Efficient Pairwise Allocation via Priority Trading

T.C.A. Madhav Raghavan*

31 July, 2014

Abstract

We consider situations where heterogenous indivisible objects are to be distributed among a set of claimants based on preferences. We impose the additional restriction that each object must be assigned to exactly two agents, or it is not assigned at all. We show how standard rules in the literature fail to be efficient in this framework. We propose a rule which we call the Pairwise Priority Trading (PPT) Rule, and show that it characterises the set of all rules in this model that satisfy strategy-proofness, limited influence, unanimity and neutrality. It is also group-strategy-proof and Pareto efficient. The PPT rule can be thought of as a generalisation of the famous top trading cycles procedure to this particular environment.

1 Introduction

In this chapter we restrict our attention to cases in which each heterogenous object must be assigned to exactly two agents if it is to be assigned at all. To differentiate this from the classical single-unit object allocation problem, we will refer to these objects as 'projects', and the model as one of pairwise project allocation.

Such partnerships problems are common occurrences. Police precincts usually send officers out in pairs. Airlines allocate flight routes to pilots and co-pilots. Teachers often assign projects to pairs of students. Many hostels and dormitories allocate rooms to pairs of roommates. Managers may have tasks that require exactly two workers to perform them.

Individuals are concerned only about the assignment they receive. So the police officer will only care about his or her beat, the student will only care about which project she is assigned, and the hosteller is only interested in the room he is assigned. In particular, individuals will not care about the identity of the partner who is also assigned that project, even though there must be one. We will also assume that preferences are strict, in that there are no two projects to which any individual is indifferent.

An allocation in this context will be the assignment of projects to individuals, such that each project is assigned either to nobody or to exactly two people. Conceptually, it is as if each project will have two copies, such that if one copy is assigned to some agent, then the other copy *must* be assigned to some other agent as well.

What we seek in this context is a rule or a class of rules. A rule is a prescription of an allocation (or a procedure to determine an allocation) for any configuration of individual preferences. We will formulate such a rule (we call it the pairwise priority trading (PPT)[1] rule). We will impose some desirable properties on the rule. Our main result will be a characterisation, in that we will show that the class of PPT rules we identify are exactly those rules that satisfy the combination of properties[2] we consider. The properties as

*Senior Research Fellow, Indian Statistical Institute, New Delhi. Thesis advisor: Arunava Sen.

[1]The PPT rule makes assignments via an iterative process. In each stage, there are two ways in which an agent may receive his or her assignment. As the name suggests, assignments are based either on priorities or on trading. As we shall see, some of these ideas are familiar in the literature and some are novel to this rule.

[2]More precisely, the properties are strategy-proofness, limited influence, unanimity and neutrality.

well as the rule will be discussed at length in subsequent sections. The PPT rule can be considered to be an adaptation of the famous top trading cycles (TTC) procedure to our environment.

The paper is organised as follows. Section 2 presents a review of the literature on object allocation and recent extensions to project allocation. Section 3 is an informal discussion of the model. We will show how some of the common approaches used to assign objects efficiently fail in our context of pairwise project allocation. We will also discuss the workings of the PPT rule.

Next we get to the formal part of the paper. Section 4 presents the notation and main definitions that we use throughout the paper. Section 5 contains the formal specification of the PPT rule. Section 6 presents a detailed example of its working. Section 7 discusses the features of the rule and presents some special cases of PPT rules. Section 8 discusses the properties ('axioms') that we wish for our rule to satisfy. Section 9 presents our main result. Section 10 concludes and discusses possible extensions of this model. The proofs are relegated to the appendix.

2 Literature Review

The typical allocation problem deals with assigning indivisible objects. It is generally assumed that objects are such that each may be allocated to only one agent. The classical studies in the literature designate these objects as houses (Shapley and Scarf (1974)), and the general framework is called house allocation. Shapley and Scarf (1974) also introduce Gale's top trading cycles (TTC) procedure, which is an iterated procedure in which agents initially own objects and trade with each other. Trading is done via a 'pointing' mechanism that represents favourable trades via cycles. The TTC procedure will be discussed at length later, as it lies at the centre of a number of papers on efficient object allocation. The TTC procedure is also a feature of our rule.

The TTC is a robust rule. Roth and Postlewaite (1977) show how the TTC allocation coincides with the unique core allocation when preferences are strict. The TTC is also strategy-proof (Roth (1982)). Moreover, the TTC solution is the only Pareto-efficient, individually rational and strategy-proof rule (Ma (1994), Svensson (1999)). It is also group-strategy-proof (Bird (1984)).

Allocation mechanisms that are strategy-proof and Pareto efficient have been well-covered in the literature. Pápai (2000) characterises the set of Pareto efficient, group-strategy-proof and reallocation-proof mechanisms, and derives a wide class of functions called hierarchical exchange rules. Hierarchical exchange rules are an extension of the TTC procedure, via a generalisation of the initial endowment structure and by defining inheritance rules for unassigned objects. Also, Pycia and Ünver (2013) independently characterise the class of group-strategy-proof and Pareto efficient rules in this context.

Another generalisation of the TTC procedure can be found in Abdulkadiroğlu and Sönmez (1999) who consider a mixed model of house allocation and a housing market. They provide a strategy-proof, Pareto efficient and individually rational mechanism. The key feature of their model is the presence of exogenous property rights.

Under the assumption of neutrality, Svensson (1999) shows that the only strategy-proof and non-bossy mechanism in this model is the serial dictatorship. The serial dictatorship is one in which there is an exogenous fixed order of agents who get to successively pick their best options from the set available to them after previous choices have been made. Rhee (2011) extends this result to the case where each object must be assigned to a couple. Couples are ranked according to a hierarchy, and one agent in each couple serially selects an object. The other agent in the couple shares this object. The serial dictatorship is Pareto efficient in both cases. Pápai (2000) shows how the serial dictatorship can be embedded in a hierarchical exchange rule based on the TTC.

When we consider the case of multiple agents sharing an object, the extreme case is when one object must be assigned to all agents. Here, the seminal result in the literature is of course the Gibbard-Satterthwaite Theorem. Independently proposed by Gibbard (1973) and Satterthwaite (1975), it was shown that the only strategy-proof and onto rule is the dictatorial one. This result has been replicated in numerous instances (see, for example, Sen (2001), Barberà (1983)), and is also derived as a consequence of the serial dictatorship

result in Svensson (1999). As in that paper, this applies to the public goods case as well. It also applies to voting rules.

In some models, agents may receive more than one object. Pápai (2001) characterises the sequential dictatorship as the only rule that is strategy-proof, non-bossy and satisfies citizen sovereignty. Strengthening non-bossiness to total non-bossiness yields the serial dictatorship in this model. Hatfield (2009) considers a model where each agent has a capacity that must be filled exactly. He shows under a certain restriction on preferences that the only strategy-proof, Pareto optimal and nonbossy rule is a sequential dictatorship. Furthermore, he shows that the only strategy-proof, Pareto optimal, nonbossy, and neutral mechanisms are serial dictatorships.

The model in Hatfield (2009) is essentially a model of exact capacity constraints on the part of agents. In the context of capacity constraints for objects, Hylland and Zeckhauser (1979) present a model where objects have some maximum capacity constraint, and propose a strategy-proof and efficient rule. Fragiadakis et al. (2012) and Ehlers et al. (2011) also deal with capacity constraints. In particular, the objects in their models have minimum as well as maximum constraints. The former paper is relevant to our model, in that the TTC extensions they produce are related to our rule. However, they do not provide a characterisation of efficient and group-strategy-proof rules in the context of minimum and maximum capacities. The latter paper is concerned with fairness as opposed to efficiency.

3 An Informal Discussion

In this section we discuss some alternative approaches to pairwise project allocation. In particular, we show how the rules commonly used in object allocation need to be modified or extended to fit this context. After that we discuss the key features of our rule.

3.1 Other Approaches

Why is pairwise project allocation different from object allocation? Why can we not use the rules that have been shown to work in the single-unit case here as well? In what follows we attempt to demonstrate the reasons why.

Consider, as a natural starting point, the serial dictatorship. This is also known as the serial priority rule. The rule works as follows: there is an exogenous and fixed ordering of agents (agent I comes before agent J who comes before agent K, and so on). Each agent gets to pick his or her top-ranked object in turn. So the first agent is guaranteed to receive his or her top-ranked object in all cases. The second agent is faced with the objects that are left over after the first agent has made his or her selection. The second agent is then guaranteed to receive her top-ranked object from all the ones that remain. The only object she cannot receive is the one that has been selected by the first agent. Every subsequent agent in the ordering picks from the set of objects left behind after earlier agents have made their choice.

The serial priority rule has been characterised by Svensson (1999) as the only rule that is strategy-proof, non-bossy and neutral. Rhee (2011) extends this result to the case (similar to ours) where agents are organised in pairs. The equivalent rule to serial priority in this context works as follows: there is an ex ante separation of agents into pairs, an exogenous ranking of pairs, and a fixed selection of one agent from each pair. These selected agents are ordered according to how the pairs are ordered. The rule applies the serial priority method to these agents, as in the single object case. The other agent in the pair is automatically assigned the object his or her partner has selected. Rhee shows that that this extended rule is also characterised by the axioms of strategy-proofness, non-bossiness and neutrality.

However, while the original rule is also Pareto efficient, the extended rule is no longer so. To see this, note that the non-selecting partners in each pair in the extended rule have no say in their assignments. In particular, it may be that two non-selecting partners would actually prefer to be assigned each other's object, i.e., belong to a different pair. The rule does not allow any such profitable swaps, and is thus inefficient. Since we look for Pareto efficient rules in the pairwise model, this rule will not serve our purpose.

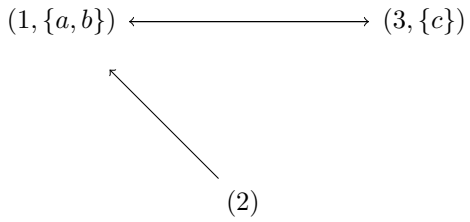
So consider the following thought experiment. Each project in our model can be considered to comprise two 'copies'. If one copy is assigned to some agent, the other copy must be assigned to some other agent as well. We could then label each copy as a distinct object in itself. For example, project a could be separated into a_1 and a_2. As long as we ensure that if one of them is assigned, that the other must be as well, there is no threat to feasibility from this approach.

So we could apply the original serial priority rule to this case with $2x$ 'objects', where x is the number of original projects. Agents are ordered, and would select according to their order. We would suitably restrict the set of objects available to later agents to ensure feasibility of the overall assignment in the original context of projects. This rule, as the original one was, will also be strategy-proof, non-bossy and neutral.[3] However, it is no longer characterised by those properties. That is, when we translate this model back to the original pairwise project allocation setting, we can find other rules as well that satisfy those properties, that are not serial or even sequential priority rules. So if we seek a full characterisation of this class of rules, we have to look beyond purely serial or sequential priority.

So let us consider the famous top trading cycles (TTC) rule attributed to David Gale (see Shapley and Scarf (1974)). In brief, the TTC works as follows:

Each object is initially owned by one agent, who brings it to the market for trade[4]. Some agents may initially own more than one object, while others may own none at all. The procedure works in stages. In any stage, each agent who is yet to receive an assignment points to the owner of the object she most prefers from the ones that are available. A top trading cycle is made up of agents who successively point to the next agent, with the last agent pointing to the first. A cycle can be a singleton, such that an agent points to herself (she owns the object she most prefers.) Since there is a finite number of agents, at every stage there must always be a cycle. Agents in a cycle trade their objects along the cycle until they receive the object they desire. This becomes their assignment and such agents leave the market along with those objects. If there are still agents and objects left unassigned, the procedure repeats in the reduced market. If preferences are strict, then given an initial ownership, the resulting allocation is unique.

The TTC rule is illustrated by an example. Suppose there are three agents $(1, 2, 3)$ and three objects (a, b, c). Suppose agent 1 initially owns a, b and agent 3 initially owns c. Agent 1 desires c, while agents 2 and 3 desire b. The TTC procedure would look as follows:



Agent 1 'points to' agent 3 who owns c, and agents 2 and 3 in turn point to agent 1 who owns b. The cycle in this stage is between agents 1 and 3, who consequently trade those objects. The TTC would assign b to agent 3 and c to agent 1.

As discussed earlier, TTC rules and their generalisations to inheritance rules (Pápai (2000)) are indeed group-strategy-proof and Pareto efficient. An inheritance rule in the above example would also specify how agent 2 'inherits' the remaining object a. The TTC procedure in the second stage would just be agent 2 pointing to herself, and a would become her assignment.

Sequential and serial priority rules form a sub-class of inheritance rules. To see this, note that a serial priority effectively grants initial ownership of all objects to some agent. This agent can always pick his or her top-ranked object from among them. Additionally, the rule specifies that all objects left over after his

[3] As an added bonus, it will also be Pareto efficient.

[4] Objects that an agent initially owns form a part of his or her 'endowment'

or her selection are inherited by some other fixed agent. This agent can always pick her top-ranked object from among them. The remaining objects are inherited by some fixed third agent, and so on.

However, when faced with projects, the idea of initial ownership needs to be generalised as well. Since there are two copies, each project could be initially collectively owned by a pair of agents. Different projects may be owned by different pairs of agents. But note that pairwise project allocation implicitly involves not just the assignment of projects, but also the question of how to select the assigned projects in the first place. How do we select the projects? What makes one pair's endowment superior to another's? When do we decide that this project can be traded while another one cannot? Unless the number of projects is exactly equal to capacity, there will always be this selection problem. Thus a specification of a TTC rule will have to account for these possibilities as well.

There is another conceptual problem with naively using the TTC rule in this setting. Note that the success of the TTC procedure rests on the existence of a distinct cycle in every stage of pointing. We can then unambiguously trade objects along the cycle until the agents are satisfied. The cycle itself arises because each agent desiring an object can unambiguously point to a single owner of that object. But how would that work here, where objects have two copies? If an agent desires an object owned by two other agents, who are both available, then to whom should she point? There exists the possibility now of multiple overlapping cycles. Addressing this problem by introducing a tie-breaker may create more problems than it solves. Thus we must be careful in designing a TTC-based solution.

We do indeed base our rule around the TTC, but we also account for the problems above. As a result, we see that the TTC rule cannot be directly used in this framework. Our rule is therefore more complicated than the simple TTC rule itself.

3.2 The Pairwise Priority Trading Rule

The pairwise priority trading (PPT) rule proceeds in stages. The information at each stage is captured by a 'state' vector, which essentially is a partial allocation that keeps track of agents' assignments up to that stage. This state information serves as input for functions that specify the relative order of agents who get to choose subsequently. These functions are called a 'pairing', a 'proposal vector' and an 'endowment', respectively. Collectively we call them the 'entitlement' for that state.

Agents may receive their assignments in one of two ways - via a proposal or by trading.

Note that for feasibility the total number of different projects that can be assigned in any feasible allocation is some integer m. At each stage in which the number of different projects assigned is less than m, the pairing component of the entitlement for that state will identify a pair of agents from those that still await their assignments. This pair has the following property: whenever both agents in the pair desire the same project from among those that are fully available (have not been assigned to anyone), they are guaranteed this project as their assignment.

We will develop the notion of a proposal, which is a sequence of agents. The first agent in the sequence is called the 'proposer'. A proposal will capture the sequence in which agents can 'join' the proposer in a project that she proposes. The second agent in the proposal will have the property that she can always join the proposer in her top-ranked project if she wishes, guaranteeing both of them that assignment. However, if she declares some other top-ranked project, the third agent may now join the proposer. If this happens, then all three agents are guaranteed their top-ranked projects (the first and third agents together, and the second agent her distinct project.) In fact, a proposal could be even longer. It may be that the fifth agent in the proposal agrees with the proposer and thus can join her in that project, provided each intermediate agent picks a distinct top-ranked project. Each intermediate agent is also guaranteed her assignment in this case. Note that the identities of the projects are immaterial to the exercise; only what matters is whether they are all distinct, and if a subsequent agent in the proposal agrees with the first agent or not. If so, they can always guarantee the projects they are associated with in the proposal. We call such a proposal 'acceptable'.

The collection of proposals for the state depend on the pairing. The proposer will be one of the two agents determined by the pairing. We will call the collection of proposals for the state the proposal vector.

In the second step of the process therefore, as long as the number of different projects assigned is less than m, we will evaluate the various proposals that may exist at this stage. If we find an acceptable proposal, we make the assignments. If we cannot, then we give the first proposer her top-ranked project. In this way, at least one agent will be guaranteed to receive an assignment at any stage in which we use this procedure.

For partially assigned projects, i.e., those that have been assigned to exactly one agent thus far, we use the TTC procedure. For each project that has already been assigned, but has only been assigned to one agent thus far, there is a designated 'partner' who can be thought of as owning the remaining copy of that project. This agent is yet to receive her assignment. We can consider this project to be part of her endowment, in that she could always guarantee its assignment for herself. But she could also put it up for trade. If there is another similarly placed agent who owns a copy of some other project, and they each desire each other's projects, they can swap. This also applies to more than two agents. However, as we shall discuss in more detail later, there is a key difference between the classical TTC procedure and the way it applies to our model. In particular, there is not necessarily a cycle at every stage, even in stages where there are agents with non-empty endowments. Note also that the TTC procedure only applies in situations where one copy of the project has already been assigned. Agents that can participate in a TTC round are those who have the remaining copy of that project in their endowment. Thus an agent who desires that project can unambiguously point to an 'available owner' of that project. In this way we do away with the possibility of multiple cycles.

The PPT rule is then a collection of the above sub-procedures. For any combination of agent preferences, the rule proceeds by evaluating each of the above three situations in turn, and making some assignments. There will always be at least one assignment made in every stage. In subsequent stages, the assignments made thus far will become inputs for the selection of agents for the sub-procedures above. The entire procedure terminates when all agents have received their assignment.

3.3 Criteria for Allocation Rules

In this paper we are concerned with the following four criteria for rules. The first is strategy-proofness, which ensures that it is always a dominant strategy for every agent to truthfully report his or her preferences. The second is a composite criterion which we call 'limited influence'. The first part of this criterion is familiar in the literature as the non-bossiness condition. Non-bossiness stipulates that an agent may not affect another agent's assignment via a change in reported preferences, if she does not change her assignment as well. The second part is new to this paper but similar versions have appeared in other papers as well. It seeks to limit the influence that an agent has on the assignment of certain projects via a change in reported preferences, even if her own assignment changes. We shall discuss this axiom in more detail in a later section.

The third criterion for a rule is unanimity, which states that a rule should respect the self-selection of agents into feasible pairs. If agents report preferences such that it is feasible to give every agent her top-ranked project, than a unanimous rule must do so. The final condition is neutrality, which requires the rule to treat all projects symmetrically. The main result in this paper is that a PPT rule is characterised by the above four axioms.

Next, we go to the formal part of the paper, where we define all the above notions rigorously.

4 Notation and Definitions

The model is described below.

- There is a finite set of *agents* $\mathcal{N} = \{1, ..., i, j, , k, ..., N\}$ and a finite set of *projects* $\mathcal{Z} = \{a, b, c, d, ...\}$. We assume that $|\mathcal{N}| = 2m$ for some integer $m \geq 2$ and that $|\mathcal{Z}| \geq m + 1$.[5]

[5]This assumption of at least two 'pairs' and three projects is equivalent to the assumption in object allocation models of at least two 'agents' and three objects.

- An *allocation* $x \in \mathcal{Z}^N$ with $x = (x_1, ..., x_N)$ is a vector that associates a project with each agent. For any agent $i \in \mathcal{N}$, $x_i \in \mathcal{Z}$ is the *assignment* of agent i in x. An allocation x is *feasible* if, for all $a \in \mathcal{Z}$, $|\{j \in \mathcal{N} : x_j = a\}| \in \{0, 2\}$. That is, an allocation is feasible if it assigns each project to exactly two agents, or to nobody. The set of all feasible allocations is given by $\mathcal{A}$.

- Preferences over assignments are strict. Formally, agent $i \in \mathcal{N}$ has *preferences*, denoted R_i, that are given by a binary relation over $\mathcal{Z}$. For any a, b, aR_ib is interpreted as 'project a is at least as good as project b for agent i under preferences R_i'. The binary relation is reflexive (for all a, aR_ia), complete (for all a, b, aR_ib or bR_ia), transitive (for all a, b, c, aR_ib and bR_ic imply aR_ic) and antisymmetric (for any a, b, aR_ib and bR_ia imply $a = b$). The associated strict relation is given by P_i, such that aP_ib if aR_ib and $a \neq b$. For any a, b, aP_ib means 'a is preferred by i to b under preferences R_i'.

- Agent preferences over allocations are selfish, in that they care only about the assignment they receive. Agents are indifferent between all allocations that give them the same assignment. An agent's preferences between two allocations that give her different assignments are governed by her preferences over the respective assignment she receives.

- A collection of preferences for all agents is called a *preference profile*, or simply a *profile*, and is denoted by $R = (R_1, ..., R_N)$. The set of all preference profiles is $\mathcal{R}$. In this model we shall usually suppress reference to $\mathcal{R}$, with the understanding that we operate on the full domain of preferences everywhere. As is the convention, we write R_{-i} for a sub-profile of preferences of all agents other than i. Similarly, for a subset of agents M, we write R_M and R_{-M} to denote the sub-profile of preferences of agents in subsets M and $\mathcal{N} \setminus M$, respectively.

- A *pairwise project allocation rule (P-PAR)* is a function $f : \mathcal{R} \to \mathcal{A}$ that maps every preference profile to a feasible allocation. For any agent i, $f_i(R)$ is the assignment she receives at preference profile R according to the rule f. Similarly, for any subset of agents M, $f_M(R)$ is the M-dimensional vector of assignments of M at R, according to f.

5 Pairwise Priority Trading Rules

5.1 States

We first present some useful terminology. The notion of a state will be useful to keep track of agents' assignments as the algorithmic procedure in our rule unfolds.

Formally, a *state* is a vector $s \in (\mathcal{Z} \cup \emptyset)^N$ with $s = (s_1, ..., s_N)$ such that:

1. $|\{j \in \mathcal{N} : s_j = a\}| \leq 2$ for any $a \in \mathcal{Z}$
2. $|\{a \in \mathcal{Z} : s_i = a \text{ for some } i \in \mathcal{N}\}| \leq m$.

A state is essentially a partial allocation. The two conditions are necessary for feasibility. The first condition ensures that no project is associated with more than two agents for any state, and the second condition ensures that not more than m projects are associated with any state. Let the set of all states be denoted $\mathcal{S}$. It follows from the definition above that the set of feasible allocations is a subset of the set of states, i.e., $\mathcal{A} \subset \mathcal{S}$.

For a state s, we define the set of *assigned agents in* s as $N(s) = \{i \in \mathcal{N} : s_i \neq \emptyset\}$, the set of *unassigned agents in* s as $\bar{N}(s) = \{i \in \mathcal{N} : s_i = \emptyset\}$, the set of *assigned projects in* s as $Z(s) = \{a \in \mathcal{Z} : s_i = a \text{ for some } i \in \mathcal{N}\}$, the set of *partially assigned projects in* s as $\hat{Z}(s) = \{a \in \mathcal{Z} : s_i = a \text{ for exactly one } i \in \mathcal{N}\}$, the set of *unassigned projects in* s as $\bar{Z}(s) = \{a \in \mathcal{Z} : s_i \neq a \text{ for all } i \in \mathcal{N}\}$, and the *number of remaining projects in* s as $m'(s) = m - |Z(s)|$. It is clear that $\hat{Z}(s) \subseteq Z(s)$, $\mathcal{N} = N(s) \cup \bar{N}(s)$, $\mathcal{Z} = Z(s) \cup \bar{Z}(s)$ and $0 \leq m'(s) \leq m$ for any s.

If $m'(s) < m$, we call s an *interim state*. Note that $\bar{N}(s) \geq 2$ for any interim state s. It shall be convenient to call the null vector $(\emptyset, ..., \emptyset)$ the *null state*. If $|\{j \in \mathcal{N} : s_j = a\}| \in \{0, 2\}$ for all $a \in \mathcal{Z}$, we call s a *terminal state*. For any preference profile, our rule will start with the null state and progressively assign projects to agents in stages until we reach a terminal state. It is easy to see that a terminal state is a feasible allocation.

For example, let $\mathcal{N} = \{1, ..., 6\}$ and let $\mathcal{Z} = \{a, b, c, d\}$. Then the null state is given by $(\emptyset, ..., \emptyset)$, an interim state could be the vector $(a, \emptyset, b, b, \emptyset, \emptyset)$, and a terminal state could be (a, c, b, b, c, a).

5.2 Entitlements

As mentioned, the PPT rule uses the state information as inputs to functions that determine the identities of agents who get to 'go next'. We will call the collection of these functions an entitlement. In this subsection we formally define an entitlement, which has three components.

5.2.1 Pairing

The first component of an entitlement is a function that, for any interim state, identifies a pair of agents that can always guarantee the mutual assignment of a project when they declare it as their top preference. This is true for any project among the ones unassigned in that state. Formally:

A *pairing* is a function $g : \mathcal{S} \to (\mathcal{N} \cup \emptyset)^2$ with $g(s) = (g_1(s), g_2(s))$ such that:

(G1). $g_i(s) \in \bar{N}(s)$ for any interim state s and any $i \in \{1, 2\}$

(G2). $g_1(s) \neq g_2(s)$ for any interim state s

(G3). $g(s) = \emptyset$ for any state s where $m'(s) = m$

Condition G1 requires that the pairing be from the set of unassigned agents at each interim state. Condition G2 ensures that each agent in a non-empty pairing is distinct. Condition G3 states that non-empty pairings exist only for interim states.

5.2.2 Proposals and Proposal Vectors

As mentioned above, a pairing $g(s)$ at an interim state s identifies a pair of agents who, whenever they agree on their top-ranked project from what's available, they can guarantee its assignment. But in many cases they may not agree. A proposal is designed to capture this situation. In what follows we elaborate on proposals and proposal vectors. Note that these objects are conditional on the pairing specified for the relevant state. In particular, the agents specified by $g(s)$ play a key role in the proposals in that state.

First we start with useful definitions. For an interim state $s \in \mathcal{S}$ and an agent i, a *proposal* $t^i(s)$ is a finite sequence of agents given by $(t^i_1(s), t^i_2(s), ...)$ such that:

(P1). $t^i_1(s) = i$

(P2). $t^i_j(s) \in \bar{N}(s)$ for all j

(P3). $t^i_j(s) \neq t^i_k(s)$ for all $j \neq k$

(P4). $2 \leq |t^i(s)| \leq (m'(s) + 1)$

Condition P1 ensures that for a proposal associated with an agent i, the first element in the proposal is the agent herself. We call this agent the *proposer*. Condition P2 ensures that all agents in the proposal are unassigned agents. Condition P3 ensures that all agents in a proposal are distinct. Condition P4 restricts the length of a proposal to lie between 2 and $m'(s) + 1$. Note that the proposal itself makes no mention of projects.

For example, let $\mathcal{N} = \{1, ..., 6\}$ and let s be the null state. Then $\bar{N}(s) = \mathcal{N}$ and $m' = 3$. Some possible proposals are the following:

$$t^1(s) = (1, 2, 5), t^2(s) = (2, 3, 5, 1), t^5(s) = (5, 6)$$

For a given state s, let the set of all proposals satisfying the above properties be given by $\mathcal{T}(s)$. For an interim state s, let $g(s)$ be a pairing as specified above. We define a *proposal vector* as a finite ordered collection of proposals $T(s) = (t^1(s), t^2(s), ..., t^K(s))$. We refer to the i^{th} entry of $T(s)$ as $T_i(s)$. The proposal vector $T(s)$ has the following properties:

(PV1). $t^i(s) \in \mathcal{T}(s)$ for all i.

(PV2). $t^1(s) = (g_1(s), g_2(s))$

(PV3). $t^i_1(s) \in \{g_1(s), g_2(s)\}$ for all i.

(PV4). For any i, j with $i \leq j$, we have that $|t^i(s)| \leq |t^j(s)|$.

(PV5). For any $2 \leq k \leq m'(s) + 1$, there are $t^i(s), t^j(s) \in T(s)$ such that $|t^i(s)| = |t^j(s)| = k$.

Condition PV1 requires that the proposal vector be composed of valid proposals for that state. Condition PV2 requires that the first proposal in the vector be simply the agents in the pairing specified for this state. Condition PV3 requires the proposer for each proposal to be one of the agents specified by g for that state. Condition PV4 orders the proposals in the collection by increasing length. Condition PV5 states that there are at least two proposals of any length between 2 and $m'(s) + 1$.

To continue the example above ($|\mathcal{N}| = 6, m' = 3$), if $g(s^0) = (1, 3)$, a proposal vector for the null state could be the following:

$$T(s) = ((1, 3), (3, 1), (3, 4, 5), (1, 3, 5), (1, 3, 2, 4), (3, 4, 2, 6))$$

It is easy to check that this proposal vector satisfies all three properties above. We will show in Section 5.3 how to use proposal vectors to make assignments.

5.2.3 Endowments

Given a state s, there may be some projects that have been assigned to only one agent so far. Feasibility requires that the copy of this project be assigned to another agent. The endowment is designed to capture the rights that unassigned agents have to the copies of such projects. Only these copies can belong to an agent's endowment. Projects that have not been assigned to any agent yet are not part of any agent's endowment.

For any two states s and s', we say that s is a *precursor state of* s' if $s_i \neq \emptyset \implies s'_i = s_i$ for all $i \in \mathcal{N}$. Correspondingly, s' is called a *successor state of* s if s is a precursor state of s'.

For a given state s, an *endowment* is a function $E : \mathcal{S} \to 2^{|\mathcal{Z}| \times N}$ with $E(s) = (E_1(s), ..., E_N(s))$ such that:

(E1). $E_i(s) \subset (\hat{Z}(s) \cup \emptyset)$ for all $i \in \mathcal{N}$

(E2). $E_i(s) \neq \emptyset \implies i \in \bar{N}(s)$

(E3). $E_i(s) \cap E_j(s) = \emptyset$ for all $i \neq j$

(E4). Let $a \in E_j(s)$. For any state s' that is a successor state of s, if $j \in \bar{N}(s')$ then $a \in E_j(s')$.

Condition E1 requires all endowments to be either empty or from the set of partially assigned projects at that state. Condition E2 allows non-empty endowments only to unassigned agents. Condition E3 requires all endowments to be distinct. Condition E4 is a consistency condition across states. Any project in an agent's endowment must remain in her endowment as long as she is unassigned, for any successor state.

If an agent with a non-empty endowment receives an assignment in a round, then the unassigned projects in her endowment are transferred to other agents for the next round. The nature of these transfers bears some resemblance to the inheritance rules in Pápai (2000), as we shall discuss subsequently.

For a state s and an endowment $E(s)$, it shall be convenient for us to refer to the set of agents with non-empty endowments, or *endowed agents in* s, as $\hat{E}(s) = \{i \in \mathcal{N} : E_i(s) \neq \emptyset\}$.

5.2.4 Entitlements

We are now ready to define an entitlement. For a state s, an *entitlement* $\Gamma(s) = (g(s), T(s), E(s))$ is a pairing, a proposal vector and an endowment. A collection of entitlements for every state is denoted Γ.

A pairwise priority trading rule f^{Γ} is a specification of Γ, i.e., an entitlement for every interim state $s \in \mathcal{S}$, along with an iterative procedure that prescribes an allocation based on these entitlements for every preference profile. In what follows, we describe this procedure.

5.3 Assignments

The rule proceeds in rounds. Let us assume that a preference profile R has been realised and that the rule has attained an interim state s. There will be a round corresponding to this state.

In any round, there are two ways in which agents may receive their assignments. Either they receive their assignments via evaluation of proposals, or they participate in the top trading cycles procedure. At least one agent receives her assignment in any round. Each of these is discussed below.

5.3.1 Proposals

First, we define the notion of an acceptable proposal, which we will use to determine assignments. Let s be an interim state, R a preference profile and let $T(s)$ be the proposal vector at this state. Note that proposals in the vector are ordered from first to last in increasing length, and there are only two distinct proposers in the vector. For every $i \in T(s)$, let the top-ranked project a_i from the set of available projects[6] in this state be called *the label of* i *in* $\bar{Z}(s) \cup \hat{Z}(s)$, i.e., $a_i = top(R_i, \bar{Z}(s) \cup \hat{Z}(s))$.

A proposal $t^i(s)$ is *acceptable* if:

AP1. $a_{i_j} \neq a_{i_k}$ for all $j, k \in \{1, ..., K-1\}$

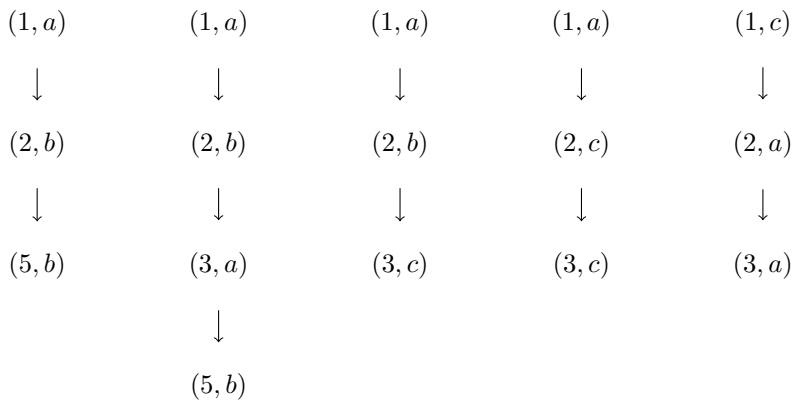
AP2. $a_{i_{K-1}} = a_{i_K}$

AP3. $a_{i_K} \in \bar{Z}(s)$

Conditions AP1 and AP2 require the label of every agent in the proposal to be distinct, except those of the last two agents, which must be the same. Condition AP3 ensures that the repeating project (the label of the last two agents) is an unassigned project, i.e., assigned to no agent. It follows that for a given state, proposal vector and preference profile, an acceptable proposal may not exist.

Let s be some interim state, let $\bar{Z}(s) = \{a, b\}$ and let $\hat{Z}(s) = \{c\}$. Consider the following proposals. The first is acceptable as the first agent's label is distinct, while the second and third agents have the same label. All labels are from the set of unassigned projects. The second proposal is not acceptable as not all intermediate labels are distinct. The third proposal is not acceptable as the last two agents do not have the same label. The fourth proposal is not acceptable as the repeating project is a partially assignment project. The fifth proposal is also acceptable. The proposer's label is from the partially assigned project set while the repeating project is from the set of unassigned projects.

[6] The set of available projects is the set of unassigned projects and the set of partially assigned projects.



We show how to use acceptable proposals to make assignments.

Proposal Vector Evaluation

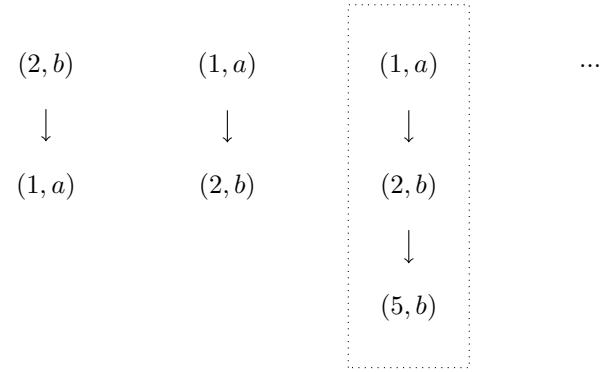
1. Start with $T_1(s)$, the first proposal in the vector. We check to see if it is an acceptable proposal.
2. If it is acceptable, assign each agent i_j her label. We will then make no more assignments at this stage.
3. If it is not acceptable, we move on to the second proposal. We check to see if this proposal is acceptable. If it is acceptable, we assign all agents in the proposal their corresponding labels. We will make no more assignments at this stage.
4. In general, if we have not encountered an acceptable proposal up to proposal $k-1$, we check to see if proposal k is acceptable. If it is, we assign all agents in the proposal their label and make no more assignments. If it is not acceptable, we check the next proposal.
5. We repeat until we either find an acceptable proposal or we exhaust all proposals. If we have exhausted all proposals without finding an acceptable proposal, we assign the proposer of the first proposal her label. We make no more assignments at this stage.

For example, let $\mathcal{N} = \{1, ..., 6\}, \mathcal{Z} = \{a, b, c, d, e\}$ and let s be the null state. For a given preference profile R, let the vector of labels (i.e. top-ranked projects) in $\mathcal{Z}$ be given by (a, b, c, d, a, a). Let a (truncated) proposal vector be given as follows:

$$T(s) = ((2, 1), (1, 2), (1, 2, 5))$$

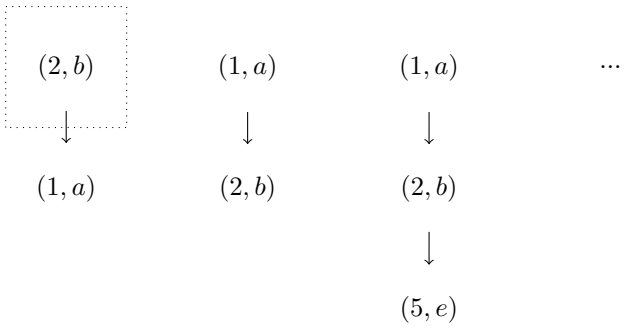
We draw each proposal vertically, labelling each agent i in the proposal by (i, a_i). Note that each agent has a label from the set of unassigned projects. We do not consider partially assigned projects at this point.

Drawing the proposals, we get:



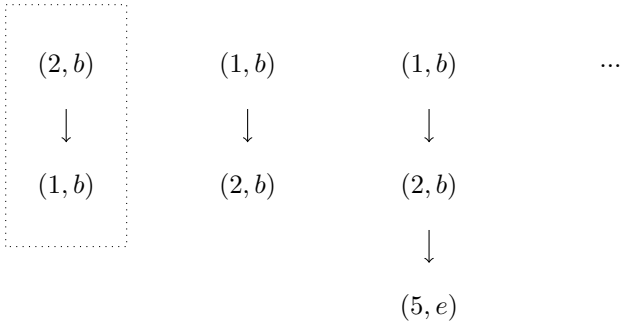
In the above example, the first acceptable proposal is $(1, 2, 5)$. So we would assign project a to agent 1 and assign project b to agent 2 and 5. We would make no further assignments.

Suppose instead that the vector of labels in $\mathcal{Z}$ is given by (a, b, c, d, e, a). Drawing the proposals, we get:

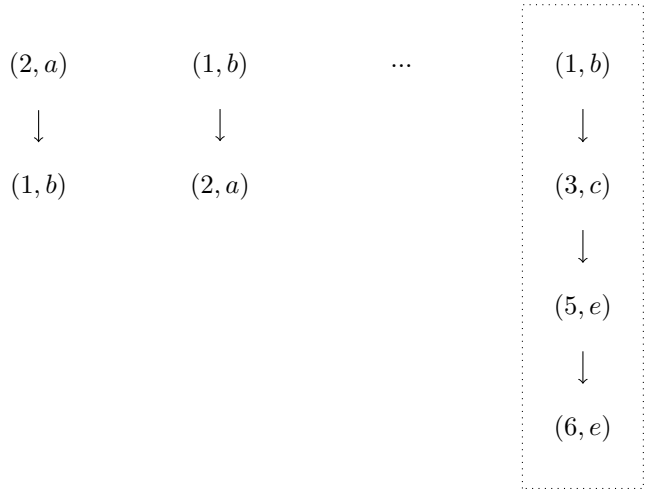


Suppose there are no acceptable proposals. Then only the proposer of the first proposal gets her assignment. No more assignments would be made here.

If the vector of labels was (b, b, c, d, e, a), then evaluating the same proposal vector for this state would yield us the assignment as in the picture below - agents 1 and 2 are assigned b.



If the vector of labels was (b, a, c, d, e, e) and the proposal vector was given as in the picture below, then the assignment would be as marked - agent 1 gets b, agent 3 gets c, while agents 5 and 6 are assigned e.



5.3.2 Top Trading Cycles

The top trading cycles method of assigning projects applies only to partially assigned projects, i.e., those that have been assigned to exactly one agent thus far. No 'new' project will be assigned this way.

In any TTC round, all agents with non-empty endowments (which are basically copies of partially assigned projects) 'point to' the agent holding her top-ranked project among the set of 'available' projects[7]. If her top-ranked project is unassigned, she points to nobody. A cycle is a distinct sequence of agents with each agent in the sequence pointing to the next agent, and the last agent pointing to the first. A cycle can be a singleton, with an agent pointing to herself. For any cycle that we might encounter, we trade projects along the cycle until each agent is holding her most preferred project. This becomes their assignments. We do this for all cycles that we find.

There is, however, one key difference between top trading cycles in our model versus the classical case. In particular, there may be stages in our rule in which a top trading cycle may not exist, even though there are agents owning copies of projects and agents are finite. To see this, note first that in the classical model, all objects are owned by some agent. So if an agent desires an object, there will always be an agent to whom she must point, even if it is herself. In our model, even though copies of assigned projects may be owned by agents, there is no ownership of unassigned projects as yet. In any interim state, these are still be available. In that case, an agent who desires that project cannot point to any agent in particular, and thus cannot be part of a cycle. While other agents may still form a cycle for copies of projects they own, it is nevertheless possible that no cycle exists. In this case, the top trading cycles part of our procedure does not produce an assignment at this stage, and we move on to the next stage.

In what follows, we set up the top trading cycles part of our procedure, keeping the above in mind.

TTC Evaluation

Let s be a state and let $E(s)$ be an endowment. Let R be a preference profile. Let $\hat{E}(s)$ be the set of endowed agents in s.

1. If s is an interim state ($m'(s) < m$), each agent $i \in \hat{E}(s)$ 'points to' the agent in $\hat{E}(s)$ who holds the project $top(R_i, \bar{Z}(s) \cup \hat{Z}(s))$. If this project is not owned by any agent, she points to nobody. Instead if $m'(s) = m$, each agent $i \in \hat{E}(s)$ points to the agent in $\hat{E}(s)$ who holds the project $top(R_i, \hat{Z}(s))$.

2. A cycle is a set of agents $(i_1, i_2, ..., i_n = i_1)$ such that each agent i_j points to i_{j+1}.

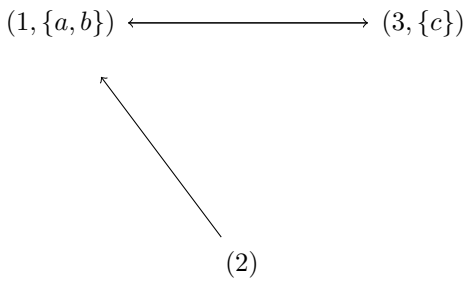
3. No cycle may exist.

4. If a cycle exists, agents in the cycle trade projects along the cycle until they hold their most preferred project. This becomes their assignment. This is done for each cycle that may exist.

5. There is only one round of pointing for a state. We stop when either we do not find a cycle or when we have traded all cycles that we find.

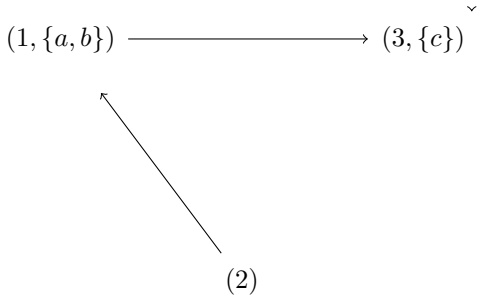
For example, suppose there are three agents $(1, 2, 3)$ with non-empty endowments, and three partially assigned projects (a, b, c). Suppose a, b are part of agent 1's endowment and c is part of agent 3's endowment. Agent 1 desires c, while agent 2 desires a and 3 desire b. The TTC procedure would look as follows:

[7] The set of available projects is the combination of partially assigned projects and unassigned projects.



Agent 1 'points' to agent 3 who owns c, and agents 2 and 3 in turn point at agent 1 who owns a and b respectively. The cycle in this stage is between agents 1 and 3, who consequently trade those objects. The TTC procedure at this stage would assign b to agent 3 and c to agent 1.

Now suppose instead that $m'(s) < m$ for this stage, and so there is at least one other project that can be assigned. Suppose further that agent 3 desires some as-yet-unassigned project e above b. Then the TTC procedure at this stage would yield no cycle, as in the following picture:



Agent 3 desires project e, which is unassigned and is thus owned by nobody. Thus she points to nobody. Thus no assignment would be made by TTC in this stage.

5.4 The Iterative Procedure

Fix a preference profile R. Then a PPT rule $f^{\Gamma}(R)$ is determined by an iterative procedure with a finite number of stages. For any preference profile, in the first stage we start with the null state $s^0(R) = (\emptyset, ..., \emptyset)$. Each subsequent stage k of the procedure begins with a state $s^{k-1}(R)$ that captures the assignments of agents made up to stage $k-1$.

At any stage k, we perform two steps.

In the first step, among agents with non-empty endowments, we run the top trading cycles procedure as described earlier. It is possible that no assignments are made in this step. However, if s is not an interim state, then at least one agent will receive an assignment in every TTC step. In the very first stage of the procedure, there are no agents with non-empty endowments so no assignments will be made. In general, after running the trading step, we go on the other step at this stage, evaluating proposals.

In this step, we use the proposal vector for this state to make assignments as described in the previous sub-section. Note that at least one assignment will be made in this step. Note also that proposal vectors are also used only for interim states. The proposal step will make no assignments for any other state.

Once the steps are complete and assignments for this stage are made, we update the state $s^{k-1}(R)$ to reflect all assignments made up to and including stage k, giving us a new state $s^k(R)$. We then proceed to stage $k+1$ of the procedure, which repeats this two-step process. The procedure continues until a *terminal*

stage K which is the first stage in which all agents have received their assignments[8]. The corresponding terminal state s^K will give us the final allocation.

Formally:

Let R be a profile. Let Γ be given.

Stage 1

The state is $s^0(R)$ (the null state). Let $\Gamma(s^0) = (g(s^0), T(s^0), E(s^0))$ be the entitlement for this state.

Trading Step: There are no endowments in this stage. So we make no assignments and go to the Proposal Step.

Proposal Step: Evaluate $T(s^0)$ using the Proposal Vector Evaluation. For any agent k receiving an assignment at this step, update $s_k^1(R)$ as this project. For all other agents j, $s_j^1(R) = s_j^0(R)$ Go to the Verification Step.

Verification Step: If $s^1(R)$ is a terminal state, we stop with the resulting allocation. If not, we proceed to Stage 2.

Stage k+1, $k \geq 1$

The state is $s^k(R)$. Let $\Gamma(s^k) = (g(s^k), T(s^k), E(s^k))$ be the entitlement for this state.

Trading Step: Run the TTC Evaluation for all agents in $\hat{E}(s^k)$. For any agent i receiving an assignment at this step, update $s_i^{k+1}(R)$ as this project. For all other agents j, $s_j^{k+1}(R) = s_j^k(R)$. Go to the Proposal Step.

Proposal Step: Evaluate $T(s^k)$ using the Proposal Vector Evaluation. For any agent j receiving an assignment at this step, update $s_j^{k+1}(R)$ as this project. For all other agents j, $s_j^{k+1}(R) = s_j^k(R)$. Go to the Verification Step.

Verification Step: If $s^{k+1}(R)$ is a terminal state, we stop with the resulting allocation. If not, we proceed to Stage $k+2$.

Note that $f^\Gamma(R)$ is unambiguously defined, as for every R and every i there is at most one stage in which i receives her assignment. The procedure is also finite as at least one agent receives her assignment at every stage.

6 A Detailed Example

We now present a detailed example that highlights the key features of the PPT rule f^Γ.

Let $\mathcal{N} = \{1, ..., 8\}, \mathcal{Z} = \{a, b, c, e, h\}$. Let f^Γ be a PPT rule and let Γ be the associated entitlement. Let a preference profile R be given as follows:

R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8
a	b	h	a	a	a	e	h
c	e	e	c	h	h	b	e
h	c	a	e	e	b	c	c
b	h	c	b	b	e	h	a
e	a	b	h	c	c	a	b

Stage 1:

State $s^0 = (\emptyset, ..., \emptyset)$.

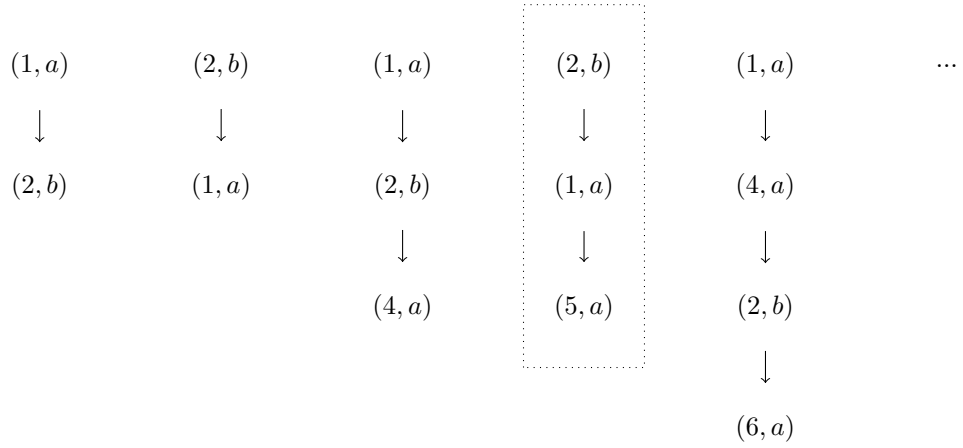
Assigned agents: $N(s^0) = \emptyset$. Unassigned agents: $\bar{N}(s^0) = \mathcal{N}$. Assigned projects: $Z(s^0) = \emptyset$. Partially assigned projects: $\hat{Z}(s^0) = \emptyset$. Unassigned projects: $\bar{Z}(s^0) = \mathcal{Z}$. Number of remaining projects: $m'(s^0) = 4$.

[8] We will show that at least one agent receives her assignment in every stage. Since the number of agents is finite, this guarantees that the procedure terminates in a finite K.

Entitlements		
$g(s^0)$	$T(s^0)$	$E(s^0)$
$\{1,2\}$	$((1,2),(2,1)$ $(1,2,4),(2,1,5),$ $(1,4,2,6),(2,5,1,4),$ $(1,4,6,2,8),(2,5,4,1,7))$	$\emptyset$

Trading Step: Since there are no endowments in this stage, we do not run the TTC.

Proposal Step: Drawing the proposals, we get:



There is an acceptable proposal, given in the box above, and so we make the following assignments: $s_1^1 = a, s_2^1 = b, s_5^1 = a$. For other agents i, we set $s_i^1 = s_i^0$.

Representing these assignments in the profile:

R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8
$\boxed{a}$	$\boxed{b}$	h	a	$\boxed{a}$	a	e	h
c	e	e	c	h	h	b	e
h	c	a	e	e	b	c	c
b	h	c	b	b	e	h	a
e	a	b	h	c	c	a	b

Stage 2:

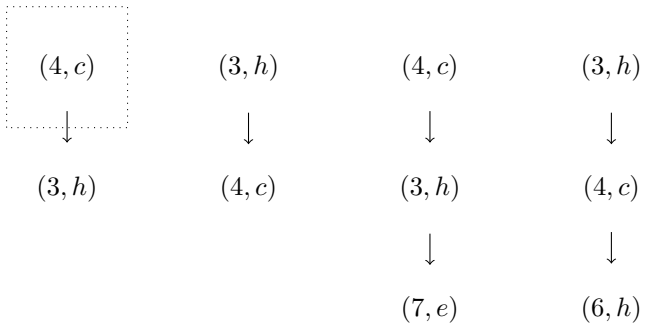
State $s^1 = (a, b, \emptyset, \emptyset, a, \emptyset, \emptyset, \emptyset)$.

Assigned agents: $N(s^1) = \{1, 2, 5\}$. Unassigned agents: $\bar{N}(s^1) = \{3, 4, 6, 7, 8\}$. Assigned projects: $Z(s^1) = \{a, b\}$. Partially assigned projects: $\hat{Z}(s^1) = \{b\}$. Unassigned projects: $\bar{Z}(s^1) = \{c, e, h\}$. Number of remaining projects: $m'(s^1) = 2$.

Entitlements		
$g(s^1)$	$T(s^1)$	$E(s^1)$
$\{4,3\}$	$((4,3),(3,4)$ $(4,3,7),(3,4,6))$	$4 \to b$

Trading Step: We have that $b \in \hat{E}_4(s^1)$, but since agent 4 points to project c which is still unassigned, there is no cycle at this stage. So we make no assignments via TTC.

Proposal Step: Drawing the proposals, we get:



There are no acceptable proposals, so the only assignment at this stage is to give the first proposer, agent 4, her label: $s_4^2 = c$. For other agents i, we set $s_i^2 = s_i^1$.

Representing these assignments in the profile:

R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8
$\boxed{a}$	$\boxed{b}$	h	a	$\boxed{a}$	a	e	h
c	e	e	$\boxed{c}$	h	h	b	e
h	c	a	e	e	b	c	c
b	h	c	b	b	e	h	a
e	a	b	h	c	c	a	b

Stage 3:

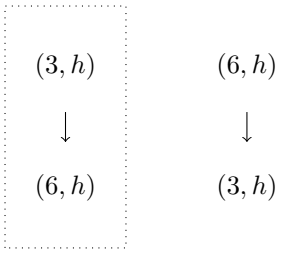
State $s^2 = (a, b, \emptyset, c, a, \emptyset, \emptyset, \emptyset)$.

Assigned agents: $N(s^2) = \{1, 2, 4, 5\}$. Unassigned agents: $\bar{N}(s^2) = \{3, 6, 7, 8\}$. Assigned projects: $Z(s^2) = \{a, b, c\}$. Partially assigned projects: $\hat{Z}(s^2) = \{b, c\}$. Unassigned projects: $\bar{Z}(s^2) = \{e, h\}$. Number of remaining projects: $m'(s^2) = 1$.

Entitlements		
$g(s^2)$	$T(s^2)$	$E(s^2)$
$\{3, 6\}$	$((3, 6), (6, 3))$	$8 \to b, 7 \to c$

Trading Step: We have that $b \in \hat{E}_8(s^2)$ and $c \in \hat{E}_7(s^2)$ but since agents 7 and 8 point to projects e and h, respectively, which is still unassigned, there is no cycle at this stage. So we make no assignments via TTC.

Proposal Step: Drawing the proposals, we get:



The first proposal is acceptable, so we make the following assignments: $s_3^3 = h, s_6^3 = h$. For other agents i, we set $s_i^3 = s_i^2$.

Representing these assignments in the profile:

R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8
$\boxed{a}$	$\boxed{b}$	$\boxed{h}$	a	$\boxed{a}$	a	e	h
c	e	e	$\boxed{c}$	h	$\boxed{h}$	b	e
h	c	a	e	e	b	c	c
b	h	c	b	b	e	h	a
e	a	b	h	c	c	a	b

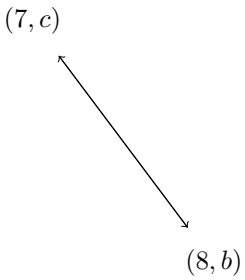
Stage 4:

State $s^3 = (a, b, h, c, a, h, \emptyset, \emptyset)$.

Assigned agents: $N(s^3) = \{1, 2, 3, 4, 5, 6\}$. Unassigned agents: $\bar{N}(s^3) = \{7, 8\}$. Assigned projects: $Z(s^3) = \{a, b, c, h\}$. Partially assigned projects: $\hat{Z}(s^3) = \{b, c\}$. Unassigned projects: $\bar{Z}(s^3) = \{e\}$. Number of remaining projects: $m'(s^3) = 0$.

Entitlements		
$g(s^3)$	$T(s^3)$	$E(s^3)$
$\emptyset$	$\emptyset$	$8 \to b$
		$7 \to c$

Trading Step: The TTC procedure gives us the following:



Agent 7 points to agent 8, agent 8 points to agent 7, so we have a cycle between agents 7 and 8. So they trade projects and we make the following assignment: $s_7^4 = b, s_8^4 = c$. For other agents i, we set $s_i^4 = s_i^3$.

This is a terminal state, so we have our final assignment (a, b, h, c, a, h, b, c). Representing these assignments in the profile:

R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8
$\boxed{a}$	$\boxed{b}$	$\boxed{h}$	a	$\boxed{a}$	a	e	h
c	e	e	$\boxed{c}$	h	$\boxed{h}$	$\boxed{b}$	e
h	c	a	e	e	b	c	$\boxed{c}$
b	h	c	b	b	e	h	a
e	a	b	h	c	c	a	b

7 Features of the PPT Rule and Special Cases

7.1 Similarities with Inheritance Rules

In the single-unit object allocation model, Pápai (2000) characterises the set of group-strategy-proof, Pareto optimal and reallocation proof rules via a class of rules called hierarchical exchange rules. As discussed previously, hierarchical exchange rules determine the initial endowment of each object to some agent. Assignments

are made using the TTC procedure in rounds. Hierarchical exchange rules also specify the inheritance of unassigned objects to other agents as a function of the structure of previous assignments. Each object is owned by some agent, and agents may own more than one object. Once they have received their assignment, the remaining objects in their endowments get transferred to (or inherited by) some other agent.

One important sub-class of hierarchical exchange rules is what Pápai (2000) calls a fixed endowment exchange rule. In such a rule, the inheritance structure depends only on which agents have received their assignments and what assignment they have received. Such a rule also covers the serial and sequential priority rules.

Hierarchical exchange rules in general satisfy two properties called the Assurance Rule and the Twin Inheritance Rule. The Assurance Rule guarantees that an object that is in an agent's endowment remains in her endowment at any later stage in which she is yet to receive an assignment. The Twin Inheritance Rule says that if there are two preference profiles such that the hierarchical exchange rule at a certain stage results in the same agents with the same preferences being assigned the same objects, then the endowments determined at that stage for other agents must also be the same in the two cases.

In our model, the endowment structure satisfies these two properties. Condition E4 on endowments captures the fact that an agent retains every project in her endowment when the other agent that is partially assigned that project remains the same, and when this agent is still unassigned. This is the Assurance Rule in other words.

Furthermore, because our entitlement specification does not depend on the name of projects, and instead with the identity of agents and whether their assignments are distinct or otherwise, it is easy to see that our rule also satisfies the Twin Inheritance Rule. That is, for any two states where the assignments and preferences of previous agents are the same, the entitlement specification for those states will also be the same.

Thus there is a connection between PPT rules and fixed endowment inheritance rules, though they operate in different environments.

7.2 The Serial Dictatorship as a Special Case

In the classical model, where each object may be assigned to at most one agent, the serial dictatorship (or serial priority) rule works as follows: There is an exogenous and fixed ordering of agents σ such that agents sequentially select projects in that order ($\sigma(1)$ selects first, $\sigma(2)$ goes next, and so on). Each agent selects her top-ranked projects from the ones that are available, given the choices of earlier agents in the sequence. It is easy to see that for any preference profile, the first agent always gets her top-ranked object, while the second agent always gets her top-ranked object whenever it is distinct from the selection of the first agent, and so on.

Extending this rule to our context is straightforward. However, the presence of copies in our model means that the feasible set of projects available to later agents in the sequence may be larger than in the classical case. So the serial dictatorship rule f^{SD} in our model works as follows: There is an exogenous and fixed ordering of agents σ. The first agent in σ will always get to select her top-ranked project. But the second agent in σ may select not only a project that differs from the first agent, but also may select the copy of the project that is assigned to the first agent, because it is still available. In general, agents' choices will affect the feasible set for subsequent agents not only via restricting the available set of *different* projects, but also the availability of copies of projects already selected.

A PPT rule f^{Γ} incorporates this special case in the following manner. Suppose with no loss of generality that there is an exogenous ordering of agents σ such that $\sigma(i)$ 'precedes' $\sigma(j)$ whenever $i < j$, where i and j are agents in $\mathcal{N}$. For any interim state s, let $g(s) = (g_1(s), g_2(s))$ be such that $g_1(s) = min\{j : j \in \bar{N}(s)\}$ and $g_2(s) = min\{k : k \in \{\bar{N}(s) \setminus \{j\}\}\}$.

Moreover, let the proposal vector always be of the form where the subsequent agents in each proposal are always indexed from minimum to maximum.

Then for any preference profile R, it is easy to see that the rule follows the procedure of the serial dictatorship, and $f^{\Gamma}(R) = f^{SD}(R)$.

8 Axioms for Pairwise Project Allocation Rules

We describe below the axioms that we impose on P-PARs, and present some key initial results. Most of these axioms are standard in the literature.

Strategy-proofness is a condition which requires truth-telling to be a dominant strategy for all agents. In other words, given the reports of all other agents, an agent must be as well off reporting her true preferences as any other preferences. When this is true for all agents and all preferences, the mechanism is said to be strategy-proof. Formally:

Axiom **1.** A P-PAR f is *strategy-proof (SP)* if, for all preference profiles R, all agents $i \in \mathcal{N}$, and all preference orderings R_i':

$$f_i(R) R_i f_i(R_i', R_{-i})$$

The next axiom, which we call the limited influence axiom, identifies conditions under which an agent may not affect another agent's assignment. The axiom has two parts. The first part is identical to the non-bossiness axiom that is pervasive in the literature on assignment rules. The condition was introduced by Satterthwaite and Sonnenschein (1981) and requires that an agent not be able to affect other agents' outcomes without affecting her own.

The second part of the limited influence axiom is new. It states that if an agent cannot obtain a particular project that she desires over her assignment (given other agents' preferences) then she cannot influence the assignment of that project. We first state the axiom formally and discuss it below.

Axiom **2.** A P-PAR f satisfies *limited influence (LIN)* if:

1. (LIN1) (Non-bossiness (NB)) For all preference profiles R, all agents $i \in \mathcal{N}$, and all preference orderings R_i':

$$[f_i(R_i', R_{-i}) = f_i(R)] \implies [f(R_i', R_{-i}) = f(R)]$$

2. (LIN2) For all preference profiles R, all projects $a \in \mathcal{Z}$, all agents $i \in \mathcal{N}$ such that $aP_i f_i(R)$, and for all R_i' such that $f_i(R_i', R_{-i}) \neq a$:

$$[f_j(R) = a] \implies [f_j(R_i', R_{-i}) = a] \text{ for all } j \in \mathcal{N}$$

Limited influence merits more discussion. LIN1, which is essentially non-bossiness, negates any effect that an agent can have on other agents' assignments in cases where she does not change her own assignment. Its main justification is that it keeps the distribution of influence in the allocation process from unduly depending on any one agent. Another justification has to do with its strategic effects. Also, its original use by Satterthwaite and Sonnenschein (1981) is on the basis of considerations of informational simplicity. Non-bossiness disqualifies rules in exchange economies that "assign all the resources to one or the other of two agents depending upon some arbitrary feature of some third agents preferences."[9]. However, Thomson (2014) also notes that its main value is in providing technical support for characterisation results.

LIN2 negates any effect that an agent can have on the assignment of a project that she also desires but cannot obtain. Suppose that there is an agent and a project she desires. For whatever reason, she is not assigned this project. Then for any reported preference in which she does not get assigned that project, it should not be the case that she can influence who else gets or does not get that project. In a sense, it says that if the agent does not have the 'rights' to that project, then she should not be able to influence who else has those rights.

Both LIN1 and LIN2 have the most effect when used in conjunction with strategy-proofness (Thomson (2013)). To see this, suppose that an agent is unable to affect his assignment to his advantage by misrepresenting his or her preferences. Then strategy-proofness is met for this agent. Such a misrepresentation may

[9]See Thomson (2014)

yet affect some other agents assignment. If this other agent benefits from it, there is an incentive for the second agent to approach the first agent and suggest the manipulation. LIN1 applies to cases where an agent is unable to change her own assignment at all, while LIN2 applies to cases where the misrepresentation does not yield the desired project (which is ensured by strategy-proofness).

LIN2 is very similar to a condition used by Pápai (2000). In fact, LIN1 and LIN2 play an important role in the characterisation of inheritance rules. In that paper, LIN2 is not assumed directly, and instead emerges as a consequence of the combination of strategy-proofness, non-bossiness and an additional condition called reallocation-proofness. Reallocation-proofness "rules out the possibility that two individuals can gain by jointly manipulating the outcome and swapping objects ex post, when the collusion is self-enforcing in the sense that neither party can lose by reporting false preferences in case the other party does not adhere to the agreement and reports honestly".[10] The corresponding version of reallocation-proofness in this model is complicated, and so we do not use reallocation-proofness. We instead take LIN2 to be a primitive requirement of the rule.

There are important single-unit object allocation rules in the literature that satisfy LIN1 and LIN2 and others that do not. As mentioned above, inheritance rules satisfy both axioms. Therefore so do sequential and serial priority rules. However, the Deferred Acceptance (DA) rules and their generalisations typically do not satisfy either condition. It is possible for an agent in the DA rule to affect the assignment of an object even when she cannot obtain it herself, whether she changes her assignment as a result or not. Thus it is a non-trivial condition to impose on project allocation rules. Moreover, the two conditions are independent, as we demonstrate by examples in Section 8.1.

Group-strategy-proofness is a stronger condition than strategy-proofness. It ensures that groups of agents do not have profitable deviations, i.e., if a group of agents deviates by reporting different preferences, then a group-strategy-proof rule ensures that it is not the case that all agents in the deviating group are at least as well off as before, and some agent strictly better off. Formally, a P-PAR f is *group-strategy-proof* if, for all profiles R, there does not exist a set of agents $M \subseteq \mathcal{N}$, and a preference sub-profile R'_M, such that $f_i(R'_M, R_{-M})R_i f_i(R)$ for all $i \in M$, and $f_j(R'_M, R_{-M})P_j f_j(R)$ for some $j \in M$.

In a wide class of assignment models, including ours, group-strategy-proofness is equivalent to the combination of strategy-proofness and non-bossiness. We reproduce the proof from Pápai (2000) here.

LEMMA **1.** *A P-PAR is group-strategy-proof if and only if it is strategy-proof and non-bossy.*

Proof: It is clear that group-strategy-proofness implies strategy-proofness (let the group size be unity). To see that it implies non-bossiness as well, consider a preference profile R, agents $i, j \in \mathcal{N}$, and preferences R'_i such that $f_i(R'_i, R_{-i}) = f_i(R)$ but $f_j(R'_i, R_{-i}) \neq f_j(R)$. Since preferences are strict, either $f_j(R'_i, R_{-i})P_j f_j(R)$ or $f_j(R)P_j f_j(R'_i, R_{-i})$. In the first case, agents i, j can manipulate at R via (R'_i, R_j), and in the second case, agents i, j can manipulate at (R'_i, R_{-i}) via (R_i, R_j). In either case, group-strategy-proofness is violated.

To show the converse, let f satisfy SP and NB. Consider a subset of agents M, a preference profile R and a sub-profile R'_M, such that for all $i \in M$, we have that $f_i(R'_M, R_{-M})R_i f_i(R)$. For each $i \in M$, consider a preference ordering $\hat{R}_i$ such that we move her assignment $f_i(R'_M, R_{-M})$ to the top of her preference $\hat{R}_i$, and leave the other projects ranked the same as they are in R_i. By SP, $f_i(\hat{R}_i, R_{-i}) = f_i(R)$. Hence by NB, $f(\hat{R}_i, R_{-i}) = f(R)$. Repeating for all agents in M, we have that $f(\hat{R}_M, R_{-M}) = f(R)$. Also, by SP and NB, $f(\hat{R}_M, R_{-M}) = f(R'_M, R_{-M})$. So $f(R'_M, R_{-M}) = f(R)$, and f is group-strategy-proof. ■

Thus by requiring our rule to satisfy strategy-proofness and limited influence, we ensure that the rule is group-strategy-proof.

For any preference P_i and any subset $X \subseteq \mathcal{Z}$, let $top(P_i, X)$ denote the top project in X according to P_i. When we mean the top-ranked project from the full set $\mathcal{Z}$, we will often suppress the set notation and refer to it simply as $top(P_i)$. Correspondingly, for a preference profile P, let $top(P, X)$ denote the N-dimensional vector of top-ranked projects in X according to preferences in P. Similarly, $top(P)$ is the vector of top-ranked projects in $\mathcal{Z}$ according to preferences in P.

[10] See Pápai (2000).

Unanimity is a full-range condition which says that if agents' preferences are such that it is feasible to give each agent her top-ranked project, then the mechanism must do so. In our model, this means that if the vector of top-ranked preferences in a particular profile is such that agents are naturally divided into m pairs, then a mechanism satisfying unanimity must prescribe those exact pairs. Formally:

AXIOM **3.** A P-PAR f is *unanimous (U)* if, for all preference profiles R:

$$[top(R) \in \mathcal{A}] \implies [f(R) = top(R)]$$

A stronger condition than unanimity is Pareto efficiency. We say that an assignment is Pareto efficient if it is not possible to make an agent strictly better off while keeping all agents at least as well off as earlier. Formally, a P-PAR f is *Pareto efficient* if, for any preference profile R, there is no feasible assignment $x \in \mathcal{A}$ such that $x_i R_i f_i(R)$ for all $i \in \mathcal{N}$, with $x_j P_j f_j(R)$ for some $j \in \mathcal{N}$.

The combination of strategy-proofness, limited influence and unanimity gives us Pareto efficiency. Pápai (2001) proves a very similar result using a condition called citizen sovereignty, which is weaker than unanimity.

LEMMA **2.** *A P-PAR satisfying strategy-proofness, non-bossiness and unanimity is Pareto efficient.*

Proof: Let f be strategy-proof, non-bossy and unanimous, and let R be a preference profile. Suppose $f(R)$ is not a Pareto efficient assignment. Then there exists some feasible assignment $x \in \mathcal{A}$ such that $x_i R_i f_i(R)$ for all $i \in \mathcal{N}$ and $x_j P_j f_j(R)$ for some $j \in \mathcal{N}$. Construct a preference profile $\hat{R}$ from R such that for all agents i, x_i is ranked top in $\hat{R}_i$ while other projects are ranked the same as in R_i.

Consider some agent i and the profile $(\hat{R}_i, R_{-i})$. By strategy-proofness, $f_i(\hat{R}_i, R_{-i}) = f_i(R)$, and so $f(\hat{R}_i, R_{-i}) = f(R)$ by non-bossiness. Repeating for all agents, we have that $f(\hat{R}) = f(R)$. By construction, $top(\hat{R}) = x$. Since $x \in \mathcal{A}$, by unanimity we have that $f(\hat{R}) = x$. But $x \neq f(R)$. This is a contradiction. Hence $f(R)$ is Pareto efficient. Since R was arbitrary, $f(R)$ is Pareto efficient for all R and so f is a Pareto efficient rule. ■

Neutrality ensures that a rule treats all projects symmetrically, and does not distinguish between them on the basis of their names. That is, if for a particular preference profile we were to perform a swap operation on a pair of projects, exchanging their positions in each agent's preferences, then such a swap must reflect exactly in the final assignments as well. When this is true for all projects and all profiles, we say a mechanism is neutral. Formally:

AXIOM **4.** A P-PAR is *neutral (NEU)* if, for all preference profiles R and all permutations[11] π of $\mathcal{Z}$:

$$f(\pi R) = \pi f(R)$$

8.1 INDEPENDENCE OF AXIOMS

To show that these axioms are independent, we now provide examples of rules satisfying all but one of the axioms in turn.

Strategy-proofness: Consider a rule that operates like a PPT rule with the following modification. There are three agents $\{i, j, k\}$ such that for any preference profile R, if $top(R_i) = top(R_j)$ then the pairing for the null state is (i, j) and is (i, k) otherwise. Also, j is not a proposer in the proposal vector at any interim state s. Let R be a profile where $top(R_i) = top(R_k) = a$, $top(R_j) = b$ and a is ranked second in R_j, and a and b are the last two projects in the preferences of all other agents. Then $aP_j f_j(R)$. But j can manipulate via a preference R'_j in which a is ranked top, since $f_j(R'_j, R_{-j}) = a$. This rule violates strategy-proofness. It is easy to check that it satisfies limited influence, unanimity and neutrality.

[11] A permutation applied to a collection of objects X is a bijection $\pi : X \to X$ that associates each object in X with a unique object in X (possibly itself). In our case, we use it to mean a relabelling of projects such that a collection of projects exchange their names. For example, under π, project a may now be called project b ($\pi(a) = b$), which is now called project c ($\pi(b) = c$), which in turn is called project a ($\pi(c) = a$). The permutation π applied to a preference profile R (written as πR) or an assignment vector x (written as πx) permutes the projects in the preferences or the assignment according to the permutation applied to the underlying set of projects $\mathcal{Z}$.

Limited influence 1: Consider a rule like the PPT rule, but with the following modification. If the pairing specified in the null state is assigned the same project, and if their second-ranked projects are the same, then the pairing for the next state is (i, j), whereas if the second-ranked projects in their preferences differ, then the pairing for the next state is (k, l), where $i \neq j \neq k \neq l$. Like the PPT rule, this rule satisfies strategy-proofness, unanimity, neutrality and limited influence 2, but violates LIN1, i.e., is bossy.

Limited Influence 2: Let $N = 6$ and $q = 2$. Consider a rule that works like the PPT rule, with the following modifications. Let R be a profile. The pairing for the null state is $(1, 2)$. If $top(R_1) \neq top(R_2)$, we look at agent 3's preferences. If $top(R_3) = top(R_1)$ and the second-ranked project is R_3 is distinct from $top(R_2)$, then the proposal for agent 1 is $(1, 2, 4)$. If $top(R_3) = top(R_1)$ and the second-ranked project is R_3 is the same as $top(R_2)$, then the proposal for agent 1 is $(1, 2, 5)$. We specify the rest of the entitlements suitably.

Consider the preference profiles R and R' given below. The assignments are given in boxes.

R

1	2	3	4	5	6
a	**b**	a	a	**a**	a
c	d	**b**	**d**	d	**d**
d	c	c	b	c	c
b	a	d	c	b	b

R'

1	2	3	4	5	6
a	**b**	a	**a**	a	a
c	d	**c**	d	d	d
d	c	b	b	**c**	c
b	a	d	c	b	**b**

Note that from R to R' only agent 3's preferences change. She does not get a which she prefers to her assignment, but still affects the assignment of a (to agent 5 in R and agent 4 in R'). Thus this rule violates LIN2. It is easy to see that it satisfies strategy-proofness, LIN1, unanimity and neutrality.

Unanimity: Consider a rule[12] that divides agents into fixed pairs $M_1, ..., M_m$ and indexes one agent $i(M_i)$ in each pair. For any preference profile R, run the sequential priority rule with agents $i(M_i)$. Whatever project they select becomes the assignment of the corresponding pair M_i. This rule violates unanimity. It is easy to check that it satisfies strategy-proofness, limited influence and neutrality.

Neutrality: Consider a rule that works like the PPT rule, with the following modification. There is a set of agents $\{i, j, k\}$ and projects a, b such that for any profile R with $top(R_i) = a$, the pairing for the null state is (i, j), and for any profile R' such that $top(R'_i) = b$, the pairing specified for the null state is (i, k). This rule violates neutrality. It is easy to check however that it satisfies strategy-proofness, limited influence and unanimity.

9 Characterisation Result

We are now ready to state our main characterisation theorem.

Theorem **1.** *A pairwise-project assignment rule is strategy-proof, unanimous, neutral and satisfies limited influence if and only if it is a pairwise priority trading rule.*

The proof is in the appendix. Here we provide the intuition behind the arguments of the proof.

[12] This rule is identical to the one proposed by Rhee (2011).

9.1 Sufficiency

We begin by proving two lemmas. The first says that if there is an agent and a project she desires over her assignment at a profile, then this project must either have been fully assigned by the PPT rule at an earlier stage, or if this project is not assigned at all, then m different projects must have been at least partially assigned by the time she gets her assignment. The second lemma shows that an agent cannot affect the assignment of any agent who receives her project in a stage before when this agent is assigned her project. We use these lemmas to show that the PPT rule satisfies the axioms.

- Strategy-proofness: Suppose for a profile there is an agent who prefers some project to her own assignment. Either this project is assigned at that profile or it is not. If it is, then it must have been fully assigned at a stage before when this agent gets her assignment. If it is not assigned to anyone, then m different projects must have already been assigned by that stage. Since no agent can affect the assignment of any agent who is assigned a project at an earlier stage, a unilateral deviation on the part of this agent will not get her the project, and SP is satisfied.

- Limited influence 1: It can easily be seen that no agent can be bossy with another agent who receives her assignment at an earlier stage or the same stage. We show that no agent can be bossy with an agent who receives her assignment at a later stage. This involves showing that the entitlement at this stage is a function only of the agents who receive their assignments and the projects they receive. As long as this agent receives the same project, the entitlement remains the same. Thus the assignments at the next stage are independent of this agent. And the same is true for all later stages as well. Thus the earlier agent cannot affect the assignment of the later agent.

- Limited influence 2: If there is a project that this agent prefers, then it is being assigned at an earlier stage than when she gets her assignment. Thus she cannot affect the assignment of this project to earlier agents even if she receives a different project later. The same is true if the project is not assigned to anyone at all.

- Unanimity: The PPT rule makes assignments based on the top-ranked projects of agents among projects not fully assigned. Thus for a unanimous profile, agents are only ever assigned their top-ranked projects, and so the allocation must be the vector of top-ranked projects.

- Neutrality: It is easy to see that neither the entitlement nor the iterative procedure of the PPT rule depends on the identity of the project. Thus for any profile R and any reshuffling of the names of projects, the allocation must incorporate the reshuffling as well.

9.2 Necessity

To prove the converse, we have to show that any rule satisfying the axioms is a PPT rule. This requires two steps: One, we construct the entitlements for any state. Two, we show that assignments are made for any preference profile via the iterative procedure using those entitlements.

First we construct the entitlements for any state.

We show that for any interim state there is a pair of agents that can guarantee the assignment of any unassigned project among themselves by declaring it their top option. The proof begins by showing that for an identical preference profile, Pareto efficiency implies that some pair gets the top-ranked project. We then show that strategy-proofness and limited influence mean that no other agent can affect the assignment of this pair. Neutrality then implies that this is true for all unassigned projects. Since our selection of interim state was arbitrary, this allows us to construct the pairing for all such states.

Next we have to build proposals. We prove a lemma that allows us to trace the sequence in which agents may join a proposal. We use the lemma and proposition to construct the proposals for any state. We also order the proposals, which gives us the proposal vector for each state.

After this, we build the endowments. For any interim state, we show that any project assigned to only one agent must have another agent who can claim that project if she desires it. Repeating for all projects, we are able to generate the endowments for that state.

Collectively, this gives us the entitlement for that state. We can repeat the above process for all states to generate the complete entitlement.

We prove a lemma that states that any pair of agents that can guarantee a project between them must have at least one of them receiving a weakly preferred project to that one. This allows us to specify what happens when there are no acceptable proposals in a stage.

Finally, we show that the rule satisfying the axioms must behave like a PPT rule. That is, using the entitlements generated above, we show that the rule prescribes assignments in stages in a manner consistent with that specified by the PPT rule. This completes the proof.

10 Conclusion

In this paper we have developed a rule to make assignments in the pairwise project allocation framework. We have shown that this PPT rule specifies a class that is characterised by the properties of strategy-proofness, limited influence, unanimity and neutrality. In what follows, we discuss some possible extensions of this model.

The first target of any extension of this rule will be from pairwise allocation to arbitrary group sizes, i.e., where each project must be assigned to exactly q agents or to nobody, where $1 \leq q \leq |\mathcal{N}|$.[13]

Note that assuming $q = 1$ returns us to the classical object allocation setup. Following from the work of Svensson (1999), it is straightforward to show that our rule would translate into serial dictatorships. At the other extreme, when $q = |\mathcal{N}|$, such that each project must be assigned either to everyone or no one, we return to the public goods setting, and our rule will become a dictatorship in the sense of Gibbard (1973) and Satterthwaite (1975).

Thus the problem effectively becomes one of determining what happens when $2 < q < |\mathcal{N}|$. In principle, this would involve expanding the components of the entitlement and then refining the iterative procedure to handle larger groups.

As far as expanding entitlements is concerned, some tasks are easier than others. The definition of a pairing can easily be generalised to account for larger group sizes that can commonly guarantee a top-ranked project. The existence of such a larger group for any interim state is also easy to show.

One difficulty lies in specifying proposals. In our model, agents can 'join' other agents who has received an assignment only one at a time. The structure of a proposal is straightforward therefore - it is a sequence of agents. When considering larger groups, however, there is more than one agent who can join any agent who receives a project. In fact, more than one agent *must* join any agent who receives a project. Thus the notion of a proposal cannot be used directly. We have some insight but not a clear picture about how this would work.

Difficulties also arise when considering a possible TTC round. More than one agent may hold in her endowment a copy of a partially assigned project. These copies are notionally identical. Thus if there is some other agent who desires this project in a TTC round, she has more than one option of agent to point to. Determining which trade, if any, is honoured in this case is a non-trivial exercise. The presence of possibly multiple overlapping cycles will require the use of a tie-breaker or the introduction of some other component of an entitlement. We have not found a way around this problem yet.

Other extensions, even in the pairwise framework, would be to obtain a full characterisation of group-strategy-proof and Pareto optimal rules. This will require dropping the axioms of neutrality and LIN2. Dropping neutrality will spread the pairwise 'ownership' of each project to possibly a different pair of agents. Dropping LIN2 leaves the rule in a possibly endogenous situation, where the preferences of other agents could influence the initial pairwise ownership of each project. These remain open problems.

[13] Another related extension would be to drop the requirement that each project have the same exact capacity constraint.

References

Abdulkadiroğlu, A. and T. Sönmez (1999): "House allocation with existing tenants," *Journal of Economic Theory*, 88, 233–260.

Barberà, S. (1983): "Strategy-proofness and pivotal voters: a direct proof of the Gibbard-Satterthwaite Theorem," *International Economic Review*, 24, 413–417.

Bird, C. G. (1984): "Group incentive compatibility in a market with indivisible goods," *Economics Letters*, 14, 309–313.

Ehlers, L., I. E. Hafalir, M. B. Yenmez, and M. A. Yildirim (2011): "School choice with controlled choice constraints: Hard bounds versus soft bounds," Mimeo.

Fragiadakis, D., A. Iwasaki, P. Troyan, S. Ueda, and M. Yokoo (2012): "Strategy-proof matching with minimum quotas," Mimeo.

Gibbard, A. (1973): "Manipulation of voting schemes: A general result," *Econometrica*, 41, 587–601.

Hatfield, J. W. (2009): "Strategy-proof, efficient, and non-bossy quota allocations," *Social Choice and Welfare*, 33, 505–515.

Hylland, A. and R. Zeckhauser (1979): "The efficient allocation of individuals to positions," *The Journal of Political Economy*, 87, 293–314.

Ma, J. (1994): "Strategy-proofness and the strict core in a market with indivisibilities," *International Journal of Game Theory*, 23, 75–83.

Pápai, S. (2000): "Strategyproof assignment by hierarchical exchange," *Econometrica*, 68, 1403–1433.

——— (2001): "Strategyproof and non-bossy multiple assignments," *Journal of Public Economic Theory*, 3, 257–271.

Pycia, M. and M. U. Ünver (2013): "Incentive-compatible allocation and exchange of discrete resources," Mimeo.

Rhee, S. (2011): "Strategy-proof allocation of individual goods among couples," *Japanese Economic Review*, 62, 289–303.

Roth, A. (1982): "Incentive compatibility in a market with indivisible goods," *Economics Letters*, 9, 127–132.

Roth, A. and A. Postlewaite (1977): "Weak versus strong domination in a market with indivisible goods," *Journal of Mathematical Economics*, 4, 131–137.

Satterthwaite, M. (1975): "Strategy-proofness and Arrow's Conditions: Existence and correspondence theorems for voting procedures and social welfare functions," *Journal of Economic Theory*, 10, 187–216.

Satterthwaite, M. and H. Sonnenschein (1981): "Strategy-proof allocation mechanisms at differentiable points," *Review of Economic Studies*, 48, 587–597.

Sen, A. (2001): "Another direct proof of the Gibbard-Sattherthwaite Theorem," *Economics Letters*, 70, 381–385.

Shapley, L. and H. Scarf (1974): "On cores and indivisibility," *Journal of Mathematical Economics*, 1, 23–37.

Svensson, L.-G. (1999): "Strategy-proof allocation of indivisible goods," *Social Choice and Welfare*, 16, 557–567.

Thomson, W. (2013): "Strategy-proof allocation rules," Mimeo, University of Rochester.

——— (2014): "On the axiomatics of resource allocation: Interpreting nonbossiness," Mimeo, University of Rochester.

11 Appendix: Proof of Theorem 1

In this section we present the proof of our main characterisation theorem.

11.1 Sufficiency Proof

We must show that the PPT rule satisfies the axioms.

Lemma **3.** *Let f^Γ be the PPT rule, let R be a profile, let i be some agent and $a \in \mathcal{Z}$ a project such that $aP_i f_i^\Gamma(R)$. Suppose i receives her assignment in stage k. Then, by some stage $k' < k$, either (1) a is fully assigned, or (2) m different projects are assigned to other agents.*

Proof: Let f^Γ be the PPT rule, let R be a profile, let i be some agent and $a \in \mathcal{Z}$ a project such that $aP_i f_i^\Gamma(R)$. Note that at any stage of the PPT rule, an agent gets her assignment by pointing to the top-ranked project among those that are available, whether as a part of a proposal vector or endowment. Suppose i receives her assignment $f_i(R)$ in stage k. She gets it by pointing to $f_i(R)$. By the properties of the PPT rule, this means that project a is no longer available in stage k otherwise she would be pointing to it instead. Thus if project a is assigned to other agents, it must have been fully assigned by at most stage $k-1$. Alternatively, project a is no longer available because m other projects have been assigned. Again, this must have happened at most by stage $k-1$. ■

Given a preference profile R and agents i and j, we say that agent i *affects the assignment of agent j* if, for some R_i', we have that $f_j(R_i', R_{-i}) \neq f_j(R)$.

Lemma **4.** *Let f^Γ be a PPT rule. For any profile R, consider agent i and let k be the stage in which she receives her assignment. Then i cannot affect the assignment of any agent j who receives her assignment in a stage earlier than k.*

Proof: The null state entitlement is given exogenously and does not depend on agent i. Assignments in the first stage are based on preferences of agents that receive their assignments in that stage. Agent i cannot influence their preferences, and so she cannot influence their assignments. Thus she cannot affect the state after the first stage, and cannot affect the entitlements in the second stage either. Let the state at any stage $k' < k$ be given. Assignments are based on preferences of agents other than i. Agent i cannot affect their preferences or assignments. Thus the state in $k'+1$ is given independently of i. So are entitlements. By induction this is true for all $k' < k$. So agent i cannot affect any assignments made in stages before when she gets her assignment. ■

Strategy-proofness: Let R be a profile, a some project and i an agent such that $aP_i f_i(R)$. Suppose i gets her assignment in stage k. There are two possibilities: (1) $f_M(R) = a$ for some M. Then by Lemma 3 agents in M are assigned project a before stage k. (2) $f_j(R) \neq a$ for all $j \in \mathcal{N}$. By Lemma 3, this means that m distinct projects have been assigned by stage k. Since by Lemma 4, agent i cannot affect the entitlement at any earlier stage, she cannot get a for any preference R_i'. Thus in each case a unilateral deviation will not get her the project, and SP is satisfied.

Limited influence 1: It can easily be seen that no agent can be bossy with another agent who receives her assignment at an earlier stage or the same stage, since by Lemma 4 she cannot affect the assignments

at earlier stages. To see that an agent cannot affect the assignment of later agents without changing her own assignment, see that the entitlement formed at the end of the stage where she receives her assignment depends on the identities of agents receiving their assignments in that stage and the projects they receive. She cannot affect any other agent who receives her assignment in the same stage without changing her own assignment. So if she continues to get the same project, the state at the end of the stage remains the same, and so does the entitlement. Later entitlements are independent of her preferences. So she cannot affect the assignment of any subsequent agent as long as she gets the same project. Thus f^Γ satisfies LIN1.

Limited influence 2: Let R be a profile, i some agent and some project a such that $aP_if_i(R)$. If a is assigned by the PPT rule for R, then by Lemma 3, there is some pair M and some stage k where both agents in M have received a as their assignment. Since by Lemma 4 agent i cannot affect entitlements at earlier stages, she cannot affect the assignment of this project to earlier agents as long as they desire it. This is true even if she were to receive some other project. If a is not assigned to any agent for R, then by Lemma 3, m different projects have already been assigned, and agent i cannot influence the assignment of project a. Thus f^Γ satisfies LIN2.

Unanimity: Let R be a unanimous preference profile. At any stage, an agent participating in a pairing, a proposal vector or an endowment does so via her top-ranked project among those that are not fully assigned. Thus every assignment that is made is the respective agent's top-ranked project. So $f^\Gamma(R) = top(R)$, and f^Γ is unanimous.

Neutrality: It is easy to see that neither the entitlement nor the iterative procedure of the PPT rule depends on the identity of the project. Thus for any profile R and any permutation π of $\mathcal{Z}$, we have that $f^\Gamma(\pi R) = \pi f^\Gamma(R)$ and so f^Γ is neutral.

11.2 Necessity Proof

We start by showing that at any interim state there is a pair of agents that can guarantee their own assignment of any unassigned project by commonly declaring it as their top-ranked project.

Lemma **5.** *Consider an interim state s. Fix preferences of agents in $N(s)$ as $R_{N(s)}$. Let f satisfy SP, LIN, U and NEU. Then there exists a pair of agents $M(s) \subseteq \bar{N}(s)$ such that, for any $R_{\bar{N}(s)}$ and any $a \in \bar{Z}(s)$, $[top(R_{M(s)}) = (a,a)] \implies [f_{M(s)}(R) = (a,a)]$.*

Proof: Let R be a profile such that $R_{N(s)}$ is as given and $R_i = R_j$ for all $i, j \in \bar{N}(s)$. Without loss of generality, let $top(R_{\bar{N}}(s)) = (a, a, ..., a)$, with $a \in \bar{Z}(s)$. Since f is Pareto efficient and $m'(s) \geq 1$, there must exist a pair of agents $M \subseteq \bar{N}(s)$ such that $f_M(R) = (a,a)$. Let $R'_{\bar{N}(s)\setminus M}$ be an arbitrary sub-profile for agents in $\bar{N}(s) \setminus M$. We will first show that $f_M(R_{N(s)}, R_M, R'_{\bar{N}(s)\setminus M}) = (a,a)$.

Pick a $j \in \bar{N}(s)\setminus M$ and consider the sub-profile (R'_j, R_{-j}). Since $aP_jf_j(R)$, SP implies that $f_j(R'_j, R_{-j}) \neq a$. If $f_j(R'_j, R_{-j}) = f_j(R)$ then $f(R'_j, R_{-j}) = f(R)$ by LIN1. In particular, $f_M(R'_j, R_{-j}) = (a,a)$. Instead, suppose $f_j(R'_j, R_{-j}) \neq f_j(R)$. Then $[f_M(R) = (a,a)] \implies [f_M(R'_j, R_{-j}) = (a,a)]$ by LIN2.

Repeating for all other agents in $\bar{N}(s) \setminus M$, we have that $f_M(R_{N(s)}, R_M, R'_{\bar{N}(s)\setminus M}) = (a,a)$.

It follows from SP and LIN1 that this is true for any R_M with $top(R_M) = (a,a)$. Since f satisfies NEU, this is true for all $a \in \bar{Z}(s)$. Thus, for any preference sub-profile $R_{\bar{N}(s)}$ and any $a \in \bar{Z}(s)$ with $top(R_M) = (a,a)$, we have that $f_M(R) = (a,a)$. Set $M(s) = M$ as determined above. ■

For convenience, we shall denote the pair $M(s^0)$ where s is the null state as M^*.

Pairing

In general, for any interim state s, we can use Lemma 5 to identify $M(s)$. We set $g(s) = M(s)$. It is easy to check that $g(s)$ satisfies properties G1-G3.

Next, we define the notion of the pair-option-set. Fixing a sub-profile of preferences of other agents, the *pair-option-set* of a pair M at that sub-profile is the set of projects that it can receive if both agents in the pair list it as their top preference. Formally:

DEFINITION **1.** Let M be a pair of agents, and let R_{-M} be an arbitrary sub-profile for the other agents. The *pair-option-set of* M *at* R_{-M} is denoted $o_M(R_{-M})$ such that:

$$o_M(R_{-M}) = \{a \in \mathcal{Z} | \exists R'_M : [top(R'_M) = (a,a)] \implies [f_M(R'_M, R_{-M}) = (a,a)]\}$$

We now state and prove a lemma that will be useful in constructing proposals. Lemma 6 identifies conditions under which agents in pairs with a non-empty pair-option-set can 'trade' projects with each other. In particular, if there are two pairs with non-empty pair-option sets and at least one agent claims the project associated with the opposite pair, then a rule satisfying our axioms must honour the swap. Formally:

LEMMA **6.** *Let f satisfy SP, LIN and U. Let R be a profile. For a pair of agents $M = \{i,j\}$, suppose that $top(R_i) = a$ and $top(R_j) = b$, and let $a \in o_M(R_{-M})$. If there is a pair $M' = \{k,l\}$ such that $top(R_k) = top(R_l) = a$ and $b \in o_{M'}(R_{-M'})$, then $f_j(R) = b$ and $f_k(R) = a$ or $f_l(R) = a$.*

Proof: Suppose for contradiction that $bP_j f_j(R)$. Construct R'_j such that b and a are ranked first and second, and all other projects are ranked the same as in R_j. By SP, $f_j(R'_j, R_{-j}) \neq b$. Since $a \in o_M(R_{-M})$, we have that $f_M(R'_j, R_{-j}) = a$. Note that $aP_k f_k(R'_j, R_{-j})$, $aP_l f_l(R'_j, R_{-j})$ and $b \in o_{M'}((R'_j, R_{-\{\mathcal{N}\setminus\{M'\cup\{j\}\}}}))$. Construct R'_k, R'_l such that a and b are ranked first and second respectively, and all other projects are ranked the same as in R_k, R_l respectively. By SP, $f_{k,l}(R'_{\{j,k,l\}}, R_{-\{j,k,l\}}) \neq a$.

As $b \in o_{M'}((R'_j, R_{-\{\mathcal{N}\setminus\{M'\cup\{j\}\}}}))$, it follows that $f_{M'}(R'_{\{j,k,l\}}, R_{-\{j,k,l\}}) = (b,b)$. By LIN, we have that $f_j(R'_{\{j,k,l\}}, R_{-\{j,k,l\}}) = a$. This violates PE as agent j and either k or l can swap assignments making them both strictly better off while keeping other agents as well off as before. This is a contradiction. So $f_j(R'_{\{j,k,l\}}, R_{-\{j,k,l\}}) = b$. But $b \in o_{M'}((R'_j, R_{-\{M'\setminus\{j\}\}}))$, so at least one of k,l must get something they strictly prefer to b. Without loss of generality, let $f_k(R'_{\{j,k,l\}}, R_{-\{j,k,l\}}) = a$. By SP and LIN, this means that $f_j(R) = b$ and $f_k(R) = a$. ■

PROPOSAL VECTOR

In what follows we show how to generate the proposals for an arbitrary interim state. First we define the notion of a closure, which will help us generate the proposal vector via acceptable proposals. Let X be a subset of projects and R a profile. Consider a vector of agents $I^{i_1} = (i_1, ..., i_n)$, and let $a_{i_j} = top(R_{i_j}, X)$. We say that I is a *closure for agent* i_1 *in* X if:

CL1. $n \geq 2$

CL2. $a_{i_j} \neq a_{i_k}$ for all $j,k \in \{1, ..., n-1\}$

CL3. $a_{i_{n-1}} = a_{i_n}$

CL4. $f_{i_j}(R) = a_{i_j}$ for all $i_j \in I$

Condition CL1 requires the minimum length of the vector to be 2. Conditions CL2 and CL3 stipulate that all projects for agents in the vector be distinct, except the project associated with the last two agents in the vector must be the same. And condition CL4 fixes the respective projects for each agent as their assignment at that profile. We now use the concept of a closure to build the proposals for a given agent.

Closure Generation

1. Let s be some interim state, and let $m'(s) \geq 2$.

2. By Lemma 5, there exists a pair of agents $M(s)$ such that $a \in o_{M(s)}(R_{-\bar{N}(s)\setminus M(s)})$ for all $R_{-\bar{N}(s)\setminus M(s)})$ and all $a \in \bar{Z}(s)$. Without loss of generality, let $M = \{1, 2\}$.

3. Note that, by Lemma 5, for all $R_{M(s)}$ with $top(R_1, \bar{Z}(s)) = top(R_2, \bar{Z}(s)) = a$, we must have that $f_M(s) = (a, a)$. Thus $(1, 2)$ and $(2, 1)$ are both closures by the definition above. We now show how to build larger closures where possible.

4. Consider agent 1. Consider a sub-profile $R_{\bar{N}(s)}$ where $top(R_1) = b$ and a is ranked second, and $top(R_k) = a$ for all $k \in \bar{N}(s), k \neq 1$.

5. By Lemma 6, $f_1(R) = b$. By PE, there is a pair $M' \subset \bar{N}(s)$ such that $f_{M'}(R) = a$.

   Claim **1.** For any $R_{\bar{N}(s)}$ and $c, d \in \bar{Z}(s)$, if $top(R_1) = c$ and $top(R_{M'}) = (d, d)$, then $f_1(R) = c$ and $f_{M'}(R) = (d, d)$.

   Proof: Let $R_{\bar{N}(s)}$ be as in the construction above. Then we have $f_1(R) = b$ and $f_{M'}(R) = a$. Consider any agent $k \neq 1$ and $k \notin M'$, and let R'_k be some arbitrary preference. We have that $aP_k f_k(R)$. So by SP, $f_k(R'_k, R_{-k}) \neq a$. By LIN, $f_{M'}(R'_k, R_{-k}) = (a, a)$. Suppose $f_1(R'_k, R_{-k}) \neq b$. Then by Lemma 5, $f_1(R'_k, R_{-k}) = a$. This contradicts LIN. Thus $f_1(R'_k, R_{-k}) = b$. Repeating for all other agents, we get that assignments of agent 1 and agents in M' are independent of other unassigned agents' preferences. By neutrality, this must be true for all such configurations of preferences. In particular, for any $c, d \in \bar{Z}(s)$, if $top(R_1) = c$ and $top(R_{M'}) = (d, d)$, then $f_1(R) = c$ and $f_{M'}(R) = (d, d)$. ■

6. So $(1, M')$ is a closure for agent 1, by definition. If $m'(s) = 2$, we are done for this agent. Instead, suppose $m'(s) > 2$. There are two possibilities. Either $2 \in M'$ or not.

   (a) If $2 \in M'$, with no loss of generality let the other agent in M' be agent k. Keeping other agent preferences the same, consider R'_k such that $top(R'_k, \bar{Z}(s)) = c$ with $c \notin \{a, b\}$, and a is ranked second. With a slight abuse of notation, we write R'_k as R_k. By Lemma 6, $f_k(R) = c$. Moreover, $f_1(R) = b$. By PE, there is a pair $M' \subset \bar{N}(s)$ such that $f_{M'}(R) = (a, a)$.

   (b) If $2 \notin M'$, let $M' = \{k, l\}$. Repeat step (a) above once each for each agent k, l. In each case, by Lemma 6, agent 1 gets b, agent k (or l) will get c, and there is a pair M^k (or M^l) that gets a.

   Claim **2.** For any $R_{\bar{N}(s)}$ and $c, d, e \in \bar{Z}(s)$, if $top(R_1) = c$, $top(R_k) = d$ and $top(R_{M^k}) = (e, e)$, then $f_1(R) = c$, $f_k(R) = d$ and $f_{M'}(R) = (e, e)$.

   Proof: Same argument as in Claim 1 above. ■

7. In each case, the sequence $(1, k, M^k)$ (or $(1, l, M^l)$) is a closure for 1. In this manner, we can extend each closure until the length is $m'(s) + 1$. We cannot extend further as there are only $m'(s)$ projects we can assign.

8. Repeat this entire procedure for agent $2 \in M(s)$, i.e., start with a profile where $top(R_2) = b$ and the second-ranked project is a, while the top-ranked project for all other agents is a.

9. When we have generated all the closures for each agent, we sort the closures by length, from minimum to maximum.

10. It is easy to see that properties (P1-P4) are satisfied by each closure, so each closure is a proposal.

11. It is also straightforward to show that properties (PV1-PV5) are satisfied by the collection of proposals generated above, thus this is a valid proposal vector for this state.

Endowment

Let s be a state and R be a profile. Let $E(s) = (\emptyset, ..., \emptyset)$. For each partially assigned project $a \in \hat{Z}(s)$, let i be the agent such that $s_i = a$. With no loss of generality, let $a = top(R_i)$. Construct a sub-profile $R'_{\bar{N}(s)}$ such that $a = top(R'_j)$ for all $j \in \bar{N}(s)$. It must be that $f_k(R_{N(s)}, R'_{\bar{N}(s)}) = a$ for some $k \in \bar{N}(s)$. Let $E_k(s) \ni a$. Repeating for all $a \in \hat{Z}(s)$, we have our endowment $E(s)$ for this state.

It is easy to check that the endowment $E(s)$ satisfies (E1-E3). To see that E4 is also satisfied, consider any agent and any project in his endowment. Note that from the construction above, other agents prefer the project but cannot get it. Thus by LIN they cannot affect the assignment of this project. So at all other states where this project is partially assigned to the same agent, and this agent remains unassigned, he can claim that project when he desires it.

Entitlements

For any interim state, we use the above steps to generate the pairing, the proposal vector and the endowment. We can do this for every state to give the complete entitlement.

We prove another useful lemma. Lemma 7 says that if f satisfies SP, LIN and U, then for any preference profile and any pair of agents with a non-empty pair-option-set, at least one agent in the pair must be assigned a project that she weakly prefers to her top-ranked project in the pair-option-set. Formally:

Lemma **7.** *Let f satisfy SP, LIN and U. Let R be a preference profile and let M be a pair of agents such that $o_M(R_{-M}) \neq \emptyset$. For each $i \in M$, let $a_i = top(R_i, o_M(R_{-M}))$. Then $f_j(R) R_j a_j$ for some $j \in M$.*

Proof: Let f satisfy SP, LIN and U, let R be a preference profile and let M be a pair of agents with $o_M(R_{-M}) \neq \emptyset$. Let $a_i = top(R_i, o_M(R_{-M}))$ for each $i \in M$. Define $Y = \{a_i | i \in M\}$. It follows that $1 \leq |Y| \leq 2$.

Case 1: Suppose $|Y| = 1$. Let $Y = \{b\}$, i.e., $a_i = b$ for all $i \in M$. For contradiction, suppose that $bP_i f_i(R)$ for all $i \in M$. For each $i \in M$, construct R'_i such that b is the top-ranked project in R'_i and all other projects are ranked the same as in R_i. Consider some $j \in M$. Since $bP_j f_j(R)$, by SP and LIN1 it follows that $f(R'_j, R_{-j}) = f(R)$. Repeating for the other agent in M, we get that $f(R'_M, R_{-M}) = f(R)$. In particular, $f_M(R'_M, R_{-M}) \neq (b, b)$. But $b \in o_M(R_{-M})$, so by definition $top(R'_M) = (b, b)$ implies $f_M(R'_M, R_{-M}) = (b, b)$. This is a contradiction. So $f_i(R) R_i b$ for at least one agent $i \in M$.

Case 2: Suppose $|Y| = 2$. Let $M = \{i, j\}$ and without loss of generality let $top(R_i) = a$ and $top(R_j) = b$. For contradiction, suppose that $aP_i f_i(R)$ and $bP_j f_j(R)$. Consider agent i. Construct R'_i such that a and b are ranked first and second in R'_i, and all other projects are ranked the same as in R_i. It follows from SP that $f_i(R'_i, R_{-i}) \neq a$. Since $b \in o_M(R_{-M})$, we have by SP and LIN1 that $f_M(R'_i, R_{-i}) = (b, b)$. For agent j, construct R'_j such that b and a are ranked first and second respectively, and other projects are ranked the same as in R_j. It follows from SP and LIN1 that $f_M(R'_M, R_{-M}) = (b, b)$.

Instead, consider agent j and the profile (R'_j, R_{-j}). By a symmetric argument, $f_M(R'_j, R_{-j}) = (a, a)$. And repeating for agent i, we have that $f_M(R'_M, R_{-M}) = (a, a)$. We thus have two outcomes for M at the same preference profile. Since $a \neq b$, this is a contradiction. Thus $f_i(R) R_i a$ or $f_j(R) R_j b$. ■

Corollary **1.** *For every R there is an $i \in M^*$ such that $f_i(R) = top(R_i)$.*

Proof: By Lemma 5, for every $a \in \mathcal{Z}$ and every $R_{-M_1^*}$, $a \in o_{M^*}(R_{-M^*})$. ■

Iterative Procedure

It remains to show that the assignments must work as described, in that the iterative procedure must be followed. Let R be a preference profile and let Γ be the entitlements as determined above.

Assignments at stage 1:

The null state s^0 is an interim state. There are no endowments so there is no trading at this stage. By Lemma 5 there is a pair of agents $M(s^0) \subset \mathcal{N}$ such that $[top_M(R) = (a, a)] \implies [f_M(R) = (a, a)]$. Let the agents in M be denoted $g_1(s^0), g_2(s^0)$. Thus if the first proposal (of length 2) is acceptable, then it must be their assignment. Otherwise, by the Closure Generation, the first acceptable proposal must be the assignment. Moreover, this is independent of the preferences of other agents. By Lemma 7, at least one of $g_1(s^0), g_2(s^0)$ must get her top-ranked project in this profile. Thus if there is no acceptable proposal, assign $g_1(s^0)$ her top-ranked project.

Assignments at stage $k+1$:

Let the state s^k is the partial allocation up to stage k. The entitlement $(g(s^k), T(s^k), E(s^k))$ is specified by construction. Fix the preferences and assignments of agents receiving their assignments up to and including stage k.

If $\hat{E}(s^k) = \emptyset$, no assignments are made via trading. Otherwise, for all $i \in \hat{E}(s^k)$ and $a \in \hat{E}_i(s^k)$, if $top(R_i, \bar{Z}(s^k) \cup \hat{Z}(s^k)$, then we have that $f_i(R) = a$. If there is a set of agents $(i_1, ..., i_k \equiv i_1)$, with $i_j \in \hat{E}(s^k)$ for all j, and $a_{i_j} \in \hat{E}_{i_j}(s^k)$ such that $a_{i_{j-1}} P_{i_j} a_{i_j}$ for all j, then if $f_{i_j}(R) \neq a_{i_{j-1}}$ for any j then PE is violated. Thus all trades must occur. It is possible that there is no cycle. However, if s^k is not an interim state, and $\hat{E}(s^k) \neq \emptyset$, then at least one cycle must occur in every trading round.

By Lemma 5 there is a pair of agents $M(s^k) \subset \mathcal{N}$ such that $[top_M(R) = (a, a)] \implies [f_M(R) = (a, a)]$. Let the agents in M be denoted $g_1(s^k), g_2(s^k)$. Thus if the first proposal (of length 2) is acceptable, then it must be their assignment. Otherwise, by the Closure Generation, the first acceptable proposal must be the assignment. Moreover, this is independent of the preferences of other agents. By Lemma 7, at least one of $g_1(s^k), g_2(s^k)$ must get her top-ranked project from the pair-option-set in this profile. Thus if there is no acceptable proposal, assign $g_1(s^k)$ her top-ranked project in $\bar{Z}(s) \cup \hat{Z}(s)$. Stop if the resulting state is a terminal state.

In each stage at least one agent receives an assignment. Thus the procedure is guaranteed to terminate in a finite number of steps. Moreover, assignments are made in according with the PPT rule procedure. This completes the proof.