House Allocation with Indifferences: A Generalization and a Unified View*

Daniela Saban† and Jay Sethuraman‡

July 2013

Abstract

We consider the problem of reallocating indivisible objects amongst a set of agents when the preference ordering of each agent may contain indifferences. The same model, but with strict preferences, goes back to the seminal work of Shapley and Scarf in 1974. When preferences are strict, we now know that the Top-Trading Cycles (TTC) mechanism invented by Gale is Pareto efficient, strategy-proof, and finds a core allocation, and that it is the only mechanism satisfying these properties. In the extensive literature on this problem since then, the TTC mechanism has been characterized in multiple ways, establishing its central role within the class of all allocation mechanisms. The question motivating our work is the extent to which these results can be generalized to the setting with indifferences. Our main contribution is a general framework to design strategyproof mechanisms that find a Pareto optimal allocation in the weak-core. Along the way, we establish sufficient conditions for a mechanism (within a broad class of mechanisms) to be strategyproof and use these conditions to design fast algorithms for finding a "good" reallocation. Our results generalize and unify two (different) mechanisms for the reallocation problem derived, independently of each other, by Manjunath & Jaramillo, and Alcalde-Unzu & Molis.

KEYWORDS. housing market; full-preference domain; mechanism design; TTC mechanism; allocation of indivisible objects

1 Introduction

Consider a market consisting of a set of agents, each of whom is initially endowed with an indivisible object (e.g., a house). Agents have preferences over all objects in the market and are allowed to trade their goods, but monetary transfers are not permitted. Hence, the market activity will result in a redistribution of the objects based on agents' preferences. This simple market, known as the *housing market*, naturally appears in many real-life applications such as housing allocation [1] and kidney exchange [8].

Shapley and Scarf [10] proposed the Top Trading Cycles (TTC) algorithm (attributed to David Gale) for the re-allocation of objects in markets of this type, but for agents with *strict* preferences. Agents perform trades according to the following rules, which are repeated until no agent is left:

*A preliminary version of this work, without proofs, has been presented in our own conference paper in the Fourteenth ACM Conference on Electronic Commerce (EC'13) [9].

†Graduate School of Business, Columbia University, New York, NY; dhs2131@columbia.edu

‡IEOR Department, Columbia University, New York, NY; jay@ieor.columbia.edu. Research supported by NSF grant CMMI-0916453.

1. Construct a graph with one vertex per agent. Each agent points to the owner of his top-ranked object among the remaining ones. As all vertices have out-degree 1, at least one cycle must exist and no two cycles overlap. Select the cycles in this graph.
2. Permanently assign to each agent in a cycle the object owned by the agent he points to. Remove all agents and objects involved in a cycle from the problem.

The allocation produced by the TTC algorithm is in the *strict-core*, which means that no coalition of agents can weakly improve upon it. The fact that the allocation is in the strict-core automatically implies that it has two very desirable properties: it is *Pareto efficient*, i.e., no agent can obtain a better object without making at least one agent worse off, and also *individually rational*, meaning that every agent weakly prefers his allocation to his initial endowment. The TTC mechanism is also *strategy-proof*, that is, no agent can do better by misreporting his preferences [7]. Furthermore, it is the only mechanism satisfying individual rationality, Pareto-efficiency and strategy-proofness on the strict preference domain [5].

An important limitation of the original Shapley-Scarf model is that agents are assumed to have strict preferences. In many situations, however, it is natural to consider the *full preference* domain, where agents are allowed to be indifferent between two or more objects. For instance, when applying to university housing, students are allowed to express their preferences only on some characteristics of the houses (e.g., number of rooms and roommates) and all houses sharing those characteristics are considered to be in the same indifference class. Another class of examples are online trading sites, which are very popular tools for certain types of goods, such as video games and anime. A widely known website among the gamer community is GameTZ, where users keep track of their wanted and available games in the site's database and GameTZ's matching system helps the users find the trades that they're interested in. In this application, users are allowed to rank their wanted games in a non-strict fashion.

A straightforward approach to account for indifferences is to resolve the indifferences using an arbitrary tie-breaking rule and then apply the TTC algorithm to the resulting strict-preference instance. It is unfortunate, therefore, that this idea may lead to an allocation that is *not* Pareto efficient. To illustrate, consider two agents, 1 and 2 endowed with objects a and b respectively. Suppose agent 1 is indifferent between objects a and b, whereas agent 2 strictly prefers a to b. To guarantee Pareto efficiency, the agents *must* trade their endowments, but an arbitrary tie-breaking rule may rank a before b for agent 1, resulting in the inefficient allocation where agents do not trade. This argues for a nuanced tie-breaking rule that takes into account the reported preferences of the agents; but in such a case, ensuring strategy-proofness is tricky, as agents may be able to manipulate the tie-breaking rule (and hence the outcome) by changing their preferences. In addition, some properties that hold with strict preferences may fail to hold when allowing indifferences. In particular, the strict-core may be empty when indifferences are present. Therefore, the core solution is usually relaxed to finding allocations in the *weak-core*, which is the set of allocations such that no coalition of agents can strictly improve upon it. Unfortunately, membership in the weak-core does not guarantee Pareto efficiency.

A compelling idea in studying the Shapley-Scarf model with indifferences, as opposed to a different model, is to see if one can extend the TTC mechanism in a natural way. From a practical point of view, there has been a renewed interest in TTC mechanism. The New Orleans Recovery School District has recently adopted the TTC mechanism to assign more than 28,000 students to school seats, taking into account both the preferences of students and the priorities of schools. In addition, the San Francisco School Board approved in 2010 the use of a mechanism based on the TTC for the assignment of school seats in their district.

From the theoretical point of view, the special case of strict preferences is exploited by the TTC algorithm in two (related) ways: in any iteration of the TTC algorithm, each agent has

a unique best object (because of strict preferences), and so points to exactly one agent (the owner of the said best object); furthermore, the cycles that are formed in this graph have no overlaps, as each agent points to exactly one other agent. When indifferences are permitted, the natural extension of the TTC algorithm is to let each agent point to *all* the owners of his most preferred objects (among the remaining ones) and perform the trades as indicated by the cycles of this graph. This is difficult because the cycles in the TTC graph may overlap, causing an agent to be involved in multiple cycles. In the earlier example, agent 1 is a member of two cycles: one involving only himself (a self-loop), and the other involving agents 1 and 2. Even if one uses some kind of a priority rule to resolve these cycles, ensuring Pareto efficiency and strategy-proofness is likely to be non-trivial. Additionally, in the original model, an agent can leave the problem as soon as he takes part in a trading cycle; in the model with indifferences, however, it may be necessary for an agent to "stay" in the problem to help execute trades that are Pareto-improving for others, but not for him. In the earlier example with two agents, if agent 1 leaves because he has one of his best objects, the resulting allocation will be inefficient. Thus, while we can think of a natural extension of the TTC framework to accommodate indifferences in preferences, ensuring that the resulting mechanism is Pareto-efficient, individually rational, and strategy-proof is non-trivial and necessitates a more careful understanding of the interactions between these requirements.

To the best of our knowledge, four recent papers deal with generalizations of the TTC algorithm for the full preference domain. Simultaneously and independently, Alcalde-Unzu and Molis [2] and Jaramillo and Manjunath [4] proposed different generalizations of the TTC algorithm, both of which are strategy-proof and produce efficient allocations that are also in the weak core. These mechanisms are discussed in more detail in Section 4. Aziz and de Keijzer [3] generalize the ideas of the previous mechanisms into a single family of mechanisms, called the GATTC class. The show that every member of that family is Pareto-efficient and individually rational, and provide an example to illustrate that strategy-proofness may fail to hold. Their main contribution is to show that the family of mechanisms introduced by Alcalde-Unzu and Molis has an exponential running time. Finally, in a recent paper that was written at the same time as ours, Plaxton [6] provides a different generalization of the TTC mechanism with the same properties and proposes a $O(n^3)$ implementation. These works establish the existence of several strategy-proof mechanisms that find efficient allocations in the weak core whenever indifferences in preferences are allowed. This is in sharp contrast with the strict preferences case, in which the TTC is the only mechanism satisfying the three properties.

This paper unifies and generalizes these earlier results by describing a family of mechanisms in which all members produce Pareto-efficient allocations in the weak-core. As in all the prior related papers, our focus is on generalizing the TTC mechanism in the following manner: We construct a graph in which each agent points to the owners of his most preferred objects and provide general departure conditions which ensure that both Pareto-efficiency and the weak-core property are preserved. Each individual mechanism in this family differs only in the rule used to select the trading cycles that are implemented; thus each selection rule induces a mechanism.

Our main goal is to get insights on which selection rules induce strategy-proof mechanisms. To do so, we start by limiting the type of rules we consider by imposing two additional requirements and show that a property, named local invariance, characterizes all strategy-proof rules within this class. One of our key contributions is to derive sufficient conditions on the selection rules to guarantee the local invariance (an thus the strategy-proofness) of the induced mechanisms. In addition, we present a general family of rules inducing strategy-proof mechanisms that compute allocations in polynomial time. The mechanisms in our family only solve improving cycles, meaning that in each cycle implemented by the mechanism there is at least one agent who strictly prefers his new allocation to the old. Finally, we show that a member of that family

runs in $O(n^2 \log n + n^2\gamma)$ (where γ is the maximum size of an equivalence class in any preference list), which is the fastest known strategy-proof mechanism for computing such an allocation.

The rest of the paper is organized as follows. In Section 2, we introduce our family of mechanisms and show that every mechanism in the class produces an efficient and weak-core allocation. In Section 3, we discuss further conditions that the selection rules for trading cycles must satisfy and show that, in this setting, proving strategy-proofness is equivalent to showing a much simpler statement. We end Section 3 by providing sufficient conditions on the rules for that statement to hold. In Section 4, we discuss the existing mechanisms and show why they are members of our family of mechanisms. In addition, we present a new class of selection rules and show that the mechanisms induced by these rules are strategy-proof. We conclude with a brief discussion of some open problems in Section 5.

1.1 Definitions and Notation

We consider a market consisting of a set of n agents $N = \{1, \ldots, n\}$. Each agent is initially endowed with an object and has preferences, possibly non-strict, over all the objects, including his endowment. We denote agent i's endowment by $\omega(i)$ and his preferences by P_i. We denote by $\omega = (\omega(1), \ldots, \omega(n))$ the vector of initial endowments and by $P = (P_1, \ldots, P_n)$ the preference profile for all agents. A problem is completely defined by the triple $\langle N, \omega, P \rangle$.

For any two objects a and b we write $a \geq_{P_i} b$ (resp. $a =_{P_i} b$) if agent i weakly prefers a to b (resp. is indifferent between a and b). We write $a >_{P_i} b$ to indicate that agent i strictly prefers object a to object b. An *allocation* is a re-distribution of the objects in which each agent obtains exactly one object. For the allocation μ, let $\mu(i)$ be the object allocated to agent i. An allocation μ is *Pareto-efficient* if, for every allocation μ' and every agent i, $\mu'(i) >_{P_i} \mu(i)$ implies $\mu(j) >_{P_j} \mu'(j)$ for some $j \in N$; that is, we cannot improve agent i's allocation without making someone worse-off. An allocation μ is in the *weak core* if it is not possible for any subset S of agents to reallocate their endowments in such a way that every one of them *strictly* prefers this new allocation to their allocation under μ. In particular, if S is a singleton agent, this condition reduces to *individual rationality*. A mechanism is said to be Pareto-efficient and in the weak-core if it always finds allocations that are Pareto efficient and in the weak-core respectively. A mechanism is *strategy-proof* if the allocation an agent obtains when reporting his true preference ordering is weakly preferred to the allocation he obtains by reporting any other preference ordering.

Given a (reduced) problem, an agent is said to be *satisfied* if he owns one of his most-preferred objects among the remaining ones and *unsatisfied* otherwise. The *TTC-graph* associated with a problem is a graph containing one vertex per agent and each agent points to the owner of his top-ranked objects among the remaining ones. Given the TTC-graph associated with a (reduced) problem, a trading cycle is said to be *improving* if it contains at least one unsatisfied agent, and *non-improving* otherwise. A trading cycle is *solved* or *implemented* when we assign to each agent in the cycle the object owned by the agent he points to.

A directed graph is *strongly connected* if there is a path from each vertex in the graph to every other vertex. The *strongly connected components* of a directed graph are its maximal strongly connected subgraphs. A strongly connected component S is a *sink* if all the neighbors of each vertex in S are also in S. It is easy to see that every directed graph has at least one sink. Let us note that all the previous definitions can be applied to the TTC-graph, since it is a directed graph. We say that a sink in the TTC-graph is a *terminal sink* if all vertices in the sink have an edge pointing to themselves, i.e., if all the agents in the sink are satisfied.

Given a graph $G = (V, E)$, a *subgraph* of G is a graph whose vertex set is a subset $V' \subseteq V$, and whose edges are a subset of E restricted to this subset V'. A graph $H = (V', E')$ is an

induced subgraph of G if it is a subgraph and for every pair of vertices $v, w \in V'$ such that $(v, w) \in E$, then $(v, w) \in E'$. Under this setting, we denote by $G[V']$ the subgraph of G induced by the vertex set V'.

2 The Trading Algorithm

In this section, we present a family of mechanisms that produces allocations in the weak-core satisfying Pareto-efficiency. Recall from the Introduction that the extension of the TTC mechanism to the full-preference domain present two main design challenges: the departure condition for agents and objects, and the selection rule to decide which trading cycles to execute when overlapping cycles exist.

The family of mechanisms that we propose is explained in Algorithm 1. We note that this is essentially the one presented in Aziz and de Keijzer [3], and follows the same basic setup as in all prior works on this problem. Every mechanism in this family consists of the same two phases: the removal and update phase and the improvement phase. During the removal and update phase, we address the departure problem by iteratively removing those agents who are satisfied and cannot be part of an improving cycle (together with the objects they own) and updating the problem accordingly. Each time one of the outer loops is executed, i.e., both the removal and update and improvement phases are executed, we say that a *step* of the algorithm takes place. We refer to the executions of the loop in the removal and update phase as *iterations within a step*.

During the improvement phase, we solve some top trading cycles. The selection among overlapping cycles will be done by a rule that we refer to as the *selection rule*. The choice of rule will uniquely determine a mechanism in the family, therefore we say that each rule *induces* a mechanism.

Hereafter, we will refer to the selection rule as the F-rule. In the rest of this paper, we will just focus on rules satisfying the following additional properties:

1. For each agent i the selection rule will return a unique vertex j to whom i will point. Vertex j must own one of i's most preferred objects among the remaining ones.

2. The mechanism obtained when using the selection rule in Algorithm 1 terminates.

The second condition needs to be explicitly enforced, since not every rule satisfying the first condition induces a mechanism that terminates. To illustrate, consider the market with three agents ($N = \{1, 2, 3\}$) endowed with objects $\omega = (a, b, c)$. Agent 1 is indifferent between a, b and c, agent 2 is indifferent between a and b and both are strictly preferred over c and agent 3 has object a as his unique top-choice. Suppose that a common priority ordering over agents is given, where 1 has the highest priority followed by 2 and finally by 3. Let the selection rule be: each agent points to the highest priority agent different from himself that owns one of his top ranked objects. Then, agents 1 and 2 will keep trading among themselves and the mechanism does not terminate.[1]

[1]The reason termination condition needs to be enforced is that we are not requiring the selection rule to select *only* improving cycles. Indeed, in both the works by Alcalde-Unzu and Molis and Jaramillo and Manjunath, non-improving cycles can be selected. In Section 4 we limit our attention to rules which only select improving cycles.

Input: Agents' non-strict preference lists, ownership list
Output: A Pareto efficient and individually rational allocation. Strategy-proofness depends on the choice of rule F.

Repeat until no agent is left:

1. (Removal and Update phase)

   (a) Construct a TTC-graph G: there is a vertex per agent and each agent points to the owners of his top choices.

   (b) Repeat until G has no terminal sinks:

   Analyze the strongly connected components of G. There must be at least one sink component S. For every sink component S:

   If S is a terminal sink (i.e., every agent in S owns one of his top choices), permanently assign to each agent in S his own object and remove all the agents and objects in S from the problem, as well as from the preference lists of the remaining agents.

   Update the graph G so that each agent points to the owners of his (new) top-choices.

2. (Improvement phase) Apply the selection rule on G to obtain a set of disjoint trading cycles. Solve the cycles.

Algorithm 1: Trading Algorithms

2.1 Pareto efficiency, weak-core and generality of the Trading mechanisms

Our first result is that every member of the family of mechanisms described in Algorithm 1 produces a Pareto efficient allocation.

Theorem 1. *Whenever the selection rule leads to termination, the allocation given by Algorithm 1 is Pareto efficient.*

Proof. Consider the allocation given by the mechanism. Let T_1 be the first step in which a terminal sink is removed. Every agent leaving in step T_1 will end up owning one of his top-ranked objects. Let $T_2 > T_1$ be the first step after T_1 in which a terminal sink is removed. Any agent leaving in step T_2 will end up with one of his top-ranked objects among those remaining. If an agent leaving in step T_2 strictly prefers another object to his current allocation, that object must have left in step T_1. Since all agents who left in step T_1 were in a terminal sink, none of them will be willing to trade his allocated object with anyone who was not in their sink. Hence, no agent leaving in step T_2 can do strictly better without forcing another agent to do worse.

In general, any agent leaving in step T_k will obtain one of his top choices among the remaining objects at that step. To strictly improve his allocation, he will need to obtain an object allocated in a previous step. By the definition of step 1b. in Algorithm 1, this cannot be achieved without someone leaving in a previous step being worse-off. □

Next, we show that the family of mechanisms described in Algorithm 1 always finds a weak-core allocation.

Theorem 2. *Whenever the selection rule leads to termination, the allocation given by Algorithm 1 is in the weak-core. In particular, the allocation is individually rational.*

Proof. Let μ be the allocation given by Algorithm 1 and suppose μ is not in the weak-core. Then, some subset of agents S can redistribute their original endowments among themselves in such a way that every agent in S prefers his new allocation to that in μ. Without loss of generality, let $S = \{1, \ldots, k\}$ and assume that in the dominating reallocation, agent 1 gets object $\omega(2)$, agent 2 gets $\omega(3)$, ..., and agent k gets $\omega(1)$.

Since agent 1 strictly prefers $\omega(2)$ over $\mu(1)$, it follows that object $\omega(2)$ was not in the problem the first time agent 1 pointed to $\mu(1)$ in the TTC-graph. Hence, it was not in the problem when agent 1 left. This implies that agent 2 either traded $\omega(2)$ or left with that object before agent 1 is able to leave. Following the same argument, object $\omega(3)$ must leave before agent 2 trades or leaves and so on. Finally, object $\omega(1)$ must leave before agent k trades or leaves, implying that agent 1 can only trade or leave the problem only after $\omega(1)$ has left the problem, which is a contradiction. □

It would be interesting to show that the framework proposed in Algorithm 1 is without loss of generality, meaning that any Pareto efficient and weak-core allocation can be obtained by the above algorithm (with some minor modifications). In Figure 1, we provide an example which illustrates how the trading algorithm may fail to find some efficient and weak-core allocations.

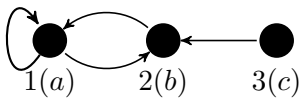
Original endowments:

$\omega(1) = a$, $\omega(2) = b$, $\omega(3) = c$.

Preference Lists:

1	{a,b}, c
2	a, c, b
3	b, ...

(a) Endowments and preferences. Objects within braces are in the same indifference class. Indifference classes are separated by a comma. One can verify that, in this setting, allocation $\mu = (a, c, b)$ is efficient and in the weak-core.



(b) The allocation μ cannot be implemented as the TTC-graph is missing an edge from 2 to 3. The edge will only appear once agent 1 departs with a, but in that case we need to find a different departure condition.

Figure 1: Example illustrating how the trading algorithms may fail to find some efficient and weak-core allocations.

2.2 Computational complexity

We now analyze the computational complexity of our family of mechanisms. We may assume that individual preferences are stored as a list of sets representing the indifference classes, so the graph is readily available. We have an ownership vector indexed by objects, which keeps track of who is the current owner of each object. We also have a vector indexed by agents to record the final allocation. We assume that both vectors can be accessed in $O(1)$ time.

Let the TTC-graph be $G = (V, E)$. The strongly connected components of a graph can be found in $O(|V| + |E|)$ using Tarjan's algorithm [11]. Identifying all the strongly connected components that are sinks can be done in $O(|V| + |E|)$, as it is a basic reachability problem. Detecting whether the sinks are terminal sinks can be done in $O(|V| + |E|)$, as we just need

to check if all the agents in a sink are satisfied. Before removing an object from the problem, we update the solution and the ownership record. Therefore, each iteration in the removal and update phase takes $O(|V| + |E|)$ time, without counting the removal of objects.

Removing each object from the preference lists can be done in $O(|E|)$, since there is no need to iterate through all the preference lists. We can delete the objects leaving the problem from the individual preference lists when they are among the top choices of some agent. Every time an agent reveals a new indifference class, it can be updated by deleting the objects that are no longer in the problem. Then, we will have a total of $O(|V|^2)$ updates, plus the $O(|E|)$ in every iteration. Note that if the preferences are strict, all of the above reduces to $O(|V|)$ as in the original TTC mechanism.

We now bound the number of iterations of the removal and update phase that take place during a complete execution of the mechanism. In each step, at least one iteration of the removal and update phase takes place. Additional iterations will only occur if a terminal sink is found. Note that at most n terminal sinks can be found before the algorithm terminates. Therefore, a total of at most (n + # improvement phase steps) executions of the removal and update phase take place throughout the algorithm, each one consisting of $O(|V| + |E|)$ operations. Hence, the complexity of the algorithm is basically determined by the number of improvement phase steps, and by how fast we can calculate the selection.

3 Strategy-proofness

As shown in Section 2, every mechanism in our family finds a Pareto efficient, weak-core allocation, regardless of the selection rule used. However, as the following example shows, not every mechanism is strategy-proof.

Rule 1. *A common priority order over agents is given. Each agent points to the highest priority agent (excluding himself) who owns one of his top choices.*[2]

The instance shown in Figure 2 illustrates why the mechanism induced by the above rule is not strategy-proof.

3.1 Additional conditions on the selection rules

Ideally, one would like to characterize *all* rules inducing strategy-proof mechanisms. Unfortunately, showing that the resulting mechanism is strategy-proof appears to be difficult if no additional constraints over the rules are imposed. Throughout the rest of this section, we will focus only on rules that satisfy two additional properties, namely "Independence of unsatisfied agents" and "Persistence".

Let $F(G)$ be the subgraph obtained from a TTC-graph G when the selection rule F, or simply the F-rule, is applied. Since each vertex will have out-degree exactly one, each connected component in $F(G)$ will consist of a directed cycle and directed paths ending in a vertex in that cycle. Let $P = (a_1, \ldots, a_k)$ be a path in $F(G)$. We say that path P is *F-persistent* or just persistent if P appears in all the successive steps of the algorithm until agent a_k trades his object or leaves the system. We require that the F-rule satisfy the following additional properties:

> (Independence of unsatisfied agents) Let i be an unsatisfied agent and let G_1 and G_2 be two TTC-graphs which only differ in the outgoing edges from i. Then, $F(G_1)$ and $F(G_2)$ may only differ in the outgoing edge from i.

[2]This rule may actually not terminate for some instances. It is possible to overcome this by adding some additional conditions to avoid cycling. However, details are omitted for the sake of clarity.

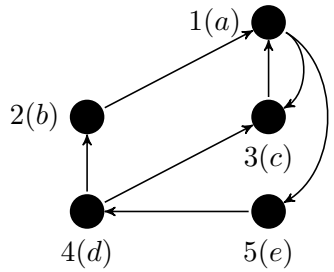
Original endowments:
$\omega(1) = a$, $\omega(2) = b$,
$\omega(3) = c$, $\omega(4) = d$
$\omega(5) = e$.

Agent order:
$1, 2, 3, 4, 5$.

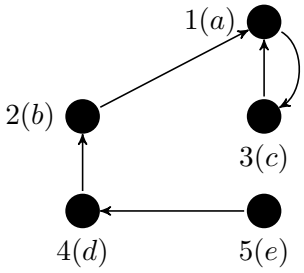
(a) Endowments and orders. Agent 1 has the highest priority followed by agents $2, 3, 4$ and 5.

Preference Lists:	
1	{c, e}, ...
2	a, d , ...
3	a, ...
4	{b, c}, ...
5	d, ...

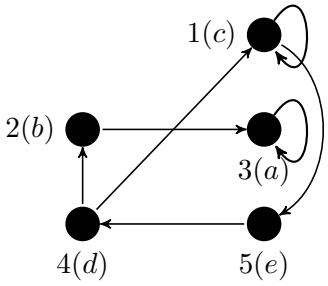
(b) Preferences.



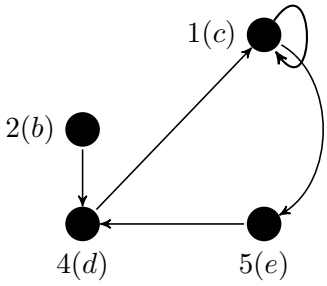
(c) G_0. There are no terminal sinks in this graph.



(d) $F(G_0)$. This graph has a unique cycle $1 \to 3 \to 1$, so agents 1 and 3 trade their objects.



(e) G_1. In the second step $\{3\}$ is a terminal sink, so agent 3 and object a are removed from the problem. After that, agent 2 points to agent 4, the owner of object a.



(f) $F(G_1)$. The trade between agents $1, 4, 5$ takes place and agent 5 leaves the problem with object d. Hence, agent 2 cannot obtain any of his first two choices.

Figure 2: This example illustrates why Rule 1 fails to induce a strategy-proof mechanism. Agent 2 cannot obtain any of his first two choices by reporting truthfully. The reader can verify that, if agent 2 lies by reporting object d as his top-choice, he will be able to get object d (his second choice in the original problem) in the first step.

(Persistence) Every path $P = (a_1, \ldots, a_k)$ where agent a_k is unsatisfied is F-persistent.

The importance of the "Independence of unsatisfied agents" property will became clear in Section 3.2, when we provide conditions under which a selection rule induces a strategy-proof mechanism. Proving that an arbitrary F-rule induces a strategy-proof mechanism appears to be difficult in the absence of this property.

A similar argument can be given for the persistence property. While persistence might not be necessary for a rule to induce a strategy-proof mechanism, we justify this requirement by explaining the intuition behind it.[3] Let i be a vertex corresponding to an unsatisfied agent. For a given TTC-graph G, consider the set of agents that are in a path to i in $F(G)$ and let o be agent i's most-preferred object among those owned by agents in that set. If the "Independence of unsatisfied agents" condition holds, the rule needs to guarantee that i obtains an object at least as good as o. Otherwise, i will be better off misreporting his preferences by declaring object o as his unique top-choice at that point and thus forming a trading cycle under the F-rule.

[3]The importance of persistence property is also noted by Jaramillo and Manjunath. Indeed, they explicitly enforce persistence in their rule as well.

We conclude this section by highlighting that the persistence and "Independence of unsatisfied agents" properties are not sufficient to guarantee strategy-proofness in the induced mechanism. An example of a rule which satisfies all conditions and yet does not induce a strategy-proof mechanism is included in the appendix.

3.2 Characterizing strategy-proofness in our setting

Throughout this section we fix agent i and assume that i is the only agent misreporting his preference ordering. Let $\Delta = \langle N, \omega, P \rangle$ be the problem with the true preferences $P = (P_{-i}, P_i)$ and $\Delta' = \langle N, \omega, P' \rangle$ be the problem with preferences $P' = (P_{-i}, P_i')$, which only differs from Δ in the preferences reported by agent i. Given a (reduced) problem Δ, let G^Δ be the TTC-graph of Δ, i.e., a graph such that every agent in Δ points to his top-ranked objects among the remaining ones. Denote by G_m^Δ the graph obtained at the end of step m (or, equivalently, at the beginning of step $m+1$) when we start with the (reduced) problem Δ. Given a graph G^Δ, let $F(G^\Delta)$ be the subgraph of G obtained when the F-rule is applied. Finally, we denote by μ the final allocation obtained by the algorithm when preferences P and endowments ω are used.

We start with some remarks and claims that will lead us to the final result.

Remark 1. *Let T be the first iteration in which agent i trades his original endowment (if any), and let α be the object he obtains after the trade. Then, $\mu(i) =_{P_i} \alpha$.*

To see this, note that when agent i is assigned a new object, it must be one of i's top ranked objects among those remaining in step T. If agent i never trades the object again, we are done. If agent i trades it in step $T' > T$, it must be for one of his top ranked objects among those remaining in step T'. Since object α is still in the problem and the objects remaining at step T' are a subset of those remaining in step T, it follows that i must be indifferent between his top-ranked objects in iteration T' and α. Then, $\alpha =_{P_i} \beta$, where β is the new object assigned to i. Remark 1 follows by induction.

Claim 1. *Let t (resp. t') be the first step in which agent i is involved in a trading cycle or is satisfied in Δ (resp. Δ'). Then, for every step m', $0 \leq m' < m = \min(t, t')$, the same terminal sinks and trading cycles occur in both problems.*

Furthermore, by end of any step $m' < m$, both graphs ($G_{m'}^\Delta$ and $G_{m'}^{\Delta'}$) will only differ in the outgoing edges from i, every agent will be endowed with the same objects, and the same sets of agents and objects remain in both problems.

Proof. If $m = 0$, the claim vacuously holds. Suppose it holds for all $k' < m' < m$, we want to show it holds for $m' < m$. As $m' \leq m$, none of the terminal sinks found in the first iteration of the removal an update phase in Δ (if any) will contain i. Given that all the vertices other than i are pointing to the same vertices in both problems, it follows that every terminal sink in step m', iteration 1 in Δ will also be a terminal sink in Δ'. Since all agents are endowed with the same objects in both problems (by inductive hypothesis), the TTC-graphs obtained after the removal of those sinks will only differ in the outgoing edges from i. Given that i will continue to be unsatisfied throughout the step, the same argument applies for every iteration of the removal and update phase in step m' of the algorithm. Hence, once the removal and update phase ends, we obtain the same reduced problem in both cases, except possibly for the outgoing edges from i.

Next, we analyze what happens with the trading cycles. Note that agent i will be unsatisfied in both problems until step m. Then, by the "Independence of unsatisfied agents" condition, $F(G_{m'}^\Delta)$ and $F(G_{m'}^{\Delta'})$ will only differ in the outgoing edge from i. Since i is not in a trading cycle in this step in either problem and all other agents are pointing to the same object, the same

trading cycles will occur. We highlight that without the "Independence of unsatisfied agents" condition we cannot guarantee that the same trading cycles will be formed. As we started with the same reduced problem except for the outgoing edges from i, it follows that by the end of the step every agent will be endowed with the same objects and the same sets of agents and objects remain in both problems. □

Fix a mechanism M in the family of mechanisms described in Algorithm 1. Then M satisfies *local invariance* if and only if the following property holds for all P and P':

Property 1. *(***Local Invariance***) Let $P = (P_{-i}, P_i)$ be the preference lists of the agents, where $P_i = (p_1, \ldots, p_r)$ and p_j represents the set of objects corresponding to agent i's j^{th} indifference class. Suppose that agent i obtains object $\alpha \in p_k$ (α is in agent i's k^{th} indifference class) when mechanism M is applied to the preference profile P, and suppose and $\alpha >_{P_i} \omega(i)$. Let $P' = (P_{-i}, P_i')$, where $P_i' = (p_1, \ldots, p_{k-1}, \alpha, p_k \backslash \{\alpha\}, p_{k+1}, \ldots, p_r)$. Then, when mechanism M is applied to P', agent i still obtains α.*

This property of "local invariance" turns out to be a key intermediate step in the proofs of strategy-proofness in both [2] and [4], even though their rules are quite different. That this is not a coincidence is shown in the following theorem, which asserts that this local invariance property is equivalent to strategy-proofness in our setting.

Theorem 3. *A mechanism satisfying the "Independence of Unsatisfied Agents" and the "Persistence" properties is strategy proof if and only if it satisfies local invariance.*

Proof. First, suppose that the mechanism is strategy-proof but does not satisfy local invariance. Let Δ be a counterexample, and let Δ' be as in the statement defining local invariance, i.e., it differs from Δ only in agent i's preferences P_i' as indicated in the statement of the property. As the mechanism is strategy-proof, i should obtain in Δ' an object that he prefers weakly less than α. By assuming that Property 1 does not hold, we conclude that i will get something worse than α when reporting P_i', but he will get α when reporting P_i. Thus, if P_i' are indeed agent i's true preferences he will be better off by reporting P_i, contradicting strategy-proofness.

To show the converse, suppose that Property 1 holds and the algorithm is not strategy-proof. Then, there exists an agent i who can report preferences P_i' and obtain an object $\alpha >_{P_i} \mu(i)$. By Property 1, we may assume that α is the only object in its indifference class for P_i'. Let $m = \min(t, t')$, where t (resp. t') is the first step in Δ (resp. Δ') in which i either trades or is satisfied. By Claim 1, up to the beginning of step m both reduced problems are the same except for the outgoing edges from i. Now we consider the following cases:

$t \leq t'$: Since agent i has not traded in Δ' before t, it follows that α needs to be in the problem at the beginning of step t. Then, i has to be pointing to α in Δ at step t, and since $\alpha >_{P_i} \omega(i)$ and $\alpha >_{P_i} \mu(i)$, agent i cannot trade or be satisfied in problem Δ.

$t > t'$: Given that agent i will not be satisfied before trading in Δ', let $C = (q_0, \ldots, q_k, i)$ be the F-trading cycle in which i takes part in step t' in Δ'. By the persistence and the "Independence of unsatisfied agents" conditions, it follows that the path $q_0 \to \ldots \to q_k \to i$ also appears in $F(G_{t'}^{\Delta})$ and will appear in every step until i trades or is satisfied. This implies that q_0 will not trade his current endowment α as long as i does not trade his endowment or leave the system. Hence, i will be satisfied before α leaves the system and, by Remark 1, $\mu(i) \geq_{P_i} \alpha$, which is a contradiction.

□

The above result has a very strong implication. Whenever one wishes to verify whether a mechanism is strategy-proof, it is necessary to check that an agent cannot improve his allocation under all possible manipulations. By establishing the equivalence between local invariance and strategy-proofness in our setting, Theorem 3 implies that we only need to check for the type of manipulations described in Property 1.

3.3 Sufficient conditions for local invariance

Theorem 3 motivates us to find simple conditions on the rules that are sufficient for local invariance (Property 1) to hold in our setting. Such conditions are presented in Theorem 4.[4]

As usual, let Δ be the original problem with the original preferences $P = (P_{-i}, P_i)$ and Δ' be the problem with preferences $P' = (P_{-i}, P_i')$. Let t (resp. t') is the first step in Δ (resp. Δ') in which i either trades or is satisfied. At any step m, let $C_m^{\Delta}(i)$ be the set of vertices v such that there is a directed path from v to i in $F(G_m^{\Delta})$. Let A_m^{Δ} (resp. O_m^{Δ}) be the set of remaining agents (resp. objects) at step m in problem Δ. Given an agent j and an object α, we denote by $o_m(j)$ the object owned by agent j at step m and by $a_m(\alpha)$ the agent who owns α at step m.

Theorem 4 (Sufficient Conditions for Local Invariance)**.** *Any one of the following conditions is sufficient for local invariance (Property 1) to hold:*

(a) At each step $m < t'$, each agent $j \in A_m^{\Delta'}$ that is not in $C_m^{\Delta'}(i)$ is also in A_m^{Δ} and is endowed with the same object in both problems. In addition, the same trading cycles involving agents not in $C_m^{\Delta'}(i)$ are solved in both problems.

(b) At each step $m < t'$, each agent $j \in A_m^{\Delta'}$ that is not in $C_m^{\Delta'}(i)$ is also in A_m^{Δ}. In addition, if j points to agent k in $F(G_m^{\Delta})$ and $\beta = o_m^{\Delta}(k)$, then j points to $a_m^{\Delta'}(\beta)$ in $F(G_m^{\Delta'})$.

(c) At each step $m < t'$, each agent $j \in A_m^{\Delta'}$ that is not in $C_m^{\Delta'}(i)$ is also in A_m^{Δ}. In addition, each j points to the same agent in $F(G_m^{\Delta})$ and $F(G_m^{\Delta'})$.

(d) For each agent i, let R be the set of i's most-preferred objects among the remaining ones. Let o be the object selected by the F-rule for agent i, that is, i points to the owner of o in $F(G)$. The F-rule has the following property: for any arbitrary instance such that $S \subseteq R$ represents agent i's top-choices and $o \in S$, the F-rule will also select object o for agent i.

Proof. Using the same definition of preferences as in the statement of the local invariance property, we know that eventually i will point to the objects in p_k and this can only happen when all objects in $W = \cup_{j=1}^{k-1} p_j$ are no longer in the problem. Let T_1 be the first step in which none of the objects in W are in the problem under P. For all $w \in W$, we have that $w >_{P_i} \alpha >_{P_i} \omega(i)$, so agent i cannot be satisfied before T_1. Since all the reported preferences but P_i are the same and i cannot be satisfied until T_1, by Claim 1 it follows that the state of both problems will be the same up to step T_1. Therefore, we may assume we start with the reduced problem in T_1.

By Claim 1, at the beginning of step $T = min(t, t')$, the state of the problem is the same in both instances except for the outgoing edges from i. Then, object α must still be in the problem and if $T = t'$ we know agent i gets α. Hence, we may assume $T = t < t'$. Again, by Claim 1, we may assume that we start with the reduced problem obtained in the beginning of step T. From now on, we will call these reduced problems Δ and Δ' respectively. At this point, both problems have the same set of agents endowed with the same objects and all agents but i have the same preference lists (this follows once more by Claim 1).

[4] We emphasize that the requirements imposed on selection rules in Section 3.1, namely "Persistence" and "Independence of unsatisfied agents", are not sufficient to guarantee that Property 1 holds. The reader is referred once again to the example provided in the appendix.

Proof of (a) Suppose that, under P', i does not get α. Then, α must be removed from problem Δ' before i trades his object (or is satisfied), meaning α must leave before t'. The idea of the proof is to show that α cannot be in a terminal sink without i in problem Δ'. This implies that i must be satisfied while α is still in the problem (since he is in a terminal sink) and, therefore, i must obtain α in Δ'.

It is sufficient to show that, at each step $m < t'$, every terminal sink in step m in Δ' is also a terminal sink in step m in Δ, every trading cycle that occurs in Δ' at step m also occurs in Δ at step m and all agents that are not in $C_m^{\Delta'}(i)$ will end the step with the same endowments. Since every terminal sink that occurs before t' in Δ' is also a terminal sink in Δ, none of them can contain α, as the unique terminal that contains α in Δ also contains i and i is not in a terminal sink before time t' in Δ'. The proof will be done by induction in the step number. Recall that step T is the base case.

Start with problems Δ and Δ', which only differ in i's reported preferences at step T. As there is an edge from vertex i to the owner of $\alpha >_{P_i} \omega(i)$ in the TTC-graph in both problems, i is an unsatisfied vertex. Hence, i cannot be in a terminal sink in any of both problems and thus the set of terminal sinks that we process at step T will be the same in Δ and Δ'. Furthermore, the improvement phase will start with the same agents endowed with the same objects and all agents having the same outgoing edges except for i.

We now consider the improvement phase of the current step T. As all agents but i point to the same agents and all are endowed with the same objects, by the "Independence of unsatisfied agents" property, $F(G_T^{\Delta})$ and $F(G_T^{\Delta'})$ only differ in the outgoing edge from i. Therefore, every trading cycle in Δ' will also be a trading cycle in the improvement phase in Δ. This establishes the basis for induction.

Assume that the inductive hypothesis holds for each step $m' < m$, that is, every terminal sink in step m' in Δ' is also a terminal sink in step m' in Δ, every trading cycle that occurs in Δ' at step m' will also occur in Δ in step m' and all agents that are not in $C_{m'}^{\Delta'}(i)$ end step m' endowed with the same objects in both problems. We want to show that this holds for $m < t'$. First, notice that every agent in a terminal sink will be a satisfied agent that is not in $C_m^{\Delta'}(i)$, as none of those agents can trade or leave before i. Since once an agent is part of $C_{m'}^{\Delta'}(i)$ it will continue to be so at least until t' by the persistence property, it follows that $C_T^{\Delta'}(i) \subseteq C_{T+1}^{\Delta'}(i) \subseteq \ldots \subseteq C_m^{\Delta'}(i)$. Hence, all the agents involved in a terminal sink were never part of $C_{m'}^{\Delta'}(i)$ for $m' < m$. Furthermore, since the terminal sinks at step m were not sinks at step $m-1$, it must be that at least one trade from the previous iteration is necessary for that terminal sink to be formed. By hypothesis, the same trading cycles involving agents that are not part of $C_m^{\Delta'}(i)$ took place in both problems. Then, they must also be in a sink in the original problem at step m and this holds for every iteration of the removal and update phase in step m. During the improving phase, all cycles in Δ' will be formed Δ by hypothesis. Thus, the property holds. $\diamondsuit$

Proof of (b) First, note that once an agent is part of $C_m^{\Delta'}(i)$ (for $m < t'$) it will continue to be so at least until time t' by the persistence property. Then, it suffices to show that, for all $m < t'$, all agents outside $C_m^{\Delta'}(i)$ are endowed with the same objects in both problems as local invariance will then follow by the proof of (a).

For $m = T$ it is trivial by using the same arguments as in the proof of (a). Suppose this holds for all $m' < m$, and we want to show it holds for m. By inductive hypothesis, at $m' = m-1$ the agents that are not in $C_{m'}^{\Delta'}(i)$ always point to the same objects in both problems and those objects are owned by the same owners, so all the same trades involving them took place at m' in both problems. By the end of iteration m', all agents not in $C_{m'}^{\Delta'}(i)$ will be endowed

with the same objects. Since the removal and update phase does not shift the endowments and $C_{m-1}^{\Delta'}(i) \subseteq C_m^{\Delta'}(i)$, the claim follows. $\diamondsuit$

Proof of (c) Again, note that once an agent is part of $C_m^{\Delta'}(i)$ (for $m < t'$) it will continue to be so at least until time t' by the persistence property. As in the proof of (b), it suffices to show that, for all $m < t'$, all agents outside $C_m^{\Delta'}(i)$ are endowed with the same objects in both problems as local invariance will then follow by the proof of (a).

The case of $m = T$ is trivial by using the same arguments as in the proof of (a). Suppose this holds for all $m' < m$, and we want to show it holds for m. By inductive hypothesis, at $m' = m - 1$ the agents that are not in $C_{m'}^{\Delta'}(i)$ always point to the same agent in both problems and those agents are endowed with the same objects, so all the same trades involving them took place at m' in both problems. Hence, by the end of iteration m', all agents not in $C_{m'}^{\Delta'}(i)$ will be endowed with the same objects. Since the removal and update phase does not shift the endowments and $C_{m-1}^{\Delta'}(i) \subseteq C_m^{\Delta'}(i)$, the claim follows. $\diamondsuit$

Proof of (d) Using induction on the step number m, we show that $O_m^{\Delta} \subseteq O_m^{\Delta'}$, every object in $O_m^{\Delta'} \backslash O_m^{\Delta}$ is owned by an agent in $C_m^{\Delta'}(i)$ and all agents that are not in $C_m^{\Delta'}(i)$ are endowed with the same objects in both problem at the end of step m. These statements imply, by the same arguments as in the proof of (a), that all sinks and cycles in Δ' will also appear in Δ and thus strategy-proofness will follow.

For $m = T$, we can use the same arguments as in the proof of (a) to show that the same terminal sinks are removed in the "removal and update" phase. Then, by the end of step m, $O_m^{\Delta} \subseteq O_m^{\Delta'}$ holds. Furthermore, the improvement phase starts with the same agents endowed with the same objects and all agents having the same outgoing edges in the TTC-graph except for i. Then, by the "Independence of unsatisfied agents" property, the same trading cycles occur for agents outside $C_m^{\Delta'}(i)$ and the basis for induction is established.

Assume that the inductive hypothesis holds for each step $m' < m$, that is, $O_m^{\Delta} \subseteq O_m^{\Delta'}$, every object in $O_m^{\Delta'} \backslash O_m^{\Delta}$ is owned by an agent in $C_m^{\Delta'}(i)$ and all agents that are not in $C_m^{\Delta'}(i)$ are endowed with the same objects in both problems at the end of step m. We shall show that this holds for $m < t'$. First, every agent in a terminal sink must be a satisfied agent that is not in $C_m^{\Delta'}(i)$, as none of those agents can be in a terminal sink without i. As $C_T^{\Delta'}(i) \subseteq C_{T+1}^{\Delta'}(i) \subseteq \ldots \subseteq C_m^{\Delta'}(i)$ by the persistence property, all the agents involved in a terminal sink in step m were not part of $C_{m-1}^{\Delta'}(i)$. Furthermore, since the terminal sinks at step m were not sinks at step $m-1$, it must be at least one trade from the previous iteration is necessary for that terminal sink to be formed. By inductive hypothesis, the agents that are not part of $C_{m-1}^{\Delta'}(i)$ were endowed with the same objects in both problems in step $m-1$ and $O_{m-1}^{\Delta} \subseteq O_{m-1}^{\Delta'}$ so the hypothesis of (c) the same trading cycles involving only agents that were not $C_{m-1}^{\Delta'}(i)$ took place at step $m-1$ in both problems. Hence, every sink in Δ' must also be in a sink in Δ problem at step m and this holds for every iteration of the removal and update phase in step m. Furthermore, sinks that are in Δ and not in Δ' can only include agents (and objects) in $C_{m-1}^{\Delta'}(i)$. Therefore, $O_m^{\Delta} \subseteq O_m^{\Delta'}$ and every object in $O_m^{\Delta'} \backslash O_m^{\Delta}$ is owned by an agent in $C_m^{\Delta'}(i)$.

During the improvement phase, every cycle formed in $F(G_m^{\Delta'})$ involving only agents that are not in $C_m^{\Delta}(i)$ will also be formed in $F(G_m^{\Delta})$. To see why, note that by the inductive hypothesis the agents that are not part of $C_{m-1}^{\Delta'}(i)$ are endowed with the same objects in both problems in step $m-1$. Since $C_{m-1}^{\Delta'}(i) \subseteq C_m^{\Delta'}(i)$, all agents that are not in $C_m^{\Delta'}(i)$ are endowed with the same objects in the beginning of the improvement phase. Moreover, $O_m^{\Delta} \subseteq O_m^{\Delta'}$ which by hypothesis implies that every agent not in $C_m^{\Delta'}(i)$ pointing to another agent not in $C_m^{\Delta'}(i)$ in $F(G_m^{\Delta'})$ must

point to the same agent in $F(G_m^{\Delta})$. Therefore, the same trades will involving agents not in $C_m^{\Delta'}(i)$, which completes the proof $\diamondsuit$ □

4 Selection Rules: Old and New

We start this section with an overview of the selection rules used by the two prior papers that designed strategyproof mechanisms. Alcalde-Unzu and Molis [2] propose the The Top Trading Absorbing Sets (TTAS) mechanism. The authors define an absorbing set to be what we call a sink. A unique strict priority ordering over all the objects is given. At every step, the authors consider only the sinks of the TTC graph. Those that are terminal sinks are removed from the problem. For those sinks that aren't terminal sinks, the trading rule is defined as follows:

For each agent pointing to more than one object, select a unique object to point to using the following criterion: she points to the maximal object with the highest priority from among those that have not been assigned to her yet. If all maximal objects have been assigned to her at least m times, then she points to the maximal object with the highest priority that has not been assigned to her $m+1$ times yet.

We highlight that trading only takes place within a sink. However, the rule behaves equivalently if applied to the whole TTC-graph at once. The intuition behind this is as follows: suppose a cycle $C = (q_0, \ldots, q_k)$ is found that is not in a sink. Then, all the vertices that form that cycle must be in a sink together for the first time and cannot trade their objects before that. At that time, q_{i+1} will still own the highest priority object among q_i's top-ranked objects. Then, cycle C will be solved. This equivalent view is explained formally when the authors show that their mechanism is strategy-poof.

It is easy to see that if the rule is applied to all agents rather than only to those in a sink, it satisfies the additional properties required in Section 3.1. First, each agent points to a unique top-ranked object among the remaining ones. The termination condition is shown in their paper and also applies to this case. Persistence holds by definition. Deciding to whom an agent points to is done independently of whom all other agents are pointing to (in particular, this includes unsatisfied agents). We highlight that it can be easily shown that, when applied to all the graph, this rule satisfies property (b) in Theorem 4 thus establishing strategy-proofness. The TTAS mechanism, however, was later shown to run in exponential time by Aziz and de Keijzer [3].

A second generalization of the TTC mechanism was given by Jaramillo and Manjunath [4]. The mechanism proposed in their work, called the Top Cycles Rule with Priority (TCRP), uses the same departure condition used in Algorithm 1. Indeed, their mechanism is a member of our family, as we will establish later. They consider a unique strict priority ordering over all agents and use it to create their *pointing* rule. In their rule, agent will point to a single other agent decided as follows:

Each agent i that pointed to an agent j who did not trade in the previous step continues to point to the same agent j in the current step. Each agent i with a unique most-preferred object (among the remaining ones) points to the agent holding that object.

The pointing of remaining agents will be decided so as they are in a shortest path to an unsatisfied agent whenever possible, with ties broken in favor of the highest-priority unsatisfied agent. Specifically:

Each agent who is at unit-distance from an unsatisfied agent in the TTC-graph (i.e., at least one of his most-preferred objects among the remaining ones is held by an unsatisfied agent) and is not pointing to anyone yet, points to the highest priority agent among such unsatisfied agents.

After all such agents are processed, the algorithm considers agents who are at a distance of 2 from an unsatisfied agent: if i is one such agent and U_i is the set of unsatisfied agents that are at a distance of 2 from i, then i points to an agent who points to the highest-priority agent in U_i, with ties broken in favor of higher-priority neigbors of i (in the event that two distinct neighbors of i point to the same unsatisfied agent in U_i). And so on.

Finally, anyone who cannot reach an unsatisfied agent points to the agent with highest priority, other than himself, holding one of his most preferred objects.

Note that their rule might solve some non-improving cycles when agents cannot reach unsatisfied agents. However, the authors show that the number of consecutive steps in which non-improving cycles take place cannot exceed the number of satisfied agents, thus guaranteeing termination.

Clearly, the persistence property is satisfied as it is explicitly enforced. The "Independence of unsatisfied agents" property is also satisfied. To see why, note that every agent ends up pointing to one of the agents that leads to a shortest path to an unsatisfied agent if such a path exists in the TTC-graph. Hence, it only matters who *satisfied* agents point to, establishing that their rule is a member of our family. Finally, as shown in the original paper, their rule induces a polynomial time mechanism which runs in $O(n^6)$ time.

4.1 Improving-cycles-only rules

We now focus on the rules inducing mechanisms capable of producing an allocation with the desired properties in polynomial time. As mentioned in Section 2.2, the complexity of the members of our family of mechanisms depends on how fast we can compute the F-rule and how many steps are needed until the final allocation is obtained. One way to ensure that the algorithm runs in $O(n)$ steps is by guaranteeing that the F-rule solves at least one improving cycle every $O(1)$ steps. A special class of these rules is the one that guarantees that *all* the cycles solved at each step are improving. Note that this is not accomplished by any of the existing rules.

We now present a new family of rules, the *"Common ordering on agents, individual ordering on objects"* rules, with the property that each member induces a strategy-proof mechanism in which only improving trading cycles are solved. In these rules, two disjoint sets of agents are maintained at each step: labeled and unlabeled. The labeled vertices are those for which their outgoing edge under the F-rule has already been fixed for the current step. The unlabeled vertices are those for whom their outgoing edge is yet to be decided. By growing the set of labeled vertices, we will be able to fully define $F(G)$. The idea is to label all unsatisfied agents first, and then label the unlabeled agents (which can only be satisfied agents) in such a way that every satisfied agent is in a path to an unsatisfied agent in $F(G)$. Therefore, we can guarantee that only improving cycles are formed.

To decide which edges in the TTC-graph G are be selected to form $F(G)$, the rule will use distinct ordering criteria over both agents and objects. In general, we say that an ordering criterion over a set S is *consistent* if, for all sets S_1, S_2 such that $S_1, S_2 \subseteq S$, the relative order of the agents in $S_1 \cap S_2$ agrees in the individual orders of S_1 and S_2 obtained by applying the ordering criterion. We will only require consistency of an ordering in a specific case: at each step m, the order in which unlabeled agents are labeled in both Δ and Δ' is consistent. As indicated by the name of the family, each member uses two different types of orderings: a common consistent ordering over all agents and an individual ordering over the objects for each agent.In sharp contrast with the previous mechanisms, all of the orderings are allowed to change within steps of the algorithm, as long as the consistency property is maintained for the ordering

over agents.

The family of "Common ordering on agents, individual ordering on objects" rules are formally described as follows:

Step 1:

(1.a) Each unsatisfied agent chooses to point to the owner of the highest priority object (according to his own ordering) among all of his top-ranked objects.

(1.b) Repeat until all satisfied agents are labeled:

> *Using a consistent ordering, select the highest priority agent among all the unlabeled agents adjacent to a labeled vertex. Make him point to the owner of the highest priority object (according to his own ordering) among all which are labeled. Label him.*

For each satisfied vertex v, we keep track of the first unsatisfied vertex reachable from v in $F(G)$ and we denote it by $X(v)$. For each unsatisfied vertex v, we denote by $X(v)$ the vertex he points to in $F(G)$. For step k, the rule is as follows:

Step k:

(k.a) Each agent v for which $X(v)$ still holds the same object as in the previous step will continue to point to the same agent as in the previous step. Label all such agents. All other agents remain unlabeled.

(k.b) Each unsatisfied unlabeled agent will point to the owner of the highest priority object (according to his own ordering) among all of his top-ranked objects. Label all unsatisfied agents.

(k.c) Repeat until all the remaining unlabeled agents are labeled:

> *Using a consistent ordering, select the highest priority agent among all the unlabeled agents adjacent to a labeled vertex. Make him point to the owner of the highest priority object (according to his own ordering) among all which are labeled. Label him.*

By the end of each step all satisfied agents will be in a path to an unsatisfied vertex in $F(G)$. Hence, every cycle formed is improving, ensuring termination in $O(n)$ steps. In addition, persistence is satisfied by construction. Thus, only the "Independence of unsatisfied agents" property needs to be verified. To that end, note that once an agent is labeled, the following choices are independent of whom he points to. Since unsatisfied agents start as labeled, all choices are independent of whom they point to.

We show that each member of this family is strategy-proof mechanism by showing that it satisfies condition (b) of Theorem 4.

Theorem 5. *The "Common ordering on agents, individual ordering on objects" rules satisfy condition (b) of Theorem 4. Thus, each rule in this family induces a strategy-proof mechanism.*

Proof. Let A_m^Δ (resp. O_m^Δ) be the set of agents (resp. objects) that remain in the reduced problem obtained from Δ at the beginning of the improvement phase at step m. In addition, let L_m^Δ (resp. UL_m^Δ) be the set that of vertices are labeled (resp. unlabeled) at the beginning of the improvement phase at step m in Δ. As in Section 3.2, let $C_m^\Delta(i)$ be the set of vertices v such that there is a directed path from v to i in $F(G_m^\Delta)$. Given an agent j and an object α, we denote by $o_m(j)$ the object owned by agent j at step m and by $a_m(\alpha)$ the agent who owns α

at step m. We show that "Common ordering on agents, individual ordering on objects" rules induce strategy-proof mechanisms by showing that they satisfy condition (b) of Theorem 4.

Let t (resp. t') be the first step in which agent i trades or is satisfied in Δ (resp. Δ'). By induction in the step number m, we show that $A_m^{\Delta} \subseteq A_m^{\Delta'}$, $O_m^{\Delta} \subseteq O_m^{\Delta'}$, all agents in $A_m^{\Delta'} \backslash C_{m-1}^{\Delta'}(i)$ remain in Δ and are endowed with the same objects in both problems, and condition (b) holds, i.e., each agent $j \in A_m^{\Delta'} \backslash C_m^{\Delta'}(i)$ is in A_m^{Δ} and if j points to agent k in $F(G_m^{\Delta})$ and $\beta = o_m^{\Delta}(k)$, then j points to $a_m^{\Delta'}(\beta)$ in $F(G_m^{\Delta'})$.

As in the proof of Theorem 4, we may assume that we start with the reduced problem obtained at the beginning of step $T = min(t, t') = t$ and we denote these reduced problems by Δ and Δ' respectively. At this point, both problems have the same set of agents endowed with the same objects and all agents but i will have the same preference lists. Since we start our analysis at step T, we rename step T as step 1 before continuing with the proof. At step 1, all same terminal sinks and cycles will be formed except for the one involving i by the same arguments used in the proof of Theorem 4. Therefore, the base case for induction is shown.

Before proceeding to the inductive step, let us illustrate what happens in step 2. Consider an arbitrary terminal sink of Δ_2'. Clearly, that sink depended on one or more cycles of the previous step to be solved and those cycles also appeared in Δ_1. Furthermore, as the set of agents and objects are the same in both problems and none of those vertices has a path to an object in $C_1^{\Delta'}(i)$ in Δ', no such a path will exist in Δ either. Therefore, every sink in Δ' will also appear in Δ.

Let S be a terminal sink in Δ but not in Δ' in step 2. Since S is not a sink in Δ', for every vertex in S there is a path to an unsatisfied vertex in Δ'. Furthermore, all such unsatisfied vertices must be in $C_1^{\Delta'}(i)$, as otherwise that path will be in Δ as well. Hence, all vertices in S will join $C_2^{\Delta'}(i)$ during this current step.

The above remarks hold for every iteration of the removal and update phase at this current step, so we conclude that $A_2^{\Delta} \subseteq A_2^{\Delta'}$, $O_2^{\Delta} \subseteq O_2^{\Delta'}$ and all agents in $A_2^{\Delta'} \backslash C_1^{\Delta'}(i)$ remain in Δ and are endowed with the same objects in both problems. Given that the cycles solved during the first step were the same in both cases (expect for the one including i) and $A_2^{\Delta} \subseteq A_2^{\Delta'}$, we have $L_2^{\Delta} \subseteq L_2^{\Delta'}$. Furthermore, $L_2^{\Delta'} \backslash L_2^{\Delta} \subseteq C_1^{\Delta'}(i)$. In addition, as the cycles solved during the first step were the same in both cases (expect for the one including i), every vertex in $UL_2^{\Delta'} \backslash UL_2^{\Delta}$ must be a vertex that is no longer in Δ (that is, a vertex in $A_2^{\Delta'} \backslash A_2^{\Delta}$), and every unlabeled vertex in $UL_2^{\Delta} \backslash UL_2^{\Delta'}$ must be in $C_1^{\Delta'}(i)$ and it must be endowed in Δ with one of the objects owned by an agent in $C_1^{\Delta'}(i)$. Finally, every agent that is in $UL_2^{\Delta} \cap UL_2^{\Delta'}$ cannot be in $C_1^{\Delta'}(i)$ and thus is endowed with the same object in both problems.

Keeping the above properties in mind, let C be a cycle found in Δ' during the improvement phase in step 2. We show that must be C is solved in Δ and well. Note that $C \cap C_2^{\Delta'}(i) = \emptyset$ and $C_1^{\Delta'}(i) \subseteq C_2^{\Delta'}(i)$. Every satisfied labeled in C must also be satisfied and labeled in Δ as the set of labeled vertices only differs in those in $C_1^{\Delta'}(i)$. Hence, labeled vertices must point to the same object in both problems and that object is owned by the same agent as no trading involving that agent occurred in the previous step.

Let $v \in C$ be an unlabeled vertex. As argued before, all vertices that were in a sink in Δ but not in Δ' will join $C_2^{\Delta'}(i)$, so v could not have been in a sink in Δ. Then, it must still be in both problems. Furthermore, since $L_2^{\Delta} \subseteq L_2^{\Delta'}$, it follows that v must also be unlabeled in Δ. Therefore, $v \in UL_2^{\Delta} \cap UL_2^{\Delta'}$. Finally, all unsatisfied agents in C will point to the same object in both problems, and those objects will be owned by the same agents. This follows from the the fact that in both problems all agents in $UL_2^{\Delta} \cap UL_2^{\Delta'}$ and in $L_2^{\Delta} \cap L_2^{\Delta'}$ will hold the same object in both problems by inductive hypothesis. Since every unsatisfied agent points to the owner of the highest priority object among his top-choices and $O_2^{\Delta} \subseteq O_2^{\Delta'}$, the same choices will be made by those agents in both problems.

From the above, it suffices to show that every vertex in $UL_2^{\Delta} \cap UL_2^{\Delta'}$ who does not point to someone in $C_2^{\Delta'}(i)$ will choose to point to the same object in both problems. We shall show that, as we grow the set of labeled vertices, at all times we have $O(L_2^{\Delta}) \subseteq O(L_2^{\Delta'})$, i.e., the objects owned by the set of label agents in Δ is a subset of those owned by labeled agents in Δ'. Clearly, this holds at the beginning of step 2 since $L_2^{\Delta} \subseteq L_2^{\Delta'}$ and only unlabeled agents may have changed their endowments during the previous step. Furthermore, all unsatisfied agents in Δ will hold their initial endowment in both problems.

Consider now the set of agents in UL_2^{Δ} ordered by *their priority*, and suppose we start the improvement phase. Let u be the highest priority agent among those in UL_2^{Δ}. If $u \in UL_2^{\Delta} \backslash UL_2^{\Delta'}$, he must be in $C_1^{\Delta'}(i) \subseteq C_2^{\Delta'}(i)$, so we do not care who he points to. However, since he is endowed with an object owned by someone in $C_1^{\Delta'}(i)$, the property $O(L_2^{\Delta}) \subseteq O(L_2^{\Delta'})$ is maintained. If $u \in UL_2^{\Delta} \cap UL_2^{\Delta'}$ and he does not point to one of the vertices in $C_2^{\Delta'}(i)$ in the problem Δ', it means that his top-choice among the remaining ones in Δ' is held by an agent that is not in that component. Let that agent be a. Clearly, a must be labeled as well in Δ as $L_2^{\Delta'} \backslash L_2^{\Delta} \subseteq C_1^{\Delta'}(i)$. Furthermore, since $a \notin C_1^{\Delta'}(i)$, he must hold the same objects in both problem by inductive hypothesis. Then, u will end up pointing to a in both problems. In addition, since $u \notin C_1^{\Delta'}(i)$, he must hold the same object in both problems. Therefore, $O(L_2^{\Delta}) \subseteq O(L_2^{\Delta'})$ is maintained. We can reason inductively to show that, each time a vertex u is labeled in Δ, either $u \in UL_2^{\Delta} \backslash UL_2^{\Delta'}$ so the object he owns is already owned by a labeled agent in Δ', or $u \in UL_2^{\Delta} \cap UL_2^{\Delta'}$ and it is endowed with the same object in both problems by inductive hypothesis. Since the vertices in $u \in UL_2^{\Delta} \cap UL_2^{\Delta'}$ will be added in the same relative order in both problems by the consistency property, we conclude that $O(L_2^{\Delta}) \subseteq O(L_2^{\Delta'})$ is maintained at all times. Hence, if $u \in UL_2^{\Delta} \cap UL_2^{\Delta'}$ selects to point to an agent $a \notin C_1^{\Delta'}(i)$ in Δ', a is endowed with the same object in both problems by inductive hypothesis and since the property $O(L_2^{\Delta}) \subseteq O(L_2^{\Delta'})$ is maintained at all times, a will also own the highest priority object of u in Δ. Therefore, u points to a in Δ and the property holds.

We can continue the proof by induction in the step number using the same arguments used for step 2 to show that the following holds at each step $m < t'$: $A_m^{\Delta} \subseteq A_m^{\Delta'}$, $O_m^{\Delta} \subseteq O_m^{\Delta'}$, all agents that are not in $C_{m-1}^{\Delta'}(i)$ are endowed with the same objects in both problems, $L_m^{\Delta} \subseteq L_m^{\Delta'}$ and they only differ in the vertices in $C_m^{\Delta'}(i)$. In addition, every vertex in $UL_m^{\Delta'}$ that is not in UL_m^{Δ} must be a vertex that is no longer in Δ, every unlabeled vertex in UL_m^{Δ} that is not in $UL_m^{\Delta'}$ must be in $C_{m-1}^{\Delta'}(i)$ and it must be endowed with one of those objects in Δ. Finally, every agent that is in $UL_m^{\Delta} \cap UL_m^{\Delta'}$ must be endowed with the same object in both problems. □

4.1.1 Highest Priority Object Rule

We now focus on the *Highest Priority Object (HPO) rule*, which is a member of the "Common ordering on agents, individual ordering on agents" family and therefore induces a strategy-proof mechanism. We will show the mechanism induced by this rule can be implemented in $O(n^2 \log(n) + n^2\gamma)$, where γ is the maximum size of an equivalence class in any preference list.

To properly define this rule, we need to specify the orderings that are used. A common priority ordering over objects is given. This ordering will be used by each agent as his individual ordering over objects. Furthermore, the common ordering over objects will be used to induce the common priority over agents, which will be given by the object they own. Note that this ordering may vary between steps. However, it can be seen that HPO rule is consistent by using the same inductive arguments as in the proof of Theorem 5 to show that, at each step m, all unlabeled agents in $UL_m^{\Delta} \cap UL_m^{\Delta'}$ are endowed with the same object in both problems, thus inducing the same relative order. Hence, the rule induces a strategy-proof mechanism.

Recall from the description of the "Common ordering on agents, individual ordering on

agents" family that, at each step, only agents for which persistence needs to be enforced will start as labeled and all others will remain unlabeled. Keeping that in mind, the rule can be summarized as follows:

Rule 2 (Highest Priority Object (HPO))**.** *A common priority ordering over objects is given. Every unsatisfied agent points to the owner of the highest priority object among his top-choices. Label all unsatisfied agents.*

Let L be the set of labeled vertices, and let AL be the set of all agents adjacent to the set of labeled vertex. At each step, select the agent in AL who owns the highest priority object among all those in AL and make that agent point to the owner of the highest priority object among his top-choices that are in L. Add this agent to L and all of its neighbors that are not in L to AL.

A 6-agent example showing how the HPO rule works is in Figure 3. Figures 1.*a* and 1.*b* show the preferences, endowments, and the common ordering of the objects. The TTC graph obtained in the first step (G_0) is in Figure 1.*c*. There are no terminal sinks, so the improvement phase starts right away: The unsatisfied agents (agents 4, 5 and 6) point to the owner of the highest priority object among his most-preferred objects (3, 6 and 2 respectively). All unsatisfied agents are labeled immediately after and the others remain unlabeled. The only two agents adjacent to a labeled vertex are 2 and 3. As agent 2 owns the highest priority object, he points to the labeled agent that owns the highest priority object among his top-choices. In this case 2 points to 4. Following the same reasoning, agent 3 points to agent 5 and agent 1 to agent 3. We therefore obtain the graph $F(G_0)$ as shown in Figure 1.*d*. This graph contains a unique trading cycle formed by agents 1, 3 and 5. After this trade is implemented, the TTC graph G_1 is obtained. Since there are no terminal sinks in G_1, we move forward to the improvement phase. According to the rule, the two unsatisfied vertices (4 and 6) and the one corresponding to agent 2 start as labeled (as agent 2 point to 4, who still holds the same object as in the previous iteration). Agent 5 is the only unlabeled agent adjacent to a labeled one, so he points to 6. By following the rule, agent 1 points to 5 and agent 3 points to 1 to complete $F(G_1)$. Note that a trading cycle involving all agents but 3 is formed, so by the end of this step all agents will be satisfied.

We next note that the HPO rule is different from both previously known rules, as shown in the examples provided in Figure 4.

Implementation Finally, we discuss the implementation of the mechanism induced by the HPO rule. Recall that the TCRP mechanism can be implemented in $O(n^6)$ and the Top Trading Absorbing Sets mechanism has been shown to run in exponential time in the worst case. We show that the mechanism induced by the Highest Priority Object rule can be implemented in $O(n^2 \log(n) + n^2\gamma)$, where γ is the maximum size of an equivalence class in any preference list, which is considerable improvement over the existing mechanisms.

As usual, let $G = (V, E)$ be the TTC-graph at a given iteration. We will maintain a set AL which will store the all vertices currently pointing to a labeled vertex. Using a Fibonacci heap to implement AL, we can add vertices in $O(1)$, obtain the next vertex that must be labeled in $O(1)$ and delete in $O(\log(|V|))$. We can also assume that we keep track of labeled vertices and those in AL in two arrays, so updating those will take $O(|V|)$ time (throughout all the step) and checking whether a vertex is in them will be done in $O(1)$. Hence, deciding whether a vertex should be added to AL can be done in $O(1)$.

Every time a vertex v is labeled, we need to update AL by checking if each of the vertices with an incoming edge to v need to be added. By storing the set of incoming edges to a vertex (which can be done in $O(|E|)$), we check a total of $O(|E|)$ times per step if a vertex needs to be added to AL (each takes $O(1)$ time). In addition, we perform $O(|V|)$ insertions and $O(|V|)$

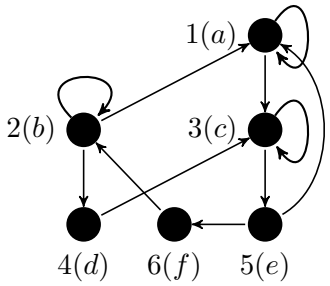
Original endowments:
$\omega(1) = a$, $\omega(2) = b$,
$\omega(3) = c$, $\omega(4) = d$
$\omega(5) = e$, $\omega(6) = f$

Common object order:
a, b, c, d, e, f.

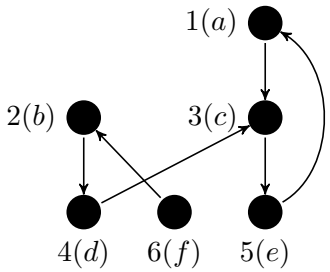
(a) Endowments and orders.

Preference Lists:	
1	{a, c}
2	{a, b, d}
3	{c, e}
4	c
5	{a, f}
6	b

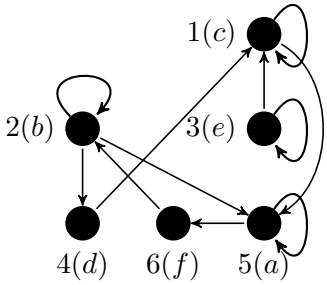
(b) Preferences.



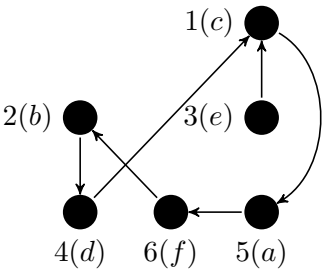
(c) G_0.



(d) $F(G_0)$.



(e) G_1.



(f) $F(G_1)$.

Figure 3: Example illustrating the steps of the mechanism induced by the Highest Priority Object rule. Figures 1.*a* and 1.*b* show the original edowments, the preference profile and the common ordering over the objects. Figure 1.*c* shows the original TTC graph and Figure 1.*d* shows the graph obtained in the improvement phase. Finally, Figure 1.*e* and 1.*f* show the TTC and trading graphs obtained in the second step. Once the improvement phase in the second step concludes, all agents are satisfied and own their final alocation.

deletions from AL per step, each of them requiring $O(1)$ and $O(\log(|V|))$ time respectively. So far, the total number of operations per step are $O(|V|\log(|V|) + |E|)$.

Selecting whom a vertex v should point to can be done in $O(\#$ outgoing edges from v), since we need to check which is the highest priority object which is labeled. The total pointing can be done in $O(|E|)$ and updating $X(v)$ can be done in $O(1)$ for each vertex, giving a total time of $O(|V| + |E|)$. Finding and solving the cycles and updating endowment lists can be done in $O(|V|)$. Then, the improvement phase at each step can be computed in $O(|V|\log(|V|) + |E|)$ time.

In Section 2.2, we argued that at most $O(n + \#$ improving phase steps) iterations of the removal and update phase take place, each one consisting of $O(|V| + |E|)$ operations. Knowing that the total time of steps is $O(n)$, then the total running time can be bounded by $O(n^2\log(n) + n^2\gamma)$, where γ is the maximum size of an equivalence class in any preference list.

5 Discussion

The TTC mechanism is the only mechanism satisfying individual rationality, Pareto-efficiency and strategy-proofness on the strict preference domain. However, when indifferences are permitted, several distinct mechanisms satisfying these properties exist, but a characterization of all

Original endowments:
$\omega = (a, b, c, d)$
Common orders:
agents: $1, 2, 3, 4$.
objects: a, b, c, d.

(a) Endowments and orders. The priority order over agents is used by the TCRP rule, while the order over objects is used by the HPO and TTAS rules

Preference Lists:	
1	{a, c}
2	{a, b, d}
3	b
4	b

(b) Using this preference profile, one can verify that object b is allocated to agent 3 (resp. 4) when using the mechanism induced by the HPO rule (resp. TCRP rule)

Preference Lists:	
1	{a, b, c}
2	{a, b, d}
3	a
4	a

(c) Using this preference profile, one can verify that object a is allocated to agent 3 (resp. 4) when using the mechanism induced by the HPO rule (resp. TTAS rule).

Figure 4: Examples illustrating that the Highest Priority Object rule is different from the TCRP and TTAS rules

such mechanisms is still lacking, and would be interesting. In this work, we take a step forward toward that goal by showing sufficient conditions on a family of mechanisms that guarantee these properties, but we do not know if these conditions are necessary. Also, it would be nice to find necessary and new sufficient conditions for Property 1 to hold.

In Section 3.1 we added additional conditions for the rules that we were willing to consider, namely"Persistence" and "Independence of unsatisfied agents". It would also be interesting to explore the necessity of these conditions. As mentioned earlier, without the "Independence of unsatisfied agents" condition being enforced proving that an abstract selection rule induces a strategy proof mechanism seems to be difficult.

Finally, we note that while the "Common ordering on agents, individual ordering on agents" rules and the rule described in the appendix are very similar (they only differ in the individual orderings used), the former induces strategy-proof mechanism while the latter does not. It would be interesting to be able to formalize what fails in the rule described in the appendix as an intermediate step towards finding necessary conditions for a mechanism to be strategy proof.

Acknowledgments. The authors thank Haris Aziz for comments on the paper. They thank Vikram Manjunath, Paula Jaramillo, Jorge-Alcalde Unzu and Elena Molis for many useful discussions on this problem.

References

[1] Abdulkadiroğlu, A. and Sonmez, T. 1999. House allocation with existing tenants. *Journal of Economic Theory 88,* 2, 233 – 260.

[2] Alcalde-Unzu, J. and Molis, E. 2011. Exchange of indivisible goods and indifferences: The top trading absorbing sets mechanisms. *Games and Economic Behavior 73,* 1, 1–16.

[3] Aziz, H. and de Keijzer, B. 2012. Housing markets with indifferences: A tale of two mechanisms. In *Proceedings of AAAI'12.*

[4] Jaramillo, P. and Manjunath, V. 2012. The difference indifference makes in strategy-proof allocation of objects. *Journal of Economic Theory 147,* 5, 1913 – 1946.

[5] Ma, J. 1994. Strategy-proofness and the strict core in a market with indivisibilities. *International Journal of Game Theory 23,* 1, 75–83.

[6] Plaxton, C. G. 2012. A simple family of top trading cycles mechanisms for housing markets with indifferences.

[7] Roth, A. 1982. Incentive compatibility in a market with indivisible goods. *Economics Letters 9,* 2, 127 – 132.

[8] Roth, A., Sonmez, T., and Unver, U. 2004. Kidney exchange. *Quarterly Journal of Economics 19,* 2, 457 – 488.

[9] Saban, D. and Sethuraman, J. 2013. House allocation with indifferences: a generalization and a unified view. In *Proceedings of the fourteenth ACM conference on Electronic commerce.* EC '13. ACM, 803–820.

[10] Shapley, L. and Scarf, H. 1974. On cores and indivisibility. *Journal of Mathematical Economics 1,* 1, 23 – 37.

[11] Tarjan, R. 1972. Depth-First Search and Linear Graph Algorithms. *SIAM Journal on Computing 1,* 2, 146–160.

APPENDIX

We now present a selection rule which satisfies all of the required conditions (namely, unique pointing, termination, persistence and "Independence of unsatisfied agents"), and still fails to be strategy-proof. This rule, called the *Common ordering on agents, individual ordering on agents rule* is defined as follows:

There is a common ordering on agents and also each agent will has his own (individual) ordering of other agents. All orderings will be fixed throughout the algorithm. At each step, agents will are divided into two sets: labeled agents and unlabeled agents. Labeled agents are those for whom their outgoing edge according to the selection rule has already been decided, while unlabeled agents are those for which who should they point to is yet to be decided. We will grow the set of unlabeled agents until every agent is labeled, and thus the rule is defined.

In the first step, the selection rule is as follows:

Step 1:

(1.a) Each unsatisfied agent points to the highest priority agent (according to his own ordering) among those who own one of his top-ranked objects. Label all unsatisfied agents.

(1.b) Repeat until all satisfied agents are labeled:

> *Select the highest priority agent among all the agents adjacent to a labeled vertex, and make it point to the highest priority agent (according to his own ordering) owning one of his top-choices among all those which are labeled. Label the unlabeled agent.*

For each satisfied vertex v, we keep track of the first unsatisfied vertex reachable from v in $F(G)$ and we denote it by $X(v)$. For each unsatisfied vertex v, we denote by $X(v)$ the vertex he points to in $F(G)$. For step k, the rule is as follows:

Step k:

(k.a) Each agent v for which $X(v)$ still holds the same object as in the previous step will continue to point to the same agent as in the previous step. Label all such agents. All other agents remain unlabeled.

(k.b) Each unsatisfied agent v for which $X(v)$ does not hold the same object as in the previous step will point to the highest priority agent (according to his own ordering) among those who own one of his top-ranked objects. Label all unsatisfied agents.

(k.c) Repeat until all the remaining unlabeled agents are labeled:

> *Select the highest priority unlabeled agent among all those adjacent to a labeled vertex, and make it point to the highest priority agent (according to his own ordering) owning one of his top-choices among all those which are labeled. Label the unlabeled agent.*

Note that by the end of each step all satisfied agents will be in a path to an unsatisfied vertex in $F(G)$. Hence, every cycle formed is improving, ensuring termination in $O(n)$ steps. In addition, persistence is satisfied by construction. Finally, the "Independence of unsatisfied agents" property is satisfied as all unsatisfied agents start as labeled and unlabeled agents choose who to point to based only on priorities over the labeled agents, regardless who labeled agents are pointing to.

In Figure 5, an example is provided which illustrates why this rule fails to induce a strategy proof mechanism. Agent 1 obtains c in problem Δ. We define Δ' from Δ by only modifying agent 1's preferences so that he strictly prefers c over g. Agent 1 gets object c in Δ but fails to obtain it in Δ', implying that whenever i has true preferences which agree with those in Δ', he is better off by reporting his preferences as in Δ. Therefore, the mechanism induced by this rule is not strategy-proof.

Those readers who are already familiar with the results in Section 3.2 should note that the definition of Δ' from Δ agrees with the one considered in Property 1. Therefore, this example shows that the mechanism is not strategy-proof by showing that Property 1 fails to hold. Furthermore, we can see that the sufficient conditions provided in Theorem 4 fail to hold as well, as in the example agent 2 points to someone outside $C_2^{\Delta'}(1)$ in $F(G_2^{\Delta'})$ (therefore $2 \notin C_2^{\Delta'}(1)$) but points to agent 1 in Δ. If the choice was made based solely on objects, then the object would have been be in $C_2^{\Delta'}(1)$ and thus there will not be a conflict, as shown by the rules defined in Section 4.

Original endowments:
$\omega(1) = a$, $\omega(2) = b$,
$\omega(3) = c$, $\omega(4) = d$,
$\omega(5) = e$, $\omega(6) = f$,
$\omega(7) = g$.

Common agent ordering:
1, 2, 3, 4, 5, 6, 7.

(a) Original endowments and common agent ordering for Δ and Δ'

Preference Lists:	
1	{g, c}
2	{f, g, d}
3	{b, e}, c
4	e
5	d
6	b, f
7	a

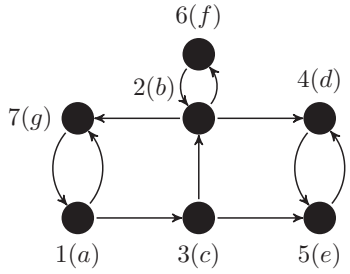
(b) Preferences for Δ. In Δ', agent 1 strictly prefers c over g.

Ind. agent orderings:	
1	7, 3
2	4, 1, 6, 7
3	5, 2
4	5
5	4
6	2
7	1

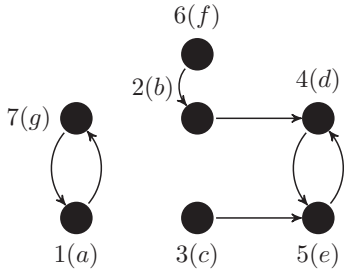
(c) Individual agent orderings for Δ and Δ'.

Final alloc. for Δ:	
$\mu(1)$	c
$\mu(2)$	g
$\mu(3)$	b
$\mu(4)$	e
$\mu(5)$	d
$\mu(6)$	f
$\mu(7)$	a

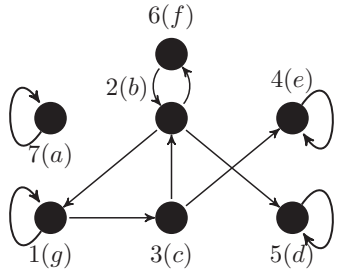
(d) Final allocation for problem Δ.



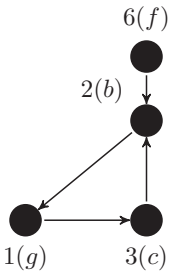
(e) G_0^Δ



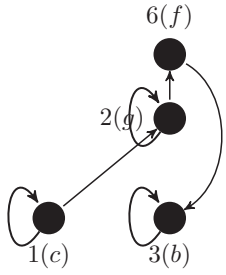
(f) $F(G_0^\Delta)$



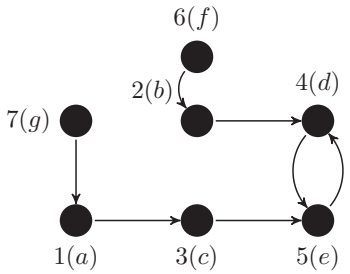
(g) G_1^Δ. Terminal sinks $\{4\}, \{5\}, \{7\}$ are removed.



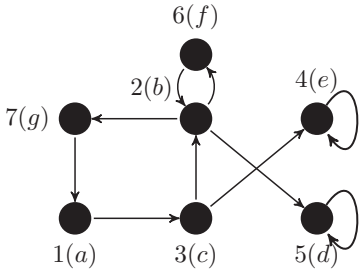
(h) $F(G_1^\Delta)$



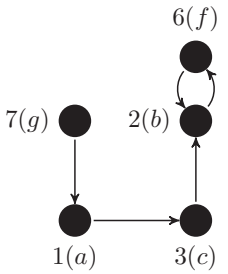
(i) G_2^Δ: everyone leaves with their own objects



(j) $F(G_0^{\Delta'})$



(k) $G_1^{\Delta'}$



(l) $F(G_1^{\Delta'})$: 6 gets b after which 3 leaves with c, so 1 cannot obtain c.

Figure 5: This example shows that "Common ordering on agents, individual ordering on agents" is not strategy-proof. Figures 2.a to 2.d show the initial setting for problems Δ and Δ'. Problem Δ' is defined from Δ by only changing agent 1 preferences so that he strictly prefers c over g. Figures 2.e to 2.i show how the final allocation is computed in Δ. Figures 2.j to 2.l show that agent 1 cannot get c in Δ', implying that whenever i has true preferences agreeing with those in Δ', he will be better off by reporting as in Δ. Hence, strategy-proofness is violated.