–1756.

Nicolò, A., A. Sen, and S. Yadav (2017): “Matching with Partners and Projects,” *Available at https://sites.google.com/site/anicolo68/.*

Pycia, M. (2012): “Stability and preference alignment in matching and coalition formation,” *Econometrica*, 80, 323–362.

Raghavan (2014): “Efficient Pairwise Allocation via Priority Trading,” *mimeo, Indian Statistical Institute.*

Roth, A. E., T. Sönmez, and M. U. Ünver (2005): “Pairwise kidney exchange,” *Journal of Economic theory*, 125, 151–188.

Roth, A. E. and M. A. O. Sotomayor (1992): *Two-Sided Matching: A Study in Game-Theoretic Modeling and Analysis*, 18, Cambridge University Press.

Ruef, M., H. E. Aldrich, and N. M. Carter (2003): “The structure of founding teams: Homophily, strong ties, and isolation among US entrepreneurs,” *American sociological review*, 195–222.

Selfhout, M., S. Branje, T. ter Bogt, and W. Meeus (2009): “The role of music preferences in early adolescents friendship formation and stability.” *Journal of adolescence*, 32, 95–107.

Sethuraman, J. and A. Smilgins (2016): "Two-sided matching with objects," *Mimeo, University of Copenhagen.*

Verbrugge, L. M. (1983): "A research note on adult friendship contact: a dyadic perspective," *Social Forces*, 78–83.