Journal of Artificial Intelligence Research 63 (2018) 515-555 Submitted 04/17; published 11/18

# A Complexity Approach for Core-Selecting Exchange under Conditionally Lexicographic Preferences

**Etsushi Fujita** FUJITA@AGENT.INF.KYUSHU-U.AC.JP
*Kyushu University,*
*Motooka 744, Fukuoka, Japan*

**Julien Lesca** JULIEN.LESCA@DAUPHINE.FR
*Université Paris Dauphine, PSL Research University,*
*CNRS, LAMSADE, Paris, France*

**Akihisa Sonoda** SONODA@AGENT.INF.KYUSHU-U.AC.JP
*Kyushu University,*
*Motooka 744, Fukuoka, Japan*

**Taiki Todo** TODO@INF.KYUSHU-U.AC.JP
*Kyushu University,*
*Motooka 744, Fukuoka, Japan*

**Makoto Yokoo** YOKOO@INF.KYUSHU-U.AC.JP
*Kyushu University,*
*Motooka 744, Fukuoka, Japan*

## Abstract

Core-selection is a crucial property of rules in the literature of resource allocation. It is also desirable, from the perspective of mechanism design, to address the incentive of agents to cheat by misreporting their preferences. This paper investigates the exchange problem where (i) each agent is initially endowed with (possibly multiple) indivisible goods, (ii) agents' preferences are assumed to be conditionally lexicographic, and (iii) side payments are prohibited. We propose an exchange rule called augmented top-trading-cycles (ATTC), based on the original TTC procedure. We first show that ATTC is core-selecting and runs in polynomial time with respect to the number of goods. We then show that finding a beneficial misreport under ATTC is NP-hard. We finally clarify relationship of misreporting with splitting and hiding, two different types of manipulations, under ATTC.

## 1. Introduction

Designing rules/mechanisms that achieve desirable properties is a central research topic in the literature of mechanism design and social choice theory. An assignment problem is defined by a set of indivisible goods and a set of agents, and the purpose is to compute an assignment of goods to agents such that no good is assigned more than once. Such an assignment should prescribe a socially desirable outcome in the sense that the assignment should reflect the preferences of the agents over goods. In this paper we study the exchange problems which is a particular assignment problem with following properties: (i) each agent is initially endowed with a set of indivisible goods, (ii) each agent has a strict ordinal preference relation (shortly, a preference) over the set of possible bundles of goods, and (iii) compensation using monetary transfers is prohibited. The ex-

change problem has many real applications, such as on-campus university housing markets (Chen & Sönmez, 2002) and nation-wide kidney exchanges (Roth, Sönmez, & Ünver, 2004).

Core-selection is one of the most well-studied properties that exchange rules are expected to achieve. An exchange rule is said to be core-selecting if no group of agents has an incentive to make a cartel and trade their goods among themselves. By definition, core-selecting rules encourage agents to participate in the rules (i.e., implies individual rationality) and result in a Pareto efficient trade of goods, which in a sense is a socially optimal outcome. When each agent is assumed to be initially endowed with a single indivisible good, Gale's Top-Trading-Cycles (TTC) procedure is known to be core-selecting (Shapley & Scarf, 1974; Roth & Postlewaite, 1977).

Another common requirement is strategy-proofness, which requires that each agent has no incentive to misreport her preference. To be more precise, for each agent, submitting her true preference to the exchange rule is a dominant strategy. This is a quite strong requirement for agents' incentives. Actually, Sönmez (1999) showed that, when there is at least one agent who initially owns more than one good and the agents' preferences over bundles of goods are strict, as in our exchange problems, there exists no rule that is simultaneously strategy-proof and core-selecting in general.

In the wake of this impossibility, we tackle the incentive issue from the perspective of computational complexity. The idea is as follows: even if an agent is selfish and hopes to benefit by misreporting, if finding a beneficial misreport is hard, e.g., it requires to solve an NP-hard problem, and its power of computation is limited, then it will refrain from doing such a manipulation. Under this assumption, in a sense, the NP-hardness of finding a beneficial preference misreport guarantees that agents do not have strong incentive to misreport their preferences. Such a complexity approach for agents' incentives has attracted much attention from computer scientists, especially in the literature of computational social choice (Bartholdi, Tovey, & Trick, 1989; Pini, Rossi, Venable, & Walsh, 2011).

In the exchange problem, agents have preferences over bundles of goods. Describing preferences by an exhaustive ordering of the bundles may be infeasible whenever the number of goods is too large. Indeed, the number of bundles growths exponentially with the number of goods. Therefore, compact representations are worth considering for representing preferences of the agents in our exchange problem. The lexicographic preferences are a well studied restriction over the preference domain which allows a compact representation of the preferences (Saban & Sethuraman, 2014; Todo, Sun, & Yokoo, 2014; Aziz, Kalinowski, Walsh, & Xia, 2016). Indeed, under this restriction the preferences over bundles can be fully described by an ordering over goods. Informally speaking, lexicographic preferences $\succ$ are defined as follows. In order to compare two bundles of goods, we start by comparing their most preferred goods. If the most preferred good in one bundle is preferred to the most preferred good in the other bundle then the former bundle is preferred to the later one. Otherwise, we compare the second most preferred goods of those bundles, and in case of equality, we compare the third most preferred goods, and so on. Let us consider an illustrative example with three goods: beef ($\mathcal{B}$), cake ($\mathcal{C}$) and white wine ($\mathcal{W}$). Suppose that $\mathcal{B}$ is preferred to $\mathcal{C}$ which is preferred to $\mathcal{W}$. In lexicographic preferences model, the preferences over bundle is $\{\mathcal{B},\mathcal{C},\mathcal{W}\} \succ \{\mathcal{B},\mathcal{C}\} \succ \{\mathcal{B},\mathcal{W}\} \succ \{\mathcal{B}\} \succ \{\mathcal{C},\mathcal{W}\} \succ \{\mathcal{C}\} \succ \{\mathcal{W}\}$, where $\succ$ means that the left part is strictly preferred to the right part.

The drawback of focusing on lexicographic preference domain is that such restriction seems too strong to reflect real preferences. In this paper we focus on a larger preference domain over bundles of goods, called *conditionally lexicographic preferences*, which was introduced by Booth,

Chevaleyre, Lang, Mengin, and Sombattheera (2010). Lexicographic preferences are a special case of conditionally lexicographic preferences. But the compact representation used to represent conditionally lexicographic preferences is richer in the sense that the ordering over goods may be conditional to the existence of some more preferred goods in the bundle. Let us change our example by replacing white wine with red wine ($\mathcal{R}$). In that case, the preferences of an agent over the red wine and the cake may be different if she obtains the beef or not. For example, she may prefer $\mathcal{R}$ to $\mathcal{C}$ if she has $\mathcal{B}$, and otherwise she prefers $\mathcal{C}$ to $\mathcal{R}$. In that case, the preferences over bundles would be the following: $\{\mathcal{B},\mathcal{C},\mathcal{R}\} \succ \{\mathcal{B},\mathcal{R}\} \succ \{\mathcal{B},\mathcal{C}\} \succ \{\mathcal{B}\} \succ \{\mathcal{C},\mathcal{R}\} \succ \{\mathcal{C}\} \succ \{\mathcal{R}\}$. Note that such preferences are not lexicographic because $\mathcal{C}$ is once preferred to $\mathcal{R}$ and $\mathcal{R}$ is once preferred to $\mathcal{C}$. Actually, the number of possible conditionally lexicographic preferences is exponentially larger than the number of possible lexicographic preferences (Booth et al., 2010).

In this paper, we propose an exchange rule called *augmented top-trading-cycles (ATTC) procedure* for the exchange problem. The proposed rule possesses nice properties in terms of quality of solutions and agents' incentives. Concerning the quality of solutions, the ATTC procedure is core-selecting. For agents' incentives, finding a beneficial misreport for a manipulator under the ATTC procedure is NP-hard. We also consider two different types of manipulations called *splitting* and *hiding*, and clarify their relationship with preference misreporting. We show that, for any given splitting/hiding manipulation, there exists a corresponding preference misreport that gives the same bundle of goods to the manipulator, and such a misreport can be computed in polynomial time.

The paper is organized as follows. In Section 2, we review several existing works and clarify the difference with our approach. In Section 3, we give the formal model of our exchange problems and define the conditionally lexicographic preferences. In Section 4, we define the ATTC procedure and illustrate its behavior by a simple example. In Section 5, we show that the ATTC procedure is core-selecting. In Section 6, we formalize the problem of finding a beneficial misreport under the ATTC procedure and show that it is NP-hard. In Section 7, we give the definitions of splitting and hiding manipulations, and present two algorithms to emulate these manipulations by misreporting preference. In Section 8, we conclude the paper and give some possible future research directions.

## 2. Related Works

The exchange model we deal with in this paper is a generalization of the well-studied housing market (Shapley & Scarf, 1974), where each agent initially owns a single house, agents' preferences are strict, and monetary transfers are prohibited. In housing market literature, the TTC procedure is characterized by three properties: individual rationality, Pareto efficiency, and strategyproofness (Ma, 1994). Moreover, it always chooses the unique core assignment (Roth & Postlewaite, 1977). Other properties have been considered in AI, e.g., fairness (Endriss, Maudet, Sadri, & Toni, 2006; Lesca & Perny, 2010; Lesca, Minoux, & Perny, 2013) and envy-freeness (Chevaleyre, Endriss, & Maudet, 2007; de Keijzer, Bouveret, Klos, & Zhang, 2009). Constraints on the set of possible allocations have also been considered, e.g., for exchanges over a social network (Gourvès, Lesca, & Wilczynski, 2017). On the other hand, when at least one agent initially owns more than one house, as well as the impossibility result presented in the previous section advents, we can no longer guarantee the uniqueness of the core assignment (Sönmez, 1999). Cechlárová (2009) studied the exchange problem for more than one type of goods and show that deciding the nonemptyness

of the core is NP-hard even for two types of goods. Furthermore, Cechlárová and Lacko (2012) studied the core property for the kidney exchange problem.

One common approach for going beyond such an impossibility result is to weaken one of the requirements. In the literature of exchange with multiple endowments, which is what we are dealing with in this paper, several strategy-proof rules have been developed by weakening the core-selecting property (Pápai, 2003, 2007; Todo et al., 2014). Various restriction over the preference domain have also been considered in the literature, such as binary domain (Luo & Tang, 2015), asymmetric preferences (Sun, Hata, Todo, & Yokoo, 2015), and additive preferences (Sonoda, Fujita, Todo, & Yokoo, 2014; Aziz, Biró, Lang, Lesca, & Monnot, 2016).

Our approach maintains the core-selecting requirement, but weakens strategy-proofness by focusing on the computational hardness of beneficial manipulation. In various mechanism design/social choice problems, many works consider the computational hardness of beneficial manipulation, such as voting (Bartholdi et al., 1989), two-sided matching (Teo, Sethuraman, & Tan, 2001; Pini et al., 2011) and sequential allocation (Aziz, Bouveret, Lang, & Mackenzie, 2017). Although it has been pointed out that such a computational complexity approach is not always sufficient as a barrier for agents' incentives (Faliszewski, Hemaspaandra, & Hemaspaandra, 2010; Faliszewski & Procaccia, 2010), we believe that discussing complexity of misreporting in exchange problems is an important first step toward the development of useful exchange rules for self-interested agents in practice.

In this paper we also focus on the conditionally lexicographic preference domain. This restriction leads to compact representations of the agents' preferences, and makes sense whenever they are non compensatory (Gigerenzer & Goldstein, 1996). Thus such preferences have been well studied in the AI community and especially in elicitation (Booth et al., 2010) and voting (Conitzer & Xia, 2012; Lang, Mengin, & Xia, 2012). Responsive preferences are one of the most famous restriction over the preference domain in matching theory. Such domain of preferences generalize both lexicographic and additive preferences. Informally, the requirement for a preference to be responsive is that, for a given bundle of goods, the marginal contribution of an additional good only depends on the ordering over goods. Note that, according to the illustrative example provided earlier, conditionally lexicographic preferences may not be responsive.

Similar exchange rules as ATTC procedure have been studied in recent papers (Sikdar, Adalı, & Xia, 2017, 2018). Sikdar et al. (2017) consider an exchange problem where goods are partitioned into types (car, house, and so on). Each agent has initially a single good of each type and receives at the end of the procedure a single good of each type. Therefore, preferences are over bundles of goods of same size with one good per type. Agents' preferences are represented by CP-nets where vertices represent types. In short, CP-nets are an oriented graphs which represent conditional preferences over types. The preferences over the goods of a given type will depend on the other goods allocated whose types are parents in the graph of the type under consideration. In the paper, the CP-nets considered are lexicographic i.e., there exists a linear order over the types such that a type can be the ancestor of another one only if the first type precedes the second type in this linear order. This model somehow extends the one presented by Fujita et al. (2015), where agent's preferences over the bundles of goods are lexicographic. However, the model of Sikdar et al. (2017) impose additional constraint on the admissible allocation (exactly one good per type) whereas such constraint does not appear in the paper of Fujita et al. (2015). Conditionally lexicographic preferences considered in this paper generalize the model studied by Sikdar et al. (2017) except that it does not allow indifferences. On the other hand, the model of preferences considered

by Sikdar et al. (2018) generalizes conditionally lexicographic preferences in order to allow indifferences. Sikdar, Adali, and Xia (2017, 2018) show that TTC-like procedures for these two models of preferences are core-selecting.

The effect of splitting manipulations has been studied in several algorithmic/economic environments, such as scheduling (Moulin, 2008), voting (Conitzer, 2008; Todo, Iwasaki, & Yokoo, 2011), combinatorial auctions (Yokoo, Sakurai, & Matsubara, 2004), two-sided matching (Todo & Conitzer, 2013; Afacan, 2014), and coalitional games (Aziz, Bachrach, Elkind, & Paterson, 2011; Yokoo, Conitzer, Sandholm, Ohta, & Iwasaki, 2005; Ohta, Conitzer, Satoh, Iwasaki, & Yokoo, 2008), some of which are also known as *false-name* manipulations. Especially for the case of exchange problem, a class of exchange rules resistant to splitting manipulations, as well as to another class of manipulations called *hiding* (Atlamaz & Klaus, 2007), has been proposed by Todo et al. (2014), while they do not satisfy the core-selecting property.

## 3. Preliminaries

In this section we provide the formal definition for the exchange problem with multiple indivisible goods studied in this paper as well as several properties of exchange rules that have been discussed in the literature.

### 3.1 Notation

We consider set of agents $N = \{1, \dots, n\}$ and finite set of heterogeneous indivisible goods $K$ of size $m$. An *assignment* $\boldsymbol{x} = (x_1, \dots, x_n)$ is a partition of $K$ into $n$ subsets, where $x_i$ is the bundle of goods assigned to agent $i$ under assignment $\boldsymbol{x}$. We denote by $\mathcal{X}$ the set of all possible assignments.

The assignment of the goods to the agents is made based on their preferences, which are orders/rankings over bundles of goods. In this paper we focus on the domain of *conditionally lexicographic preferences*, in which a preferences over the bundles of goods are modeled by lexicographic preference trees (or LP-trees).

**Definition 1** (LP-tree)**.** *An LP-tree over $K$ is a rooted tree such that*

- *every vertex $v$ is labeled with good $g(v)$ belonging to $K$*
- *every good appears once, and only once, on any path from the root to a leaf*
- *every non-leaf vertex $v$ has either one outgoing edge labeled by $g(v) \vee \overline{g(v)}$, or two outgoing edges labeled by $g(v)$ and $\overline{g(v)}$, respectively.*

Figure 1 provides three examples of LP-trees over $K = \{o_1, o_2, o_3, o_4\}$. Let $\mathcal{L}$ be the set of all possible LP-trees over $K$. For any vertex $v$ of an LP-tree, let $a(v)$ be the set of goods labeling the ancestors of $v$, i.e. the vertices on the path from the root to $v$. For example, in $\tau_1$ described in Figure 1, if we define $v$ as the leftmost vertex with label $o_4$ then $a(v) = \{o_1, o_3\}$. An LP-tree $\tau$ represents a preference over bundles in a graphical manner. In order to evaluate a bundle $A \subseteq K$, we follow the directed path starting from the root, ending at one of the leaf and containing only edges consistent with $A$, i.e. for each vertex $v$ visited by this directed path, an edge consistent with $A$ would be labeled by either $g(v)$ or $g(v) \vee \overline{g(v)}$ if $g(v) \in A$, and by either $\overline{g(v)}$ or $g(v) \vee \overline{g(v)}$ otherwise. Note that such a path is unique. Let $\tau(A)$ denote such a path. For example, $\tau_1(\{o_1, o_2\})$ is the path

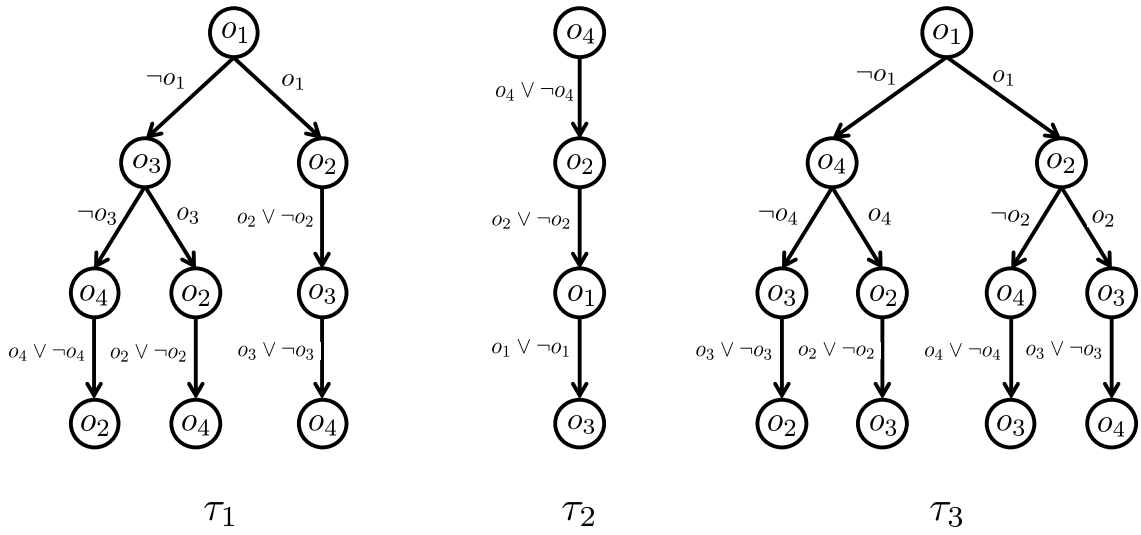



Figure 1: Three examples of LP-trees

from the root that reaches the rightmost leaf, and $\tau_3(\{o_2, o_3\})$ is the path that reaches the leftmost leaf. In order to compare two bundles $A$ and $B$, with $A \neq B$, we consider the first vertex $v$ in the sequence of vertices visited by both $\tau(A)$ and $\tau(B)$ and such that $g(v)$ belongs to exactly one of the two bundles. Let $\tau(A, B)$ denote such vertex. For example, $\tau_1(\{o_2, o_3\}, \{o_2, o_4\})$ is the leftmost vertex labeled by $o_3$ and with two children in $\tau_1$, and $\tau_3(\{o_1, o_2, o_3\}, \{o_1, o_3\})$ is the rightmost vertex labeled by $o_2$ and with two children in $\tau_3$. More formally, $\tau(A, B)$ is the unique vertex that is visited by both $\tau(A)$ and $\tau(B)$ and such that $a(\tau(A, B)) \subseteq A \oplus B$ and $g(\tau(A, B)) \notin A \oplus B$, where $A \oplus B = \{o \in K \mid o \in A \Leftrightarrow o \in B\}$. Preferences associated with LP-trees are defined as follows:

**Definition 2.** *For a given LP-tree $\tau$, let $\succ_\tau$ denote the preference over bundles such that, for any two bundles $A, B \subseteq K$, $A \succ_\tau B$ iff both $g(\tau(A, B)) \in A$ and $g(\tau(A, B)) \notin B$ hold.*

For example, considering the LP-tree $\tau_1$ of Figure 1, we have $\{o_2, o_3\} \succ_{\tau_1} \{o_2, o_4\}$ because $g(\tau_1(\{o_2, o_3\}, \{o_2, o_4\})) = o_3$ and $o_3$ belongs to $\{o_2, o_3\}$ but not to $\{o_2, o_4\}$. Note that both $\succ_{\tau_1}$ and $\succ_{\tau_3}$ are not responsive, and $\succ_{\tau_2}$ is lexicographic because $\tau_2$ is a path. In order to simplify notation, we denote by $\succsim_\tau$ the preference such that for any $A, B \subseteq K$, $A \succsim_\tau B$ iff either $A \succ_\tau B$ or $A = B$ holds.

In the rest of this paper, we assume that agent preferences are conditionally lexicographic and that the preference of each agent is represented by an LP-tree. Using this notation, we define an *exchange problem* $(\boldsymbol{e}, \boldsymbol{\tau})$ by an initial endowment $\boldsymbol{e} = (e_1, \ldots, e_n) \in \mathcal{X}$ and a profile $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_n) \in \mathcal{L}^n$ of LP-trees, where $\tau_i$ denotes the LP-tree that models the preference of agent $i$. For any $i \in N$ and any $\tau_i' \in \mathcal{L}$, we denote by $(\tau_i', \boldsymbol{\tau}_{-i})$ the preference profile $(\tau_1, \ldots, \tau_{i-1}, \tau_i', \tau_{i+1}, \ldots, \tau_n)$. Furthermore, we denote by $\delta(o)$ the owner of $o$ in $\boldsymbol{e}$.

**Example 1.** *Consider the set of goods $K = \{o_1, o_2, o_3, o_4\}$, the set of agents $N = \{1, 2, 3\}$, and the initial endowments $\boldsymbol{e}$ such that $e_1 = \{o_1, o_4\}$, $e_2 = \{o_2\}$, and $e_3 = \{o_3\}$. The LP-trees $\boldsymbol{\tau} = (\tau_1, \tau_2, \tau_3)$ presented in Figure 1 describe the conditionally lexicographic preferences of agents 1, 2 and 3, respectively. The pair $(\boldsymbol{e}, \boldsymbol{\tau})$ defines an exchange problem.*

### 3.2 Exchange Rules and Properties

An *exchange rule* is a function $\varphi : \mathcal{X} \times \mathcal{L}^n \to \mathcal{X}$ that maps any exchange problem to an assignment. Let $\varphi_i(\boldsymbol{e}, \boldsymbol{\tau})$ denote the bundle assigned to agent $i$ under assignment $\varphi(\boldsymbol{e}, \boldsymbol{\tau})$. In this section, we provide the formal definitions of three properties achievable by an exchange rule, namely *individual rationality*, *Pareto efficiency*, and *core selection*. These properties have traditionally been considered as desiderata in the literature of social choice and mechanism design.

**Definition 3** (Individual Rationality)**.** *For a given exchange problem $(\boldsymbol{e}, \boldsymbol{\tau})$, an assignment $\boldsymbol{x} \in \mathcal{X}$ is individually rational if $x_i \succsim_{\tau_i} e_i$ holds for any agent $i$. An exchange rule $\varphi$ is* individually rational (IR) *if for any exchange problem $(\boldsymbol{e}, \boldsymbol{\tau})$, $\varphi(\boldsymbol{e}, \boldsymbol{\tau})$ is individually rational.*

In other words, for each agent, as long as she truthfully reports her preference, she is never worse off by participating in an IR exchange rule. Under such an exchange rule, every agent is incentivized to participate.

**Definition 4** (Pareto Efficiency)**.** *For a given exchange problem $(\boldsymbol{e}, \boldsymbol{\tau})$, an assignment $\boldsymbol{x} \in \mathcal{X}$ is Pareto dominated by another $\boldsymbol{y} \in \mathcal{X}$ if (i) $y_i \succsim_{\tau_i} x_i$ holds for every $i \in N$, and (ii) $y_j \succ_{\tau_j} x_j$ holds for some $j \in N$. An assignment is* Pareto efficient *if it is not Pareto dominated by any other assignment. An exchange rule $\varphi$ is Pareto efficient (PE) if for any exchange problem $(\boldsymbol{e}, \boldsymbol{\tau})$, $\varphi(\boldsymbol{e}, \boldsymbol{\tau})$ is Pareto efficient.*

When an assignment $\boldsymbol{x}$ is Pareto dominated by another assignment $\boldsymbol{y}$, choosing $\boldsymbol{x}$ is sub-optimal. Therefore, using a PE exchange rule is socially optimal, in the sense that it never chooses such sub-optimal assignment.

**Definition 5** (Core Selection)**.** *For a given exchange problem $(\boldsymbol{e}, \boldsymbol{\tau})$, a coalition $T \subseteq N$ of agents blocks an assignment $\boldsymbol{x} \in \mathcal{X}$ if there exists another assignment $\boldsymbol{y} \in \mathcal{X}$ such that (i) $y_i \subseteq \bigcup_{\ell \in T} e_\ell$ for every $i \in T$, (ii) $y_i \succsim_{\tau_i} x_i$ for every $i \in T$, and (iii) $y_j \succ_{\tau_j} x_j$ for some $j \in T$. The* core *$\mathcal{C}(\boldsymbol{e}, \boldsymbol{\tau})$ is the set of assignments that are not blocked by any coalition. An exchange rule $\varphi$ is* core-selecting (CS) *if for any exchange problem $(\boldsymbol{e}, \boldsymbol{\tau})$, $\varphi(\boldsymbol{e}, \boldsymbol{\tau}) \in \mathcal{C}(\boldsymbol{e}, \boldsymbol{\tau})$ holds.*

Condition (i) means that a blocking coalition is restricted to the goods initially owned by the agents in this coalition. Conditions (ii) and (iii) means that, assignment $\boldsymbol{y}$ Pareto dominates $\boldsymbol{x}$ when we only consider the preferences of the agents belonging to the blocking coalition. Intuitively, the existence of a coalition $T$ of agents that blocks an assignment means that they jointly have incentives to form a cartel and get higher utility by leaving behind the set $N \setminus T$ of all the other agents. When an exchange rule is CS, one can expect that all agents participate without forming any such cartel. In this sense, CS can be regarded as a refinement of IR to any possible coalition of agents. Furthermore, by setting $T = N$, the definition coincides with PE. Thus, if an exchange rule is CS, it is also PE and IR.

**Example 2.** *Consider the exchange problem of Example 1. Assignment $\boldsymbol{x}^1 = (\{o_1, o_4\}, \{o_2\}, \{o_3\}) = \boldsymbol{e}$ is obviously individually rational but it is Pareto dominated by $\boldsymbol{x}^2 = (\{o_1, o_3\}, \{o_2\}, \{o_4\})$. On the other hand, $\boldsymbol{x}^3 = (\{o_1, o_2\}, \{o_3\}, \{o_4\})$ is Pareto efficient but not individually rational, since agent 2 is worse off than his initial endowment. To see that, observe that $\tau_2(\{o_3\}, \{o_2\})$ is the unique vertex with label $o_2$ (see Figure 1), and $g(\tau_2(\{o_3\}, \{o_2\})) = o_2$ belongs to $\{o_2\}$ but not to $\{o_3\}$. The assignment $\boldsymbol{x}^2$ is individually rational and Pareto efficient, but not in the core, because $T = \{1, 2\} \subset N$ is a blocking coalition with the assignment $\boldsymbol{x}^4 = (\{o_1, o_2\}, \{o_4\}, \{o_3\})$. Finally $\boldsymbol{x}^4$ is in the core, and automatically individually rational and Pareto efficient.*

**Algorithm 1** The Augmented Top-Trading-Cycles (ATTC) procedure

**Input:** exchange problem $(\boldsymbol{e}, \boldsymbol{\tau})$
**Output:** assignment $\varphi(\boldsymbol{e}, \boldsymbol{\tau})$

$t \leftarrow 1$
$V_t \leftarrow K$
**for each** $i \in N$ **do**
  $X_i^{t-1} \leftarrow \emptyset$
**end for**
**while** $V_t \neq \emptyset$ **do**
  $E_t \leftarrow \emptyset$ ▷ Construction of ATTC graph $G_t = (V_t, E_t)$
  **for each** $o \in V_t$ **do**
    $E_t \leftarrow E_t \cup \{(o, g(\tau_{\delta(o)}(X_{\delta(o)}^{t-1}, X_{\delta(o)}^{t-1} \cup V_t)))\}$
  **end for**
  $\mathcal{W}_t \leftarrow \emptyset$
  **for each** $i \in N$ **do** ▷ Assignment of goods to agents in each cycle of $G_t$
    **if** exists $o \in V_t \cap e_i$ visited by some cycle $\mathcal{C}$ in $G_t$ **then**
      Let $f$ denote the successor of $o$ in $\mathcal{C}$
    **else**
      $X_i^t \leftarrow X_i^{t-1} \cup \{f\}$
      $\mathcal{W}_t \leftarrow \mathcal{W}_t \cup \{o\}$
    **end if**
  **end for**
  $V_{t+1} \leftarrow V_t \setminus \mathcal{W}_t$ ▷ Removal from $G_{t+1}$ of assigned goods
  $t \leftarrow t + 1$
**end while**
**return** $(X_i^{t-1})_{i \in N}$

Combining the fact that CS property implies both IR and PE properties with the general impossibility result by Sönmez (1999), or more straightforwardly with the non-existence of an exchange rule that is IR, PE, and strategy-proof under the lexicographic preference domain by Todo et al. (2014), we can easily conclude that there exists no exchange rule that is CS and strategy-proof, even under the conditionally lexicographic preference domain. Our purpose in this paper is therefore to design an exchange rule that is CS and computationally hard to manipulate, while it is inevitably not strategy-proof.

## 4. Augmented Top-Trading-Cycles Rule

In this section we propose a new exchange rule, called *augmented top-trading-cycles (ATTC)*, which is inspired by, and indeed is a natural extension of, the well-known Gale's Top-Trading-Cycle (TTC) procedure. Informally, TTC is a stepwise algorithm simulating a market place, where at each step, the remaining agents point to the most preferred remaining good according to their preferences. The agents which are part of a cycle during a step of the algorithm are assigned to the good they are pointing to and leave the market. ATTC generalizes TTC to exchange problems where agents may have more than one good by dividing each agent into several atomic agents, each

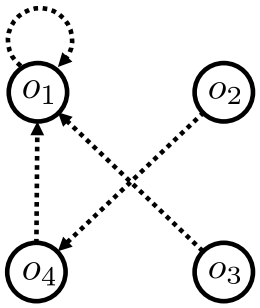



Figure 2: ATTC graph $G_1$

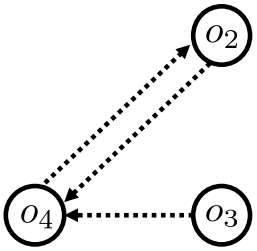



Figure 3: ATTC graph $G_2$


Figure 4: ATTC graph $G_3$

of which is assigned exactly one good from the original agents' endowments. Then the standard TTC procedure is applied to these atomic agents. Obviously, when each agent is initially endowed with a single good, ATTC results in an assignment identical to the one returned by TTC.

In order to apply the TTC procedure to conditionally lexicographic preferences over bundles, we need to know at each step which good is the best to complete the bundle already assigned to one agent. Assume that $X_i^t$ is the bundle of goods assigned to agent $i$ during the first $t$ steps, and $V_{t+1}$ are the remaining goods in the market. The best good to complete $X_i^t$, according to the preference of agent $i$, should be the good $o$ labeling the first vertex on the path $\tau_i(X_i^t)$ with a label in $V_{t+1}$. For example, assume that $V_{t+1} = \{o_2, o_3, o_4\}$ and $X_1^t = \{o_1\}$. In that case, the sequence of labels on the path $\tau_1(\{o_1\})$ is $(o_1, o_2, o_3, o_4)$. Therefore, $o_2$ is the best good to complete $X_i^t$. To see why, one could compare $\{o_1, o_2\}$ and $\{o_1, o_3, o_4\}$ according to agent 1 preference $\succ_{\tau_1}$, and see that $\{o_1, o_2\}$ is preferred to $\{o_1, o_3, o_4\}$. The vertex we are referring to i.e., the first vertex on the path $\tau_i(X_i^t)$ with a label in $V_{t+1}$, is the vertex $\tau_i(X_i^t, X_i^t \cup V_{t+1})$. Algorithm 1 provides a formal description of ATTC.

In the following, we will refer to the graph $G_t$ as the ATTC graph during step $t$. Furthermore, function $\varphi$ will denote the ATTC procedure as described in Algorithm 1. Note that at any step $t$ in the procedure, the ATTC graph $G_t$ contains at least one cycle, and such a cycle contains at least one good, from the characterization of the original TTC procedure. Therefore it is clear that the running-time of ATTC is polynomial with respect to $m$, the number of goods.

**Example 3.** *Consider once again the exchange problem presented in Example 1, where $\delta(o_1) = 1$, $\delta(o_2) = 2$, $\delta(o_3) = 3$, and $\delta(o_4) = 1$. At the beginning of ATTC, agent $1$ is divided into two atomic agents, say $1'$ and $1''$, each of which has the same preference as agent 1, and $o_1$ and $o_4$ are the initial endowments of $1'$ and $1''$, respectively. Furthermore, initially $X_1^0 = X_2^0 = X_3^0 = \emptyset$. Then the original TTC is applied for the market with four atomic agents $1'$, $1''$, $2$ and $3$ with initial endowment $\{o_1\}$, $\{o_4\}$, $\{o_2\}$ and $\{o_3\}$, respectively. The set $V_1$ of vertices of the ATTC graph $G_1$ is $\{o_1, o_2, o_3, o_4\}$, and the arrows of $E_1$ are $(o_1, o_1)$, $(o_4, o_1)$, $(o_3, o_1)$, and $(o_2, o_4)$. The graph $G_1$ is illustrated in Figure 2. During step $1$, vertex $o_1$ is the only vertex included in a cycle, and thus the good $o_1$ is assigned to agent $\delta(o_1) = 1$. Furthermore, after this step $\mathcal{W}_1 = \{o_1\}$, $X_1^1 = \{o_1\}$ and $X_2^1 = X_3^1 = \emptyset$. After the removal of $\mathcal{W}_1$, the set of vertices of the ATTC graph $G_2$ is $V_2 = \{o_2, o_3, o_4\}$, and the arrows are $(o_2, o_4)$, $(o_3, o_4)$, and $(o_4, o_2)$. The graph $G_2$ is illustrated in Figure 3. During step $2$, there is a cycle containing the vertices $o_2$ and $o_4$. Therefore, the good $o_2$ is assigned to agent $\delta(o_4) = 1$ and the*

*good $o_4$ is assigned to agent $\delta(o_2) = 2$. Furthermore, after this step $\mathcal{W}_2 = \{o_2, o_4\}$, $X_1^2 = \{o_1, o_2\}$, $X_2^2 = \{o_4\}$ and $X_3^2 = \emptyset$. After the removal of $\mathcal{W}_2$, the set of vertices of the ATTC graph $G_3$ is $V_3 = \{o_3\}$, and the unique arrow is $(o_3, o_3)$. The graph $G_3$ is illustrated in Figure 4. Finally during step 3, the good $o_3$ is assigned to agent $\delta(o_3) = 3$. Furthermore, after this step $\mathcal{W}_3 = \{o_3\}$, $X_1^3 = \{o_1, o_2\}$, $X_2^3 = \{o_4\}$ and $X_3^3 = \{o_3\}$. After the removal of $\mathcal{W}_3$, $V_4 = \emptyset$ and the algorithm terminates. The final assignment is then $\boldsymbol{x} = (\{o_1, o_2\}, \{o_4\}, \{o_3\})$, which is in the core as we observed in Example 1.*

## 5. Core Selection

We observed, from Example 3, that there is at least one exchange problem for which ATTC returns an assignment in the core. In this section we generalize this observation and show that ATTC inherits the CS property from the original TTC procedure under the conditionally lexicographic preference domain.

In order to introduce this result, we first show that the comparison of two bundles, $A$ and $B$, with $A \neq B$, through a conditionally lexicographic preference $\succ_\tau$ only depends on the goods belonging to $a(\tau(A, B)) \cup g(\tau(A, B))$ i.e., the labels of the vertices visited by both path $\tau(A)$ and path $\tau(B)$ before attaining vertex $\tau(A, B)$. Intuitively, we need to show that if $C$ and $D$ are bundles which differ from $A$ and $B$, respectively, only over the goods outside of $a(\tau(A, B)) \cup g(\tau(A, B))$ then $\tau(A, B)$ and $\tau(C, D)$ should refer to the same vertex of $\tau$. Therefore $A \succ_\tau B$ iff $C \succ_\tau D$ according to Definition 2.

**Lemma 1.** *For any $\tau \in \mathcal{L}$, any $A, B \subseteq K$ and any $C, D \subseteq K$, with $A \neq B$ and $C \neq D$, if $a(\tau(A, B)) \cup g(\tau(A, B)) \subseteq (A \oplus C) \cap (B \oplus D)$ i.e., $o \in A$ iff $o \in C$ and $o \in B$ iff $o \in D$ for any good $o \in a(\tau(A, B)) \cup g(\tau(A, B))$, then $\tau(A, B) = \tau(C, D)$.*

*Proof.* By definition of $\tau(A, B)$, we know that for any $o \in a(\tau(A, B))$ we have $o \in A \Leftrightarrow o \in B$, and this implies that $o \in C \Leftrightarrow o \in D$ since $o \in A \Leftrightarrow o \in C$ and $o \in B \Leftrightarrow o \in D$ hold. Therefore, $a(\tau(A, B)) \subseteq C \oplus D$ holds, and $\tau(A, B)$ is visited by both $\tau(C)$ and $\tau(D)$.

On the other hand, we know that $g(\tau(A, B)) \in A \Leftrightarrow g(\tau(A, B)) \notin B$. Hence $g(\tau(A, B)) \in C \Leftrightarrow g(\tau(A, B)) \notin D$, since $g(\tau(A, B)) \in A \Leftrightarrow g(\tau(A, B)) \in C$ and $g(\tau(A, B)) \in B \Leftrightarrow g(\tau(A, B)) \in D$ hold. This implies that $g(\tau(A, B)) \notin C \oplus D$. Thus, vertex $\tau(A, B)$ is visited by both $\tau(C)$ and $\tau(D)$, and satisfies both $a(\tau(A, B)) \subseteq C \oplus D$ and $g(\tau(A, B)) \notin C \oplus D$. Therefore $\tau(C, D)$ coincides with vertex $\tau(A, B)$. □

Note that during step $t$ of ATTC, all the arrows, from the copies of agent $i$, are pointing to the same good labeling vertex $\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t)$. As discussed earlier, this good is the best to complete the current bundle $X_i^{t-1}$ of agent $i$. The next lemma shows that for any step $t$, the goods labeling the ancestors of $\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t)$ in $\tau_i$ cannot belong to $V_t$.

**Lemma 2.** *For any step $t > 1$ and for any agent $i \in N$, $a(\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t)) \subseteq K \setminus V_t$.*

*Proof.* The bundle $X_i^{t-1}$ of goods acquired by agent $i$ after the first $t - 1$ steps of the procedure cannot contain any good belonging to $V_t$. Hence $X_i^{t-1} \subseteq K \setminus V_t$. This implies that $X_i^{t-1} \oplus (X_i^{t-1} \cup V_t) = K \setminus V_t$. Furthermore, from the definition of $\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t)$, we know that $a(\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t)) \subseteq X_i^{t-1} \oplus (X_i^{t-1} \cup V_t)$. Therefore, we have $a(\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t)) \subseteq K \setminus V_t$. □

Using these two lemmas, we can show that ATTC is core-selecting:

**Theorem 1.** *ATTC is CS.*

*Proof.* For the sake of contradiction we assume that there exists an exchange problem $(\boldsymbol{e}, \boldsymbol{\tau})$ such that the assignment $\varphi(\boldsymbol{e}, \boldsymbol{\tau}) = \boldsymbol{x}$, returned by ATTC, is blocked by a coalition $T \subseteq N$. Let $\boldsymbol{y} \in \mathcal{X}$ be an assignment satisfying conditions (i), (ii), and (iii) described in Definition 5. Let $\mathcal{U}_t$ be the subset of agents that obtain a good during step $t$, let $\mathcal{T}_t = T \cap \mathcal{U}_t$, and let $x_i^t$ be the good obtained by agent $i \in \mathcal{U}_t$ during this step i.e., the unique good contained in $X_i^t \setminus X_i^{t-1}$. Note that such $X_i^t \setminus X_i^{t-1}$ is a singleton because the arrows from all the goods belonging to agent $i$ point to the same good during step $t$.

Here, for each step $t$ in the procedure, consider the three following properties:

- $H_1^t$: for any $i \in \mathcal{T}_t$, $x_i^t \in y_i$ holds,
- $H_2^t$: for any $i \in \mathcal{T}_t$ and any $j \in N$, $x_i^t \in e_j \Rightarrow j \in T$ holds,
- $H_3^t$: for any $i \in \mathcal{T}_t$ and any $o \in a(\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t))$, $o \in x_i \Leftrightarrow o \in y_i$ holds.

If we prove that $H_1^t$ and $H_3^t$ are true for any step $t$, then it will imply that for any $i \in T$, $y_i = x_i$ holds, leading to a contradiction with conditions (iii) of Definition 5 i.e., $y_j \succ_{\tau_j} x_j$ holds for some $j \in T$. Therefore, to conclude the proof it suffices to show that $H_1^t$, $H_2^t$, and $H_3^t$ hold for any step $t$, where $H_2^t$ is an important property to prove $H_3^t$. We will show them by mathematical induction with respect to $t$.

**Base case ($t = 1$):** We show that $H_1^1$, $H_2^1$, and $H_3^1$ hold. Let $i$ be an arbitrary agent in $\mathcal{T}_1$. By definition of Algorithm 1, we know that $x_i^1$ labels the root of $\tau_i$ since $X_i^0 = \emptyset$ and $V_1 = K$. If $x_i^1 \notin y_1$, $g(\tau_i(x_i, y_i)) = x_i^1$ holds, which implies $x_i \succ_{\tau_i} y_i$ and contradicts the condition (ii) of Definition 5 i.e., $y_i \succsim_{\tau_i} x_i$ holds for any $i \in T$. Therefore, $x_i^1 \in y_1$ and thus $H_1^1$ hold.

Let $j \in \mathcal{U}_1$ be an arbitrary agent satisfying $x_i^1 \in e_j$ for some $i \in \mathcal{T}_1$. Assume by contradiction that $j \notin T$. Therefore, $x_i^1 \notin \bigcup_{\ell \in T} e_\ell$ must also hold. However, by property $H_1^1$, we already know that $x_i^1 \in y_i$, which contradicts the condition (i) of Definition 5 i.e., $y_i \subseteq \bigcup_{\ell \in T} e_\ell$ holds. Therefore $H_2^1$ holds.

Finally, $H_3^1$ is trivially true because $\tau_i(\emptyset, V_1)$ is the root of $\tau_i$, implying $a(\tau_i(\emptyset, V_1)) = \emptyset$.

**Induction step:** Assuming that $H_1^l$, $H_2^l$, and $H_3^l$ hold for any $l \in \{1, \dots, t-1\}$, we first show that $H_3^t$ holds. Assume by contradiction that there exist $i \in \mathcal{T}_t$ and $o \in a(\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t))$ such that $o \in x_i \not\Leftrightarrow o \in y_i$. Assume first that $o \in y_i$ and $o \notin x_i$ hold. From the condition (i) of Definition 5, we have $y_i \subseteq \bigcup_{\ell \in T} e_\ell$, implying that $\delta(o) \in T$. Furthermore, $o$ cannot belong to $G_t$ because $o \in a(\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t)) \subseteq K \setminus V_t$, where the inclusion is due to Lemma 2. Therefore, $\exists l \in \{1, \dots, t-1\}$ and $\exists j \in \mathcal{U}_l$ such that $i \neq j$ and $x_j^l = o$. Since $\delta(o) \in \mathcal{T}_l$, we obtains $j \in T$ by repeatedly applying $H_2^l$ along the cycle of $G_l$ containing $x_j^l$. However, from property $H_1^l$, we know that $o \in y_j$ since $j \in \mathcal{T}_l$ and $x_j^l = o$. Thus, we have $o \in y_j$ and $o \in y_i$ with $i \neq j$, which derives a contradiction. So we proved that $\forall o \in a(\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t))$, $o \in y_i \Rightarrow o \in x_i$ holds. Now we claim that this also implies $o \in x_i \Rightarrow o \in y_i$ for any $o \in a(\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t))$. Indeed, otherwise let $v$ be the first vertex on the path from the root of $\tau_i$ to $\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t)$ such that $g(v) \in x_i$ and $g(v) \notin y_i$. For any good $o \in a(v)$, we know that $o \in y_i \Rightarrow o \in x_i$ holds because $a(v) \subseteq a(\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t))$. Furthermore, by definition of $v$, we know that $o \in x_i \Rightarrow o \in y_i$ holds for any $o \in a(v)$. Hence $a(v) \subseteq x_i \oplus y_i$. Finally, by definition of $v$, we know that $g(v) \in x_i$

and $g(v) \notin y_i$. Hence $\tau(x_i, y_i) = v$, which implies that $x_i \succ_{\tau_i} y_i$. This derives a contradiction with the condition (ii) of Definition 5.

We then show that $H_1^t$ holds. Let $i \in \mathcal{T}_t$. Assume by contradiction that $x_i^t \notin y_i$ holds. Property $H_3^t$ implies that $g(\tau_i(x_i, y_i)) = x_i^t$. Therefore $x_i \succ_{\tau_i} y_i$, which is in contradiction with the condition (ii) of Definition 5. Hence $x_i^t \in y_i$.

Finally we show that $H_2^t$ holds. Let $i \in \mathcal{T}_t$ and $j \in N$ such that $x_i^t \in e_j$. If $j \notin T$, then $x_i^t \notin \bigcup_{r \in T} e_r$. However, from property $H_1^t$ we know that $x_i^t \in y_i$, which leads to a contradiction with the condition (i) of Definition 5. Therefore $j \in T$. □

The CS property of ATTC implies that, by utilizing ATTC, it can be checked in polynomial time whether the initial endowment is Pareto efficient under a given profile of conditionally lexicographic preferences. Indeed, ATTC returns an assignment identical to the initial endowments if and only if it is Pareto efficient since Theorem 1 asserts that ATTC returns a PE and IR assignment and an assignment is PE if no other assignment Pareto dominates it. This complements the hardness result related to such a verification under the additive preference domain (de Keijzer et al., 2009), which is a subclass of the responsive preference domain and intersects with the conditionally lexicographic preference domain.

**Corollary 1.** *Under the conditionally lexicographic preference domain, it can be checked in polynomial time, with respect to the number $m$ of goods, whether the initial endowment is Pareto efficient.*

## 6. Complexity of Finding Beneficial Misreport

In practice, even if an agent is selfish and hopes to benefit by misreporting, her computation power is limited (Bartholdi et al., 1989). Under this "bounded rationality" assumption, we expect that an agent will refrain from misreporting, if she needs to solve an NP-hard problem to find a beneficial manipulation. We show in this section that finding a beneficial preference misreport under ATTC is NP-hard, which gives agents reasonable incentives to report their true preferences.

Note that when we restrict LP-trees to paths, the set of conditionally lexicographic preferences corresponds to the lexicographic preference domain. In order to simplify the following proofs, we first show that the set of misreport manipulations under consideration can be restricted to lexicographic preferences. To achieve this aim, we show that for any revealed preference $\tau_i$ by agent $i$, there exists an LP-tree $\tau_i'$ which is a path and provides the exact same outcome under ATTC.

**Proposition 1.** *For any exchange problem $(\boldsymbol{e}, \boldsymbol{\tau})$, if agent $i$ reveals $\tau_i'$ instead of $\tau_i$, where $\tau_i'$ is the restriction of LP-tree $\tau_i$ to the path $\tau_i(\varphi_i(\boldsymbol{e}, \boldsymbol{\tau}))$, then ATTC returns the exact same outcome. More formally, $\varphi(\boldsymbol{e}, \boldsymbol{\tau}) = \varphi(\boldsymbol{e}, (\tau_i', \boldsymbol{\tau_{-i}}))$.*

*Proof.* Let $G_t = (V_t, E_t)$ (resp. $G_t' = (V_t', E_t')$) denote the ATTC-graph during step $t$ when $\tau_i$ (resp. $\tau_i' = \tau_i(\varphi_i(\boldsymbol{e}, \boldsymbol{\tau}))$) is revealed by agent $i$, and let $X_j^t$ (resp. $Y_j^t$) denote the set of goods assigned to agent $j$ during the first $t$ steps of the procedure. We show, by mathematical induction with respect to $t$, that for any $t$, $G_t = G_t'$ and $X_j^t = Y_j^t$ for any $j \in N$. This property would imply that $\varphi(\boldsymbol{e}, \boldsymbol{\tau}) = \varphi(\boldsymbol{e}, (\tau_i', \boldsymbol{\tau_{-i}}))$ holds.

**Base case ($t = 1$):** At the fist step of the procedure, we have $V_1' = V_1 = K$ and the sets of edges, $E_1$ and $E_1'$, only depend on the roots of the revealed LP-trees since $\tau_j(\emptyset, K)$ is the root of $\tau_j$ for any $j \in N$. For each $j \in N \setminus \{i\}$, the revealed LP-trees are exactly the same by definition.

Furthermore, the good labeling the root of $\tau_i'$ is the same as the good labeling the root of $\tau_i$. Therefore $G_1 = G_1'$, an this implies that $X_j^1 = Y_j^1$ for any $j \in N$.

**Induction step:** Assuming that $G_{t-1} = G_{t-1}'$ and $X_j^{t-1} = Y_j^{t-1}$ for any $j \in N$, we now consider step $t$ of the procedure. From the definition of Algorithm 1, $V_t = V_t'$ holds and implies that $\tau_j(X_j^{t-1}, X_j^{t-1} \cup V_t) = \tau_j(Y_j^{t-1}, Y_j^{t-1} \cup V_t')$ holds for any $j \in N \setminus \{i\}$. In other words, this means that the outgoing edge of any good belonging to an agent of $N \setminus \{i\}$ is exactly the same in both $G_t$ and $G_t'$. It remains to show that the outgoing edge of any good $o$ belonging to agent $i$ is the same in both $G_t$ and $G_t'$.

First of all, we claim that vertex $\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t)$ is visited by path $\tau_i(\varphi_i(\boldsymbol{e}, \boldsymbol{\tau}))$. Indeed, assume by contradiction that there exists $o \in a(\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t)) \subseteq K \setminus V_t$, where the inclusion is due to Lemma 2, such that $o \notin X_i^{t-1}$ and $o \in \varphi_i(\boldsymbol{e}, \boldsymbol{\tau})$ (the case $o \in X_i^{t-1}$ and $o \notin \varphi_i(\boldsymbol{e}, \boldsymbol{\tau})$ is impossible since $X_i^{t-1} \subseteq \varphi_i(\boldsymbol{e}, \boldsymbol{\tau})$). But in that case, it is clear that $o \notin X_i^{t-1} \cup V_t$, and this implies that agent $i$ cannot obtain $o$ during ATTC and $o \notin \varphi_i(\boldsymbol{e}, \boldsymbol{\tau})$, leading to a contradiction.

We claim now that $g(\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t)) = g(\tau_i'(Y_i^{t-1}, Y_i^{t-1} \cup V_t'))$. By contradiction assume that $\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t)$ and $\tau_i'(Y_i^{t-1}, Y_i^{t-1} \cup V_t')$ do not refer to the same vertex on $\tau_i(\varphi(\boldsymbol{e}, \boldsymbol{\tau}))$, and assume that $\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t)$ is the first vertex visited by $\tau_i(\varphi(\boldsymbol{e}, \boldsymbol{\tau}))$. Note that this means that vertex $\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t)$ is an ancestor of $\tau_i'(Y_i^{t-1}, Y_i^{t-1} \cup V_t')$, and $g(\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t))$ belongs to $a(\tau_i'(Y_i^{t-1}, Y_i^{t-1} \cup V_t'))$. We know by Lemma 2 that $a(\tau_i'(Y_i^{t-1}, Y_i^{t-1} \cup V_t')) \subseteq K \setminus V_t'$, and this implies $g(\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t))$ does not belong to $V_t'$. On the other hand, $g(\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t)) \in V_t$ implies that $g(\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t)) \in V_t'$ because $V_t = V_t'$, leading to a contradiction. The proof is essentially the same when $\tau_i'(Y_i^{t-1}, Y_i^{t-1} \cup V_t')$ is the first vertex visited by $\tau_i(\varphi_i(\boldsymbol{e}, \boldsymbol{\tau}))$.

Finally, by the definition of Algorithm 1 we know that the outgoing edge of any good $o$ belonging to agent $i$ is the same in both $G_t$ and $G_t'$. Hence we have shown that $G_t$ and $G_t'$ are the same, which implies that $X_j^t = Y_j^t$ holds for any $j \in N$. □

From Proposition 1, the search for a beneficial manipulation can be restricted, without loss of generality, to the lexicographic preference domain. Let $\mathcal{P}$ be the subset of $\mathcal{L}$ restricted to paths. We formalize the manipulation problem under ATTC as follows:

**Definition 6** (Beneficial-Misreport)**.**

***Instance:*** *exchange problem* $(\boldsymbol{e}, \boldsymbol{\tau})$ *and agent* $i \in N$.

***Question:*** *is there* $\tau_i' \in \mathcal{P}$ *such that* $\varphi_i(\boldsymbol{e}, (\tau_i', \boldsymbol{\tau}_{-i})) \succ_{\tau_i} \varphi_i(\boldsymbol{e}, \boldsymbol{\tau})$*?*

Note that the size of $\mathcal{P}$ is exponentially smaller than the size of $\mathcal{L}$, and thus focusing the search on $\mathcal{P}$ is an advantage. However, The following theorem shows that it is still computationally hard even though the search can be restricted. The proof is provided in Appendix B.

**Theorem 2.** Beneficial-Misreport *is NP-complete.*

## 7. Possible Benefit by Preference Misreport

One may find that the discussion on complexity provided in the previous section only focused on the worst case behavior of ATTC. Actually, even though the optimization problem is NP-hard, an agent might find the optimal misreport in many cases. In this section we provide an important

observation on this issue; the most preferred an agent originally obtains by truth-telling cannot be improved by any preference misreport.

To be more precise, we introduce some additional notation. Let $o^*$ denote the first good acquired by agent $i$ when she reveals her true preferences $\tau_i$. Note that $o^*$ is the most preferred good acquired by agent $i$ during ATTC i.e., the singleton containing $o^*$ is preferred by agent $i$ to any singleton containing a good acquired by her during ATTC when she reveals $\tau_i$. The following proposition shows that $o^*$ is also the most preferred good that agent $i$ can obtain under ATTC by misreporting her true preferences. The proof is provided in Appendix D.

**Proposition 2.** *For any exchange problem $(\boldsymbol{e}, \boldsymbol{\tau})$, any agent $i \in N$ and any misreport $\tau_i' \in \mathcal{P}$, there is no good $o$ in $\varphi_i(\boldsymbol{e}, (\tau_i', \boldsymbol{\tau}_{-i}))$ which is strictly preferred by agent $i$ to the most preferred good $o^*$ of $\varphi_i(\boldsymbol{e}, \boldsymbol{\tau})$. More formally, $\nexists o \in \varphi_i(\boldsymbol{e}, (\tau_i', \boldsymbol{\tau}_{-i}))$ such that $o \succ_{\tau_i} o^*$*

Proposition 2 provides another evidence that, under ATTC, agents may have a reasonable incentive to report their preferences truthfully. Indeed, under conditionally lexicographic preference domain, an agent mainly cares about the favorite good in her bundle and considers all the other goods as extra. Proposition 2 thus guarantees that agents can benefit only by improving such extra.

## 8. Extended Model with Private Endowments

In this section, we consider the situation where each agent can use multiple accounts and the set of her accounts, as well as her initial endowment, is her private information. Each agent can deceive the exchange rule by pretending to be multiple agents under different accounts (splitting accounts, or shortly *splitting*). We assume that an agent can declare different preferences under different accounts, implying that splitting is more general than misreporting a preference of a single account. However, to our surprise, it turns out that the sets of possible outcomes by misreporting and splitting coincide under ATTC.

Another type of possible manipulations by an agent in this situation is to withhold some of her initial endowments (hiding endowments, or shortly *hiding*), which has already been investigated in the literature of exchange problems (Atlamaz & Klaus, 2007; Todo et al., 2014). For example, assume that agent $i$ has two laptops $\mathcal{A}$ and $\mathcal{B}$. $\mathcal{A}$ suits her working style much better than $\mathcal{B}$, and she also wants a desktop computer $\mathcal{C}$ that is owned by another agent $j$. If $i$ knows that $j$ accepts to trade $\mathcal{C}$ with either $\mathcal{A}$ or $\mathcal{B}$, $i$ may have an incentive to withhold $\mathcal{A}$ and swap $\mathcal{B}$ with $\mathcal{C}$. We show that for any hiding manipulation, there exists a preference misreport that results in the same assignment for the manipulator under ATTC.

### 8.1 Splitting Accounts

Let us first formally define splitting accounts. Under ATTC, we can assume without loss of generality that a manipulator $i$ uses as many accounts as the number of goods in her initial endowment, each of which is endowed with a single good. The reason not to consider splitting manipulation using less accounts is that, under ATTC, an agent is divided into atomic agents, and therefore the outcome obtained by using less than $|e_i|$ accounts can also be obtained by using exactly $|e_i|$ accounts.

For an agent $i \in N$ with initial endowment $e_i$, a splitting manipulation $\boldsymbol{s}_i$ is described as a profile of LP-trees $(\tau_o)_{o \in e_i} \in \mathcal{P}^{|e_i|}$. Here, each $\tau_o$ denotes the LP-tree revealed by the account

**Algorithm 2** Misreport-for-Splitting

**Input:** exchange problem $(\boldsymbol{e}, \boldsymbol{\tau})$, manipulator $i$ and splitting manipulation $\boldsymbol{s}_i \in \mathcal{S}(e_i)$
**Output:** LP-tree $\tau_i' \in \mathcal{P}$ providing the same outcome as $\boldsymbol{s}_i$ for agent $i$

**while** Exists $o, o' \in e_i$ visited by the same cycle $\mathcal{C}$ in $\varphi(\boldsymbol{e}, (\boldsymbol{s}_i, \boldsymbol{\tau}_{-i}))$. **do**
  Let $f$ and $o'$ be the goods following $o$ and $o'$ in $\mathcal{C}$, respectively.
  Let $\lambda$ and $\lambda'$ be arbitrary LP-trees of $\mathcal{P}$ with roots labeled by $f$ and $f'$, respectively.
  $\tau_o \leftarrow \lambda'$
  $\tau_{o'} \leftarrow \lambda'$
**end while**
$\tau_i' \leftarrow \emptyset$
$R \leftarrow K$
**while** $e_i \cap R \neq \emptyset$ **do**
  Let $\mathcal{C}$ be the last cycle visiting a good $o$ of $e_i$ in $\varphi|_R(\boldsymbol{e}, (\boldsymbol{s}_i, \boldsymbol{\tau}_{-i}))$.
  Let $B$ denote the set of goods visited by $\mathcal{C}$, and let $f$ be the good following $o$ in $\mathcal{C}$.
  $\tau_i' \leftarrow$ *add_root*$(\tau_i', o)$
  $R \leftarrow R \setminus B$
**end while**
Complete $\tau_i'$ arbitrarily by inserting the remaining goods at its tail.

corresponding to the good $o$ of $e_i$. Note that we have restricted the possible manipulations of the different accounts to lexicographic preferences $\mathcal{P}$. This is without loss of generality because, according to Proposition 1, any splitting manipulation using conditionally lexicographic preference can be replaced by a splitting manipulation using only lexicographic preferences and with the same outcome.

Let $\mathcal{S}(e_i)$ denote the set of all possible splitting manipulations for a given initial endowment $e_i$. Also, for a given exchange problem $(\boldsymbol{e}, \boldsymbol{\tau})$, an agent $i \in N$, and a splitting manipulation $\boldsymbol{s}_i \in \mathcal{S}(e_i)$, let $\varphi_i(\boldsymbol{e}, (\boldsymbol{s}_i, \boldsymbol{\tau}_{-i}))$ denote the bundle assigned to the accounts owned by agent $i$ when she uses the splitting manipulation $\boldsymbol{s}_i$.

In the following proposition, we clarify the relationship between misreporting and splitting in the ATTC procedure. Indeed, we show that for any splitting manipulation, there exists a preference misreport that returns the same assignment to the manipulator. The proof of Proposition 3 is in Appendix F.

**Proposition 3** (Splitting $\rightarrow$ Misreport)**.** *For any exchange problem $(\boldsymbol{e}, \boldsymbol{\tau})$, any manipulator $i \in N$, and any splitting manipulation $\boldsymbol{s}_i \in \mathcal{S}(e_i)$, Algorithm 2 returns a misreport $\tau_i' \in \mathcal{P}$ such that $\varphi_i(\boldsymbol{e}, (\tau_i', \boldsymbol{\tau}_{-i})) = \varphi_i(\boldsymbol{e}, (\boldsymbol{s}_i, \boldsymbol{\tau}_{-i}))$.*

By abuse of notation, in Algorithm 2 we denote by $\varphi(\boldsymbol{e}, (\boldsymbol{s}_i, \boldsymbol{\tau}_{-i}))$ the ATTC procedure applied to the exchange problem $(\boldsymbol{e}, (\boldsymbol{s}_i, \boldsymbol{\tau}_{-i}))$ i.e., not only the outcome of this procedure, but also the different steps to obtain this outcome. In the same vein, we denote by $\varphi|_R(\boldsymbol{e}, (\boldsymbol{s}_i, \boldsymbol{\tau}_{-i}))$ the ATTC procedure applied to the exchange problem restricted to the goods of $R$ (any agent without good in the restricted problem is removed). Note that, during ATTC, it may be the case that multiple accounts of the manipulator are involved in the same cycle. The first **while** loop of Algorithm 2 provides a splitting manipulation, under which agent $i$ obtains exactly the same bundle and no two goods in $e_i$ are involved in the same cycle during ATTC. This part is convenient to ease the proof.

The second **while** loop constructs an LP-tree $\tau_i'$ by adding one by one the goods (more precisely the vertices labeled by the goods) according to their appearance during the ATTC procedure. The construction of $\tau_i'$ relies on the procedure *add_add*, which adds a new vertex labeled by a given good $o$ on top of $\tau_i'$. The two main ideas behind Algorithm 2 are the following: (i) if an account of agent $i$ trades its good $o$ for good $o'$ during ATTC, then changing its preference by putting $o'$ on top does not change the outcome because the account owns only $o$, and (ii) if an account of agent $i$ is not able to acquire a good $o'$ (even by misreporting) then changing its preference by putting $o'$ at the bottom does not change the outcome.

One can consider the decision problem of checking if a beneficial splitting manipulation exists. The definition of this problem, called Beneficial-Split, is similar to Definition 6 except that the set of available manipulations is the set of splitting manipulations $\mathcal{S}(e_i)$ instead of $\mathcal{P}$. Proposition 3 implies that, while splitting accounts provides much richer manipulations than misreporting preferences (and actually any misreport can be obviously represented as a splitting), the spaces of the possible outcomes by splitting and misreporting coincide. From this observation, combined with the fact that Algorithm 2 runs in polynomial time, we have the following corollary:

**Corollary 2.** Beneficial-Split *is NP-complete.*

### 8.2 Hiding Endowments

We then turn to consider manipulations by hiding endowments. A hiding manipulation $\boldsymbol{h}_i$, operated by a manipulating agent $i$ with initial endowment $e_i$, is represented by a triple $(e_i^r, e_i^w, \tau_i')$ such that $e_i^r \cup e_i^w = e_i$, $e_i^r \cap e_i^w = \emptyset$, and $\tau_i' \in \mathcal{P}$. The first two components $e_i^r$ and $e_i^w$ indicate the set of goods *revealed* to ATTC and the set of goods *withheld* by agent $i$, respectively. The third component $\tau_i'$ indicates the preference reported by agent $i$. Recall that the set of preference revealed can be restricted to lexicographic preferences without loss of generality.

Let $\mathcal{H}(e_i)$ denote the set of all possible hiding manipulations by an agent $i$ who initially owns an endowment $e_i$. Given exchange problem $(\boldsymbol{e}, \boldsymbol{\tau})$, agent $i \in N$, and hiding manipulation $\boldsymbol{h}_i = (e_i^r, e_i^w, \tau_i') \in \mathcal{H}(e_i)$, let $\varphi_i(\boldsymbol{e}, (\boldsymbol{h}_i, \boldsymbol{\tau}_{-i})) := \varphi_i((e_i^r, \boldsymbol{e}_{-i}), (\tau_i', \boldsymbol{\tau}_{-i}))$ denote the bundle assigned to agent $i$ under ATTC when she operates hiding manipulation $\boldsymbol{h}_i$. Note that since the manipulator is withholding $e_i^w$, she finally obtains $\varphi_i(\boldsymbol{e}, (\boldsymbol{h}_i, \boldsymbol{\tau}_{-i})) \cup e_i^w$. The following proposition is the counterpart of Proposition 3 for hiding manipulation.

**Proposition 4** (Hiding $\to$ Splitting)**.** *For any exchange problem $(\boldsymbol{e}, \boldsymbol{\tau})$, any manipulator $i \in N$, and any hiding manipulation $\boldsymbol{h}_i \in \mathcal{H}(e_i)$, Algorithm 3 returns a splitting manipulation $\boldsymbol{s}_i$ such that $\varphi_i(\boldsymbol{e}, (\boldsymbol{s}_i, \boldsymbol{\tau}_{-i})) = \varphi_i(\boldsymbol{e}, (\boldsymbol{h}_i, \boldsymbol{\tau}_{-i})) \cup e_i^w$.*

The proof of this proposition is in Appendix E. Algorithm 3 creates a splitting manipulation by constructing the LP-trees of the different accounts belonging to the manipulator one by one. If a good was hidden then the corresponding account prefers its good to the others. If a good was not hidden then the corresponding account has the same preference as the one used in the hiding manipulation.

One can consider the decision problem of checking if a beneficial hiding manipulation exists. The definition of this problem, called Beneficial-Hide, is similar to Definition 6 except that the set of available manipulations is the set of hiding manipulations $\mathcal{H}(e_i)$ instead of $\mathcal{P}$. Note that hiding is another generalization of misreporting preference because the manipulator is able to misreport her preferences in addition to hide her endowment. Therefore, combining Proposition 4

**Algorithm 3** Splitting-for-Hiding

**Input:** exchange problem $(\boldsymbol{e}, \boldsymbol{\tau})$, manipulator $i$ and hiding manipulation $(e_i^r, e_i^w, \tau_i')$
**Output:** splitting manipulation $\boldsymbol{s}_i$ providing the same outcome for agent $i$

**for each** $o$ in $e_i$ **do**
  **if** $o \in e_i^w$ **then**
    Let $\lambda$ be an arbitrary LP-tree of $\mathcal{P}$ with root labeled by $o$.
    $\tau_o \leftarrow \lambda$
  **else**
    $\tau_o \leftarrow \tau_i'$
  **end if**
**end for**
$\boldsymbol{s}_i \leftarrow (\tau_o)_{o \in e_i}$

with Proposition 3, we can see that the space of possible outcomes by hiding also coincides with the one by misreporting. Therefore, we obtain the following corollary.

**Corollary 3.** Beneficial-Hide *is NP-complete.*

## 9. Conclusions

In this paper we investigated the exchange problem where each agent initially has a set of indivisible goods and a conditionally lexicographic preference. We proposed the ATTC rule, which is core-selecting and runs in polynomial-time. The ATTC rule may also be used to verify, in polynomial time, if the initial endowment is Pareto efficient. Concerning agents' incentives, we showed that finding a beneficial misreport under ATTC is NP-complete, while it is inevitably not strategy-proof due to Sönmez's finding. We also showed that the NP-completeness of finding a beneficial manipulation can be extended to the case where the ownerships of endowments are private, so that each agent can use splitting/hiding manipulations.

A future direction for this work would be to characterize the ATTC rule. Since the core assignment is not always unique in our model with multiple endowments, there might be a different exchange rule that is also core-selecting and runs in polynomial time. For such a characterization, we must discover unique properties of ATTC. Another possible extension would be to consider a larger set of preferences than conditionally lexicographic preferences. CP-nets (Boutilier, Brafman, Domshlak, Hoos, & Poole, 2004) may be used to compactly represent preferences over bundle of goods, including ones containing indifferences between bundles. Actually, LP-trees are a subclass of CP-nets. Contrary to LP-trees, CP-nets can represent any type of preferences over bundles of goods. It would be interesting to define a subclass of CP-nets, larger than LP-trees, where the ATTC algorithm can be extended to provide an assignment in the core.

## Acknowledgments

A preliminary version of this paper was appeared in the proceeding of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15) (Fujita et al., 2015). This work was partially supported by JSPS KAKENHI Grant Number JP24220003, JP26730005, JP17H00761, JP17H04695, JSPS

Program for Advancing Strategic International Networks to Accelerate the Circulation of Talented Researchers, and the Okawa Foundation for Information and Telecommunications. We thank Jérôme Lang for his advice to extend the paper to conditionally lexicographic preferences, as well as participants in the Warsaw Workshop on Economic and Computational Aspects of Game Theory and Social Choice at University of Warsaw, AAAI-15, and the Meeting of COST Action IC1205 on Computational Social Choice at University of Glasgow. All errors are our own.

## Appendix A. Structure of Appendix

In this appendix we provide several technical materials, including some proofs omitted in the main part of the paper. This appendix is organized as follows: Appendix B provides the proof of NP-completeness of Beneficial-Misreport. Appendix C provides several lemmas related to ATTC, which are useful for proving other technical materials. Appendix D contains some lemmas used to compare the behavior of ATTC for two different profiles of preferences, and the proof of Proposition 2. Appendix E provides the proof of Proposition 4. Finally, Appendix F provides the proof of Proposition 3.

## Appendix B. Omitted Proof for Section 6

The reduction is from the following NP-complete problem (Gold, 1978):

**Definition 7** (Monotone (M) 3SAT)**.**

***Instance:*** *set $U$ of variables, collection $C = \{c_1, \ldots, c_m\}$ of $m$ clauses over $U$ such that each clause $c_j$ has three literals, denoted $c_j^1$, $c_j^2$ and $c_j^3$, which are either only negated variables or only un-negated variables.*

***Question:*** *is there a truth assignment of the variables satisfying all clauses.*

We can assume without loss of generality that clause indices are such that all clauses containing only un-negated variable precede clauses containing only negated variables.

*Proof.* Beneficial Misreport is clearly in NP since ATTC runs in polynomial time. The reduction from an instance of M3SAT is as follows. For each clause $c_i$, we create 3 goods $\kappa_i$, $\xi_i$ and $\xi_i'$. Furthermore, for any literal $c_i^\ell$, we create a good $\chi_i^\ell$. To these $9m$ goods, we add two goods $\mu$ and $\mu'$. Other goods will be introduced later on, but for sake of simplicity we focus first on this subset of goods.

The initial endowment of the manipulator is composed by the goods $\chi_i^\ell$ for any $i$ and $\ell$, as well as good $\mu$. The true preference of the manipulator is

$$\kappa_1 \succ \xi_1 \succ \xi_1' \succ \kappa_2 \succ \xi_2 \succ \xi_2' \succ \ldots \succ \kappa_m \succ \xi_m \succ \xi_m' \succ \mu' \succ \chi_m^3 \succ \ldots$$

The manipulator will be the only agent owning more than one good. For sake of simplicity, we identify other agents by their goods. Figure 5 describes the preferences of these agents. Solid-line edge denotes the most preferred good, dash-line edge the second most preferred good, and doted-line edge either the third or fourth most preferred good. Furthermore, the doted-circles are missing goods that will be presented later on. For the time being, we can assume that inner edges and outer edges of these doted-circles are directly connected.

We are now able to present the main idea of the reduction. When the manipulator reveals his true preferences, he obtains his favorite goods except for $\mu'$. Indeed, during ATTC he first exchanges $\mu$ with $\kappa_1$, and then $\chi_1^1$ with $\xi_1$, $\chi_1^2$ with $\xi_1'$, $\chi_1^3$ with $\kappa_2$, and so on. At the end, the manipulator is not able to obtain $\mu'$ since he has already exchanged $\mu$ with another good. Therefore, he keeps his good $\chi_m^3$.

Note that the only way for him to be better of is to obtain his most preferred goods i.e., $\mu'$ instead of $\chi_m^3$. But to obtain such assignment, he must exchange

- $\mu$ with $\mu'$
- $\chi_i^{\ell_i}$ with $\kappa_i$
- $\chi_i^{\ell_i'}$ with $\xi_i$
- $\chi_i^{\ell_i''}$ with $\xi_i'$

where for each clause $c_i$, $\{\ell_i, \ell_i', \ell_i''\} = \{1, 2, 3\}$. The good $\chi_i^\ell$ exchanged with $\kappa_i$ will correspond to the literal rendering true clause $c_i$. We are now ready to introduce the remaining goods.

The aim of the goods occulted in Figure 5 is to maintain consistency on the choice of literals rendering true clauses. In other words, we want to avoid a situation where two goods $\chi_i^\ell$ and $\chi_j^k$, corresponding to a variable and its negation, are exchanged by the manipulator in order to obtain goods $\kappa_i$ and $\kappa_j$. To do so, for any pair of literals $c_i^\ell$ and $c_j^k$, such that $c_i^\ell$ is $u$ and $c_j^k$ is $\neg u$ for some variable $u \in U$, we introduce goods $\alpha_{j,k}^{i,\ell}$, $\beta_{j,k}^{i,\ell}$ and $\gamma_{j,k}^{i,\ell}$. The preferences of the owners of $\alpha_{j,k}^{i,\ell}$, $\beta_{j,k}^{i,\ell}$ and $\gamma_{j,k}^{i,\ell}$ are illustrated in Figure 6. In this figure, the most preferred (resp. second most preferred) good for the owner of $\alpha_{j,k}^{i,\ell}$ (resp. $\gamma_{j,k}^{i,\ell}$) is $\alpha_{j',k'}^{i,\ell}$ (resp. $\gamma_{j,k}^{i',\ell'}$), where $\alpha_{j',k'}^{i,\ell}$ (resp. $\gamma_{j,k}^{i',\ell'}$) is one of the goods introduced because literal $c_{j'}^{k'}$ (resp. $c_{i'}^{\ell'}$) is $\neg u$ (resp. $u$) and $c_{j'}$ (resp. $c_{i'}$) is the latest clause preceding $c_j$ (resp. $c_k$) that contains $\neg u$ (resp. $u$) as one of its literals. If $c_j$ (resp. $c_i$) is the first clause containing $\neg u$ (resp. $u$) as one of its literals, then the most preferred (resp. second most preferred) good for the owner of $\alpha_{j,k}^{i,\ell}$ (resp. $\gamma_{j,k}^{i,\ell}$) is $\chi_i^\ell$ (resp. $\chi_j^k$). Finally, the outgoing edge of $\kappa_i$ (resp. $\kappa_j$), which partially appears in Figure 5, is directed toward the $(\ell+1)^{\text{th}}$ most preferred good (resp. $(k+1)^{\text{th}}$ most preferred good) for the owner of $\kappa_i$ (resp. $\kappa_j$). This good is denoted in Figure 5 by $\alpha_{j'',k''}^{i,\ell}$ (resp. $\gamma_{j,k}^{i'',\ell''}$), where $\alpha_{j'',k''}^{i,\ell}$ (resp. $\gamma_{j,k}^{i'',\ell''}$) is one of the good introduced because literal $c_{j''}^{k''}$ (resp. $c_{i''}^{\ell''}$) is $\neg u$ (resp. $u$) and $c_{j''}$ (resp. $c_{i''}$) is the last clause that contains $\neg u$ (resp. $u$) as one of its literals.

To see how this gadget works, assume that the manipulator exchanges $\chi_i^\ell$ to obtain $\kappa_i$, corresponding to the case where $c_i$ is true because of literal $c_i^\ell$ (i.e. positive occurrence of variable $u$). This exchange involves a cycle in the ATTC-graph including the goods on the unique path from $\kappa_i$ to $\chi_i^\ell$. This path includes good $\alpha_{j,k}^{i,\ell}$ which is exchanged. The owner of $\beta_{j,k}^{i,\ell}$ does not belong to this path, and therefore he keeps his good. Furthermore, the most preferred remaining good for him after this exchange becomes $\gamma_{j,k}^{i,\ell}$. Since $\beta_{j,k}^{i,\ell}$ is the most preferred good for the owner of $\gamma_{j,k}^{i,\ell}$, the exchange between these two goods is performed at the next step of ATTC. After this exchange performed, the unique possible path from $\kappa_j$ to $\chi_j^k$ is broken and the manipulator cannot use anymore $\chi_j^k$ to obtain $\kappa_j$.

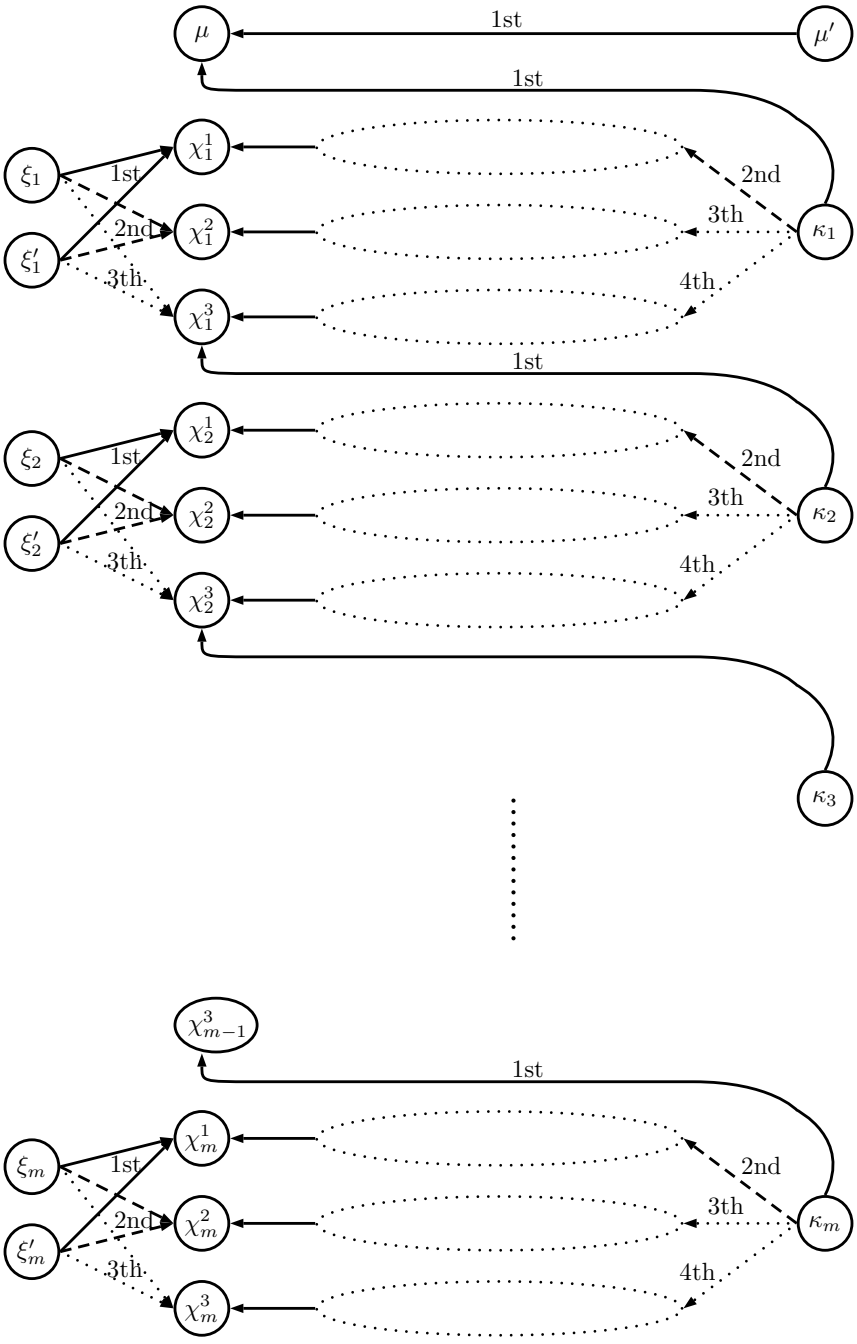



Figure 5: Partial representation of the reduction

On the other hand, assume now that the manipulator wants to use $\chi_j^k$ to obtain $\kappa_j$, corresponding to the case where $c_j$ is true because of literal $c_j^k$ (i.e. negative occurrence of variable $u$). For this exchange to be possible, it is required for good $\beta_{j,k}^{i,\ell}$ to be exchanged with another good

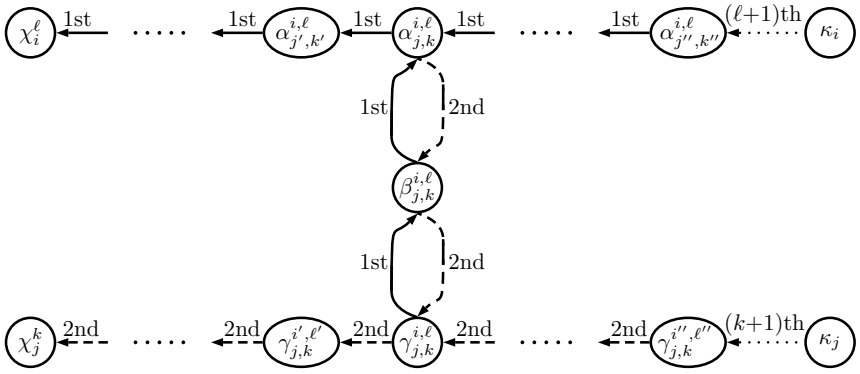



Figure 6: Goods on the paths from $\kappa_i$ to $\chi_i^\ell$ and from $\kappa_j$ to $\chi_j^k$

than $\gamma_{j,k}^{i,\ell}$. But in that case, $\beta_{j,k}^{i,\ell}$ must be exchanged with $\alpha_{j,k}^{i,\ell}$, and thus $\chi_i^\ell$ cannot be used by the manipulator to obtain $\kappa_i$.

To conclude the proof, we show that there is a truth assignment satisfying all clauses in the M3SAT instance iff there is a beneficial manipulation where the manipulator receives his $3m+1$ most preferred goods. If there is truth assignment $\Upsilon$ satisfying all clauses in the M3SAT instance then we construct a beneficial manipulation step by step as follows. Good $\mu'$ will be at the top of the preference revealed by the manipulator. Thus, the first exchange performed during ATTC will be between $\mu$ and $\mu'$. After this exchange performed, the most preferred remaining good for the owner of $\kappa_1$ changes and a path from $\kappa_1$ to $\chi_1^1$ appears in the ATTC-graph. If literal $c_1^1$ is true according to $\Upsilon$ then the second good in the preference revealed by the manipulator is $\kappa_1$, and the third and fourth goods are $\xi_1$ and $\xi_1'$, respectively. In that case, $\kappa_1$ being the most preferred remaining good according to the preference revealed by the manipulator, there is a cycle including $\chi_1^1$ and the manipulator obtains $\kappa_1$ in exchange of $\chi_1^1$. Afterward, $\chi_1^2$ and $\chi_1^3$ are exchanged with $\xi_1$ and $\xi_1'$, respectively. Note that in that case, each good $\alpha_{j,k}^{1,2}$ is exchanged with $\beta_{j,k}^{1,2}$ since the most preferred good for the owner of $\alpha_{j,k}^{1,2}$ is not in the ATTC-graph anymore. For the same reason, $\alpha_{j,k}^{1,3}$ is exchanged with $\beta_{j,k}^{1,3}$. On the other hand, if $c_1^1$ is not true according to $\Upsilon$ then the second good in the preference revealed by the manipulator is $\xi_1$, and $\chi_1^1$ is exchanged with $\xi_1$. Note that in that case, each good $\alpha_{j,k}^{1,1}$ is exchanged with $\beta_{j,k}^{1,1}$. Therefore, the second most preferred good for the owner of $\kappa_1$ is removed from the ATTC-graph, and there is a path from $\kappa_1$ to $\chi_1^2$. In that case, if $c_1^2$ is true according to $\Upsilon$ then the third and fourth goods in the preference revealed by the manipulator are $\kappa_1$ and $\xi_1'$, and both goods are obtained by the manipulator. On the other hand, if literal $c_1^2$ is not true according to $\Upsilon$ then literal $c_1^3$ is true and the third and fourth goods in the preference revealed by the manipulator are $\xi_1'$ and $\kappa_1$. We continue in a similar fashion to construct the rest of the preference profile revealed by the manipulator, and it is easy to check that at the end he obtains his $3m+1$ most preferred goods.

Assume now that there is a beneficial manipulation where the manipulator receives his $3m+1$ most preferred goods. Note first that $\mu$ must be exchanged with $\mu'$. Furthermore, for any clause $c_i$ there must be two goods $\chi_i^{\ell'}$ and $\chi_i^{\ell''}$ exchanged with goods $\xi_i$ and $\xi_i'$, because otherwise one of this good will not be obtained by the manipulator. Therefore, the remaining good, say $\chi_i^\ell$, will be the only one available to obtain good $\kappa_j$ for a clause $c_j$ possibly different to $c_i$. If $c_i$ is a clause

containing un-negated variable then $\chi_i^\ell$ must be exchanged to obtain $\kappa_i$ because otherwise the manipulator will not be able to obtain $\kappa_i$, leading to a contradiction. Indeed, assume by contradiction that another good, say $\chi_j^k$ is used by the manipulator to obtain $\kappa_i$. By construction, literal $c_j^k$ must be a negated occurrence of $c_i^\ell$, because otherwise no path from $\kappa_i$ to $\chi_j^k$ are possible in the ATTC-graph. But on the other hand, the path from $\kappa_i$ to $\chi_j^k$ passes through goods $\alpha_{j,k}^{i,\ell}$ and $\beta_{j,k}^{i,\ell}$. Therefore, both goods must be part of the remaining goods in the ATTC-graph. Since $\alpha_{j,k}^{i,\ell}$ is the most preferred good according to the preference of the owner of $\beta_{j,k}^{i,\ell}$, the path should stop with a cycle including $\alpha_{j,k}^{i,\ell}$ and $\beta_{j,k}^{i,\ell}$, leading to a contradiction. Finally, if $c_i$ is a clause containing negated variable then $\chi_i^\ell$ must also be exchanged to obtain $\kappa_i$ because there is no possible path in the ATTC-graph from $\kappa_i$ to a good $\chi_j^k$ corresponding to a negated variable. All in all, for each clause $c_i$, there is one good $\chi_i^\ell$ which is used to obtain $\kappa_i$. Furthermore, by construction we know that no pair of goods $\chi_i^\ell$ and $\chi_j^k$, corresponding to a variable and its negation, are exchanged by the manipulator in order to obtain goods $\kappa_i$ and $\kappa_j$. Therefore, one can easily construct a truth assignment of the variables in $U$ such that each literal $c_i^\ell$, corresponding to a good $\chi_i^\ell$ exchanged with $\kappa_i$, is true. Such assignment satisfies all clauses of $C$ and this concludes the proof.

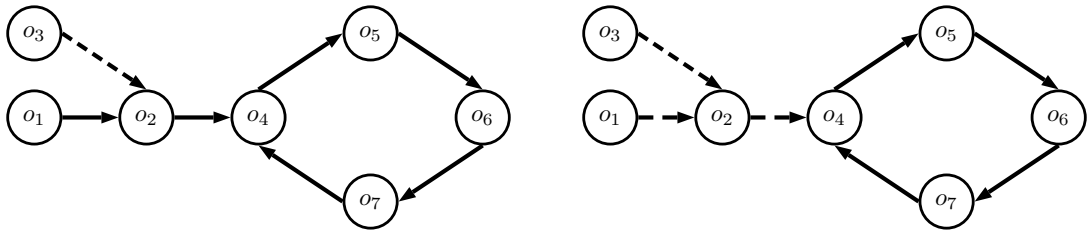
□

## Appendix C. Basic Properties of ATTC

We compile in this appendix all the basic lemmas which will be of help to prove more elaborate results. We start by providing notations which will be used in all appendices. Consider an assignment problem $(\boldsymbol{e}, \boldsymbol{\tau})$ involving a set of agents or identities $N$. In this appendix we do not make any hypothesis on the existence of manipulators using multiple identities. These manipulators may or may not exist and all the revealed identities are considered as agents belonging to $N$. Furthermore, the revealed preference profile $\boldsymbol{\tau}$ may or may not contain misreport, and this preference profile, as well as the initial endowment $\boldsymbol{e}$, are just part of the input of the ATTC procedure.

Let $G_k = (V_k, E_k)$ be the ATTC graph during the step $k$ of ATTC applied to $(\boldsymbol{e}, \boldsymbol{\tau})$, and let $X_i^k$ be the set of goods assigned to agent $i$ after the $k$ first steps. Note that for any step $k$ and for any good $o$ belonging to $V_k$, there exists a unique directed path in $G_k$ which starts from $o$ and visits various goods of $V_k$ before cycling to one of the goods already visited. Let $p_k(o)$ denote this directed path in $G_k$, and let $S_k(o)$ be the set of goods visited by $p_k(o)$. Figure 7a illustrates with solid-line edges such path $p_k(o_1)$ which could appear in $G_k$. In this example, the set of goods $S_k(o_1)$ visited by $p_k(o_1)$ is $\{o_1, o_2, o_4, o_5, o_6, o_7\}$. The dash-line edges in Figure 7a represents edges of $E_k$ which are not part of $p_k(o_1)$. Figure 7b illustrates with solid-line edges directed path $p_k(o_4)$ in $G_k$, which starts from good $o_4$. Note that for any step $k$ of the ATTC procedure and for any good $o$ belonging to $V_k$, the directed path $p_k(o)$ is unique because the outdegree of any vertex in $G_k$ is equal to 1.

It can be seen that the directed path $p_k(o_4)$ is a subpath of $p_k(o_1)$ (see Figure 7). More generally, for any good $o'$ belonging to $S_k(o)$, the directed path $p_k(o')$ is a subpath of $p_k(o)$. Furthermore, set $S_k(o_4)$ is included in set $S_k(o_1)$. Note also that $p_k(o_4)$ is a cycle containing goods $S_k(o_4) = \{o_1, o_2, o_3, o_4\}$. Therefore $p_k(o_5), p_k(o_6)$ and $p_k(o_7)$ correspond to the same cycle as $p_k(o_4)$, and $S_k(o_4) = S_k(o_5) = S_k(o_6) = S_k(o_7)$ hold.




(a) Oriented path $p_k(o_1)$ in plain edges

(b) Oriented path $p_k(o_4)$ in plain edges

Figure 7: Example of ATTC graph $G_k$

Finally for any good $o'' \in S_k(o)$ visited by $p_k(o)$, let $[p_k(o)]_o^{o''}$ be the subpath of $p_k(o)$ which starts from good $o$ and finishes at good $o''$. For example, in Figure 7 $[p_k(o_1)]_{o_1}^{o_5}$ is the subpath of $p_k(o_1)$ which contains edges $(o_1, o_2)$, $(o_2, o_4)$ and $(o_4, o_5)$.

The following lemma states that if the most preferred remaining good for agent $i$ in $G_k$ and $G_{k+1}$ are the same then no good can be assigned to agent $i$ during step $k$.

**Lemma 3.** *For any step $k \geq 1$ of Algorithm 1, for any good $o$ belonging to $V_k$ and for any good $f$ such that $(o, f)$ belongs to $E_k$, if $f$ belongs to $V_{k+1}$ then $X^k_{\delta(o)} = X^{k-1}_{\delta(o)}$.*

*Proof.* Note first that $f \in V_{k+1}$ implies that no cycle including $f$ belongs to $G_k$. Furthermore, by definition of Algorithm 1 we know that for any good $o'$ belonging to $e_{\delta(o)} \cap V_k$, $(o', f)$ is the unique outgoing edge from $o'$ in $G_k$. Therefore, $o'$ cannot be visited by a cycle in $G_k$ because otherwise $f$ would be also visited by that cycle. Thus, no good belonging to agent $\delta(o)$ is exchanged during step $k$, and $X^k_{\delta(o)} = X^{k-1}_{\delta(o)}$ holds. □

The following lemma shows that if a good labels an ancestor of the current vertex of $\tau_i$ under consideration during step $k$ then it will also be the case during ulterior steps.

**Lemma 4.** *For any step $k \geq 1$ of Algorithm 1 and for any agent $i \in N$, the following inclusion holds: $a(\tau_i(X_i^{k-1}, X_i^{k-1} \cup V_k)) \subseteq a(\tau_i(X_i^k, X_i^k \cup V_{k+1}))$.*

*Proof.* We know by Lemma 2 that $a(\tau_i(X_i^{k-1}, X_i^{k-1} \cup V_k)) \subseteq K \setminus V_k$. Furthermore, by definition of Algorithm 1 we know that $K \setminus V_k \subseteq K \setminus V_{k+1}$ and $X_i^k \subseteq K \setminus V_{k+1}$, implying that $X_i^k \oplus (X_i^k \cup V_{k+1}) = K \setminus V_{k+1}$. Hence, $a(\tau_i(X_i^{k-1}, X_i^{k-1} \cup V_k)) \subseteq X_i^k \oplus (X_i^k \cup V_{k+1})$ holds. Moreover, by definition of Algorithm 1 we know that $X_i^{k-1} \oplus (X_i^{k-1} \cup V_k) = K \setminus V_k$, implying that $a(\tau_i(X_i^{k-1}, X_i^{k-1} \cup V_k)) \subseteq X_i^{k-1} \oplus (X_i^{k-1} \cup V_k)$ holds. On the other hand, $K \setminus V_k \subseteq X_i^{k-1} \oplus X_i^k$ holds since the only good that may differ from $X_i^{k-1}$ to $X_i^k$ should be part of $V_k$. All in all, for any $o \in a(\tau_i(X_i^{k-1}, X_i^{k-1} \cup V_k))$, we have $o \in X_i^k \cup V_{k+1} \Leftrightarrow o \in X_i^k \Leftrightarrow o \in X_i^{k-1} \Leftrightarrow o \in X_i^{k-1} \cup V_k$. This means that $\tau_i(X_i^{k-1}), \tau_i(X_i^k), \tau_i(X_i^{k-1} \cup V_k)$ and $\tau_i(X_i^k \cup V_{k+1})$ have in common the subpath from the root to $\tau_i(X_i^{k-1}, X_i^{k-1} \cup V_k)$, and therefore $a(\tau_i(X_i^{k-1}, X_i^{k-1} \cup V_k)) \subseteq a(\tau_i(X_i^k, X_i^k \cup V_{k+1}))$ holds. □

**Corollary 4.** *For any steps $t$ and $r$ of Algorithm 1 such that $r \geq t$ and for any agent $i \in N$, the following inclusion holds: $a(\tau_i(X_i^{t-1}, X_i^{t-1} \cup V_t)) \subseteq a(\tau_i(X_i^{r-1}, X_i^{r-1} \cup V_r))$.*

The following lemma shows that during any step $k$, any ancestor of $\tau_i(X_i^{k-1}, X_i^{k-1} \cup V_k)$ which is not assigned to agent $i$ would be preferred by her to any goods of $V_k$.

**Lemma 5.** *For any step $k \geq 1$ of Algorithm 1, for any agent $i \in N$ and for any set $D \subseteq K \setminus X_i^{k-1}$, if $a(\tau_i(X_i^{k-1}, X_i^{k-1} \cup V_k)) \cap D \neq \emptyset$ then $g(\tau_i(X_i^{k-1}, X_i^{k-1} \cup V_k \cup D)) \in D \setminus V_k$.*

*Proof.* By contradiction, assume that $g(\tau_i(X_i^{k-1}, X_i^{k-1} \cup V_k \cup D)) \notin D \setminus V_k$. Since $D \subseteq K \setminus X_i^{k-1}$. By definition of $D$, we have $(X_i^{k-1} \cup V_k) \oplus (X_i^{k-1} \cup V_k \cup D) = K \setminus (D \setminus V_k)$. Furthermore, by definition of $\tau_i$ we have $a(\tau_i(X_i^{k-1}, X_i^{k-1} \cup V_k \cup D)) \subseteq K \setminus (V_k \cup D) \subseteq K \setminus (D \setminus V_k)$. This implies that $a(\tau_i(X_i^{k-1}, X_i^{k-1} \cup V_k \cup D)) \cup g(\tau_i(X_i^{k-1}, X_i^{k-1} \cup V_k \cup D)) \subseteq (X_i^{k-1} \cup V_k \cup D) \oplus (X_i^{k-1} \cup V_k)$ holds. Therefore, by Lemma 1 we have $\tau_i(X_i^{k-1}, X_i^{k-1} \cup V_k \cup D) = \tau_i(X_i^{k-1}, X_i^{k-1} \cup V_k)$. But this means that $a(\tau_i(X_i^{k-1}, X_i^{k-1} \cup V_k \cup D)) = a(\tau_i(X_i^{k-1}, X_i^{k-1} \cup V_k))$, which implies that $a(\tau_i(X_i^{k-1}, X_i^{k-1} \cup V_k \cup D)) \cap D \neq \emptyset$. Therefore, there is at least one good $o$ belonging to $a(\tau_i(X_i^{k-1}, X_i^{k-1} \cup V_k \cup D)) \cap D$. It is clear that $o \notin X_i^{k-1}$ because $o \in D \subseteq K \setminus X_i^{k-1}$. Hence, $o$ does not belong to $X_i^{k-1} \oplus (X_i^{k-1} \cup V_k \cup D) = K \setminus (V_k \cup D)$, leading to a contradiction with $o \in a(\tau_i(X_i^{k-1}, X_i^{k-1} \cup V_k \cup D))$. □

The following lemma shows that if the most preferred remaining good in $V_k$ for an agent is not exchanged during step $k$ then it remains her most preferred good in $V_{k+1}$ for her.

**Lemma 6.** *For any step $k \geq 1$ of Algorithm 1 and for any edge $(o, f)$ in $E_k$, if $o$ and $f$ belong to $V_{k+1}$ then $(o, f) \in E_{k+1}$.*

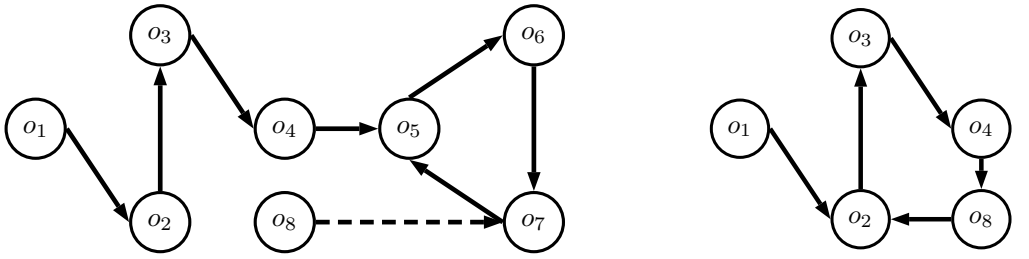
*Proof.* By Lemma 3, $(o, f) \in E_k$ and $f \in V_{k+1}$ imply that $X_{\delta(o)}^k = X_{\delta(o)}^{k-1}$ holds. Hence, $X_{\delta(o)}^k \oplus X_{\delta(o)}^{k-1} = K$ holds. On the other hand, Lemma 2 implies $a(\tau_{\delta(o)}(X_{\delta(o)}^{k-1}, X_{\delta(o)}^{k-1} \cup V_k)) \subseteq K \setminus V_k$. Moreover, by definition of Algorithm 1 we have $V_{k+1} \subseteq V_k$ and $X_{\delta(o)}^{k-1} \subseteq K \setminus V_k$, which imply with $X_{\delta(o)}^k = X_{\delta(o)}^{k-1}$ that $(X_{\delta(o)}^{k-1} \cup V_k) \oplus (X_{\delta(o)}^k \cup V_{k+1}) = (K \setminus V_k) \cup V_{k+1} \supseteq K \setminus V_k$ holds. All in all, we obtain $a(\tau_{\delta(o)}(X_{\delta(o)}^{k-1}, X_{\delta(o)}^{k-1} \cup V_k)) \subseteq (X_{\delta(o)}^{k-1} \cup V_k) \oplus (X_{\delta(o)}^k \cup V_{k+1})$.

Finally, by definition of Algorithm 1 we know that $g(\tau_{\delta(o)}(X_{\delta(o)}^{k-1}, X_{\delta(o)}^{k-1} \cup V_k)) = f$ since $(o, f) \in E_k$. Furthermore, $f$ belongs to $V_{k+1} \subseteq V_k$. This implies that $X_{\delta(o)}^k \oplus X_{\delta(o)}^{k-1}$ and $(X_{\delta(o)}^{k-1} \cup V_k) \oplus (X_{\delta(o)}^k \cup V_{k+1})$ are both supersets of $g(\tau_{\delta(o)}(X_{\delta(o)}^{k-1}, X_{\delta(o)}^{k-1} \cup V_k)) \cup a(\tau_{\delta(o)}(X_{\delta(o)}^{k-1}, X_{\delta(o)}^{k-1} \cup V_k))$. Therefore, Lemma 2 implies that $\tau_{\delta(o)}(X_{\delta(o)}^{k-1}, X_{\delta(o)}^{k-1} \cup V_k) = \tau_{\delta(o)}(X_{\delta(o)}^k, X_{\delta(o)}^k \cup V_{k+1})$ holds. Hence, by definition of Algorithm 1 we obtain $(o, f) \in E_{k+1}$. □

The result of Lemma 6 can be extended from a simple edge into a directed path of the ATTC graph. For example, Figure 8 illustrates the ATTC graph during two successive steps. The goods of $B = \{o_5, o_6, o_7\}$ belong to a cycle during step $k - 1$, as illustrated in Figure 8a. Hence, the goods of $B$ vanish from $G_k$ ,as illustrated in Figure 8b. Figure 8 illustrates that the goods of $S_{k-1}(o_1)$ belong also to $S_k(o_1)$, except for the ones of $B$ which are not in $V_k$. Furthermore, the sequence of goods visited by $p_{k-1}(o)$ before reaching the first good of $B$ remains the same in $p_k(o)$. The following lemma formalizes these observations.

**Lemma 7.** *For any step $k \geq 1$ of Algorithm 1 and any good $o \in V_k$, we have $S_{k-1}(o) \cap V_k \subseteq S_k(o)$, and for any good $f \in S_{k-1}(o) \cap V_k$, we have $[p_{k-1}(o)]_o^f = [p_k(o)]_o^f$.*

*Proof.* Note first that the outgoing edge of the last good visited by $p_{k-1}(o)$ is pointing to a good of $S_{k-1}(o)$. This outgoing edge creates a cycle in $p_{k-1}(o)$. Let $B$ be the subset of $S_{k-1}(o)$ visited by this cycle. By definition of Algorithm 1, we have $B \cap V_k = \emptyset$ and $S_{k-1}(o) \setminus B \subseteq V_k$.




(a) Directed path $p_{k-1}(o_1)$ in $G_{k-1}$

(b) Directed path $p_k(o_1)$ in $G_k$

Figure 8: Variations of $p_{k-1}(o_1)$ after one step of Algorithm 1

Let $\ell$ be the unique good of $S_{k-1}(o)\backslash B$ which precedes a good of $B$ in $p_{k-1}(o)$ (such good $\ell$ may not exist if $S_{k-1}(o) = B$). Let $o'$ be a good of $S_{k-1}(o)\backslash(B \cup \{\ell\})$ and $f'$ be the unique good of $S_{k-1}(o)\backslash B$ such that $(o', f') \in E_{k-1}$. Since $o'$ and $f'$ belong to $V_k$, we know by Lemma 6 that $(o', f') \in E_k$.

So we have proved that all the edges of subpath $[p_{k-1}(o)]_o^\ell$ belong to $G_k$. Therefore this directed path exists also in $G_k$, and all the goods of $S_{k-1}(o) \cap V_k$ are visited by this path. Furthermore, this path is a subpath of $p_k(o)$ since the outdegree of any good in $G_k$ equals 1. Hence, $[p_{k-1}(o)]_o^\ell = [p_k(o)]_o^\ell$ holds. This trivially implies that for any good $f$ belonging to $S_{k-1}(o) \cap V_k$, we have $[p_{k-1}(o)]_o^f = [p_k(o)]_o^f$. □

**Corollary 5.** *For any pair $(k, l)$ of steps of Algorithm 1 such that $k \geq l$, and for any good $o \in V_k$, we have $S_l(o) \cap V_k \subseteq S_k(o)$, and for any good $f \in S_l(o) \cap V_k$, we have $[p_l(o)]_o^f = [p_k(o)]_o^f$.*

The following lemma states that if $p_{k-1}(o)$ visits at least one good of subset $B \subseteq V_{k-1}$ during step $k-1$, and if both $o$ and all the goods of $B$ are part of $G_k$, then $p_k(o)$ should visit at least one good of $B$ during step $k$.

**Lemma 8.** *For any step $k \geq 1$ of Algorithm 1, for any good $o \in V_k$ and for any subset $B \subseteq V_k$, if $S_{k-1}(o) \cap B \neq \emptyset$ then $S_k(o) \cap B \neq \emptyset$.*

*Proof.* If $S_{k-1}(o) \cap B \neq \emptyset$ holds then there exists at least one good $o' \in S_{k-1}(o) \cap B$. $B \subseteq V_k$ implies that $o'$ belongs to $V_k$. Furthermore, Lemma 7 implies $S_{k-1}(o) \cap V_k \subseteq S_k(o)$. Therefore, $o' \in S_k(o)$ since $o' \in S_{k-1}(o) \cap V_k$. Hence, $o' \in S_k(o) \cap B$ and $S_k(o) \cap B \neq \emptyset$. □

**Corollary 6.** *For any step $k \geq 1$ of Algorithm 1, for any good $o \in V_k$ and for any subset $B \subseteq V_k$, if $S_k(o) \cap B = \emptyset$ then $S_{k-1}(o) \cap B = \emptyset$.*

## Appendix D. Relationship of ATTC Outcomes for Two Different Profiles and Omitted Proof for Section 7

In this appendix, we study the parallel evolution of the ATTC graphs when two different preference profiles $\boldsymbol{\tau}$ and $\boldsymbol{\tau'}$ are revealed by the same set of agents or identities $N$. Note first that the lemmas of Annexe C can be applied both when $\boldsymbol{\tau}$ or $\boldsymbol{\tau'}$ are revealed. Let $\mathcal{I}$ be the subset of agents or identities of $N$ who do not reveal the same LP-trees in $\boldsymbol{\tau}$ and $\boldsymbol{\tau'}$. We denote by $G'_k = (V'_k, E'_k)$ (resp. $G_k = (V_k, E_k)$) the ATTC graph during step $k$ of Algorithm 1 in the specific case where $\boldsymbol{\tau'}$

(resp. $\boldsymbol{\tau}$) is revealed, and let $Y_i^k$ (resp. $X_i^k$) denote the set of goods assigned to agent $i$ after the $k$ first steps in that case. Let $p'_k(o)$ (resp. $p_k(o)$) denote the directed path in $G'_k$ (resp. $G_k$) which starts from good $o$, and let $S'_k(o)$ (resp. $S_k(o)$) denote the set of goods visited by $p'_k(o)$ (resp. $p_k(o)$). Finally, for any good $o' \in S'_k(o)$ (resp. $o'' \in S_k(o)$), let $[p'_k]_o^{o'}$ (resp. $[p_k]_o^{o''}$) denote the subpath of $p'_k(o)$ (resp. $p_k(o)$) which starts from good $o$ and finishes at good $o'$ (resp. $o''$).

We start by focusing on the similarities in both cases. One could conjecture that if path $p_r(o)$ never visits one of the goods belonging to an agent of $\mathcal{I}$ for any step $r = 1, \ldots, k$ then path $p'_r(o)$ and path $p_r(o)$ should be the same (i.e., visits the same sequence of vertices) for any step $r = 1, \ldots, k$. This conjecture seems realistic since the owners of the goods belonging to these paths did not change their preferences. But in reality, this conjecture may be false if the owner of one of the goods visited by $p_k(o)$ prefers good $o'$ belonging to an agent of $\mathcal{I}$ to all remaining goods in $V_k$, and $o'$ does not belong to $V_k$ but belongs to $V'_k$. However, the following lemma shows that this conjecture is true under the condition that all goods belonging to the agents of $\mathcal{I}$ are still part of the ATTC graph.

**Lemma 9.** *For any step $k \geq 1$ of Algorithm 1 such that $\bigcup_{i\in\mathcal{I}} e_i \subseteq V_k$ (i.e., no good of an agent belonging to $\mathcal{I}$ has been assigned), and for any good $o \in V_k$, if $S_k(o) \cap \bigcup_{i\in\mathcal{I}} e_i = \emptyset$ (i.e., no good of an agent belonging to $\mathcal{I}$ has been visited by path $p_k(o)$) then $o \in V'_k$, $X^k_{\delta(o)} = Y^k_{\delta(o)}$ and $p'_k(o) = p_k(o)$ hold.*

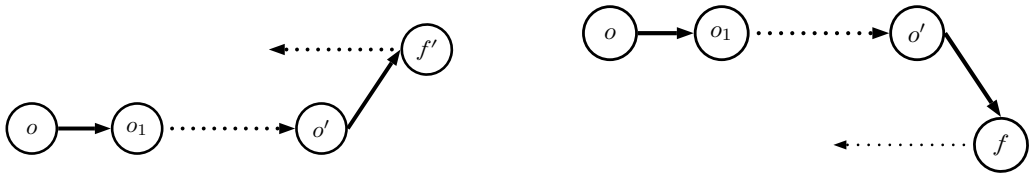
*Proof.* We prove this statement by induction on the number of steps $k$.

*Base case*: $V_1 = V'_1$ trivially holds since both sets equal $K$. Let $o$ be a good of $V_1$ such that $S_1(o) \cap \bigcup_{i\in\mathcal{I}} e_i = \emptyset$. For any good $o' \in S_1(o)$ and for any good $f'$ such that $(o', f') \in E_1$, we know by definition of Algorithm 1 that $f'$ labels the root of $\tau_{\delta(o')}$. On the other hand, $\tau_{\delta(o')}$ and $\tau'_{\delta(o')}$ are the same since $o' \notin \bigcup_{i\in\mathcal{I}} e_i$ implies $\delta(o') \notin \mathcal{I}$. Therefore, the root of $\tau'_{\delta(o')}$ is also labeled by $f'$, and by definition of Algorithm 1 we have $(o', f') \in E'_1$. The outdegree of any vertex of $V'_1$ being equal to 1, we obtain $p'_1(o) = p_1(o)$.

Let $f$ denote the good labeling the root of $\tau_{\delta(o)}$. By definition of Algorithm 1, edge $(o, f)$ belongs to $E_1$. This implies that $S_1(f)$ is a subset of $S_1(o)$, and therefore $S_1(f) \cap \bigcup_{i\in\mathcal{I}} e_i = \emptyset$ holds. As it was shown above, this implies that $p'_1(f) = p_1(f)$ holds. Furthermore, by definition of Algorithm 1 we know that for any good $o'' \in e_{\delta(o)}$, the unique outgoing edge of $o''$ in $V_1$ is $(o'', f)$. Hence, if a good belonging to $e_{\delta(o)}$ is part of a cycle in $G_1$ then this cycle should be $p_1(f)$. Therefore, this cycle should appear also in $G'_1$ under the form of $p'_1(p)$. On the other hand, if there is no cycle in $G_1$ including a good of $e_{\delta(o)}$ then $S_1(f) \cap e_{\delta(o)} = \emptyset$ should hold. Therefore, this implies with $p'_1(f) = p_1(f)$ that $S'_1(f) \cap e_{\delta(o)} = \emptyset$ holds and no good of $e_{\delta(o)}$ is visited by a cycle in $G'_1$. So we have proved that $X^1_{\delta(o)} = Y^1_{\delta(o)}$.

*Induction step*: Let $k$ be a step of Algorithm 1 such that $\bigcup_{i\in\mathcal{I}} e_i \subseteq V_k$, and let $o \in V_k$ such that $S_k(o) \cap \bigcup_{i\in\mathcal{I}} e_i = \emptyset$. Corollary 6 implies that $S_{k-1}(o) \cap \bigcup_{i\in\mathcal{I}} e_i = \emptyset$ holds. Therefore, by induction hypothesis we know that $o \in V'_{k-1}$, $X^{k-1}_{\delta(o)} = Y^{k-1}_{\delta(o)}$ and $p'_{k-1}(o) = p_{k-1}(o)$ hold. First, we show by contradiction that $o \in V'_k$ holds. If $o \notin V'_k$ then $p'_{k-1}(o)$ is the cycle visiting $o$ in $G'_{k-1}$. But this cycle appears also in $G_{k-1}$ under the form $p_{k-1}(o)$. Therefore $o$ should not belong to $V_k$, leading to a contradiction.

Now, we show by contradiction that $p'_k(o) = p_k(o)$ holds. Assume that $p'_k(o) \neq p_k(o)$ and let $f$ be the first good visited by $p'_k(o)$ which differs from $p_k(o)$, and let $f'$ be the good visited by $p_k(o)$ instead of $f$. Let $o'$ be the good preceding both $f$ in $p'_k(o)$ and $f'$ in $p_k(o)$, and let $j = \delta(o')$ denote




(a) $p_k(o)$ in $G_k$

(b) $p'_k(o)$ in $G'_k$

Figure 9: Differences between a path in $G_k$ an $G'_k$

the owner of $o'$. Figure 9 illustrates the notations. Note that we do not presume the presence or absence of $f'$ and $f$ in $V'_k$ and $V_k$, respectively. Lemma 2 implies that $a(\tau_j(X_j^{k-1}, X_j^{k-1} \cup V_k)) \subseteq K \setminus V_k$ holds. Therefore, $f$ cannot belong to both $V_k$ and $a(\tau_j(X_j^{k-1}, X_j^{k-1} \cup V_k))$, and *(i)* $f \notin a(\tau_j(X_j^{k-1}, X_j^{k-1} \cup V_k))$ or *(ii)* $f \notin V_k$ must hold.

Assume first that *(i)* $f \notin a(\tau_j(X_j^{k-1}, X_j^{k-1} \cup V_k))$ holds. In that case, we show that $f'$ cannot belong to $V'_k$. We know that $S_k(o') \subseteq S_k(o)$ holds since $o' \in S_k(o)$. This implies $S_k(o') \cap \bigcup_{i\in\mathcal{I}} e_i = \emptyset$ since $S_k(o) \cap \bigcup_{i\in\mathcal{I}} e_i = \emptyset$ holds by hypothesis. Therefore, Corollary 6 implies $S_{k-1}(o') \cap \bigcup_{i\in\mathcal{I}} e_i = \emptyset$, and we obtain by induction hypothesis $X_j^{k-1} = Y_j^{k-1}$. In other words, we have $X_j^{k-1} \oplus Y_j^{k-1} = K$. On the other hand, we know by Lemma 2 that $a(\tau_j(X_j^{k-1}, X_j^{k-1} \cup V_k)) \subseteq K \setminus V_k$ holds. Hence, for any good $o'' \in a(\tau_j(X_j^{k-1}, X_j^{k-1} \cup V_k))$ there is a step $l < k$ where $p_l(o'')$ forms a cycle that includes $o''$ in $G_l$. No good of $\bigcup_{i\in\mathcal{I}} e_i$ are visited by $p_l(o'')$ because otherwise this good would be missing in $V_k$, contradicting the assumption that $\bigcup_{i\in\mathcal{I}} e_i \subseteq V_k$. Therefore, $S_l(o'') \cap \bigcup_{i\in\mathcal{I}} e_i = \emptyset$ holds, implying by induction hypothesis that $p_l(o'') = p'_l(o'')$. In other words, the cycle containing $o''$ in $G_l$ appears also in $G'_l$. Therefore, $o'' \notin V'_k$ since $o'' \notin V'_{l+1}$ and $V'_k \subseteq V'_{l+1}$. All in all, this means that $a(\tau_j(X_j^{k-1}, X_j^{k-1} \cup V_k)) \subseteq K \setminus V'_k$ holds. As argued above, we also know that $a(\tau_j(X_j^{k-1}, X_j^{k-1} \cup V_k)) \subseteq K \setminus V_k$ and $X_j^{k-1} = Y_j^{k-1}$. Hence, $a(\tau_j(X_j^{k-1}, X_j^{k-1} \cup V_k)) \subseteq (X_j^{k-1} \cup V_k) \oplus (Y_j^{k-1} \cup V'_k) \subseteq K = X_j^{k-1} \oplus Y_j^{k-1}$ holds. Furthermore, $g(\tau_j(X_j^{k-1}, X_j^{k-1} \cup V_k))$ obviously belongs to $X_j^{k-1} \oplus Y_j^{k-1} = K$. Therefore, if $f' = g(\tau_j(X_j^{k-1}, X_j^{k-1} \cup V_k))$ belongs to $V'_k$ then $f' \in (X_j^{k-1} \cup V_k) \oplus (Y_j^{k-1} \cup V'_k)$ would hold since $f' \in V_k$. This would imply by Lemma 1 that $\tau_j(X_j^{k-1}, X_j^{k-1} \cup V_k) = \tau_j(Y_j^{k-1}, Y_j^{k-1} \cup V'_k)$, leading to a contradiction with $f' \neq f = g(\tau_j(Y_j^{k-1}, Y_j^{k-1} \cup V'_k))$. Therefore, we have proved that $f' \notin V'_k$ should hold.

Now, $f' \notin V'_k$ implies that there is a step $l < k$ where $p'_l(f')$ forms a cycle that includes $f'$ in $G'_l$. By contradiction, assume that $S_l(f') \cap \bigcup_{i\in\mathcal{I}} e_i = \emptyset$ holds. By induction hypothesis, $p'_l(f') = p_l(f')$ holds, and $f'$ is also contained in a cycle $p_l(f')$ in $G_l$. Therefore, $f' \notin V_{l+1}$ holds, leading to a contradiction since $V_k \subseteq V_{l+1}$. Hence, we have shown $S_l(f') \cap \bigcup_{i\in\mathcal{I}} e_i \neq \emptyset$. On the other hand, Corollary 5 implies $S_l(f') \cap V_k \subseteq S_k(f')$. Therefore, $S_k(f') \cap \bigcup_{i\in\mathcal{I}} e_i \neq \emptyset$ holds since $\bigcup_{i\in\mathcal{I}} e_i \subseteq V_k$. But this implies $S_k(o) \cap \bigcup_{i\in\mathcal{I}} e_i \neq \emptyset$ since $S_k(f') \subseteq S_k(o)$ holds, leading to a contradiction with the assumption that $S_k(o) \cap \bigcup_{i\in\mathcal{I}} e_i = \emptyset$. Therefore, we have proved that *(i)* $f \notin a(\tau_j(X_j^{k-1}, X_j^{k-1} \cup V_k))$ does not hold.

Assume now that *(ii)* $f \notin V_k$ holds. This means that there is a step $l < k$ where $p_l(f)$ forms a cycle that includes $f$ in $G_l$. By contradiction, assume that $S_l(f) \cap \bigcup_{i\in\mathcal{I}} e_i = \emptyset$ holds. By induction hypothesis, $p'_l(f) = p_l(f)$ holds, and $f$ is also contained in a cycle $p'_l(f)$ in $G'_l$. This implies $f \notin V_{l+1}$, leading to a contradiction with $f \in V'_k$ since $V'_k \subseteq V'_{l+1}$. Hence we have shown

$S_l(f) \cap \bigcup_{i \in \mathcal{I}} e_i \neq \emptyset$. On the other hand, this implies that at least one good $o'' \in \bigcup_{i \in \mathcal{I}} e_i$ is included in the cycle $p_l(f)$, and $o'' \notin V_{l+1}$. But this also leads to a contradiction with $\bigcup_{i \in \mathcal{I}} e_i \subseteq V_k$ since $V_k \subseteq V_{l+1}$. Therefore, we have proved that *(ii)* $f \notin V_k$ does not hold. To summarize, we have proved that both *(i)* and *(ii)* do not hold, leading to a contradiction. Therefore we have proved $p'_k(o) = p_k(o)$.

Finally, following the same logic as in the base case, it is easy to prove that the same good is assigned to $\delta(o)$ during step $k$ in both $G_k$ and $G'_k$. Furthermore by induction hypothesis we know that $X^{k-1}_{\delta(o)} = Y^{k-1}_{\delta(o)}$ holds. Therefore we have proved that $X^k_{\delta(o)} = Y^k_{\delta(o)}$.

□

The following lemma extends in a sense Lemma 9 to the first step of Algorithm 1 where path $p_k(o)$ reaches a good belonging to an agent of $\mathcal{I}$.

**Lemma 10.** *For any step $k \geq 1$ of Algorithm 1 such that $\bigcup_{i \in \mathcal{I}} e_i \subseteq V_k$, and for any good $o \in V_k$, if $k$ is the first step such that $S_k(o) \cap \bigcup_{i \in \mathcal{I}} e_i \neq \emptyset$, and if $f$ is the first good of $\bigcup_{i \in \mathcal{I}} e_i$ visited by $p_k(o)$ then $o \in V'_k$ and $[p'_k(o)]^f_o = [p_k(o)]^f_o$ hold.*

The proof of this lemma is similar to the induction proof of Lemma 9. Indeed, the proof of Lemma 10 can be obtained by replacing the edges and goods visited by $p'_k(o)$ and $p_k(o)$ by the edges and goods visited by $[p'_k(o)]^f_o$ and $[p_k(o)]^f_o$, respectively. Furthermore, the proof of this lemma does not need to be made by induction because the induction hypothesis used in the proof of Lemma 9 is replaced by the application of Lemma 9 to step $k-1$, where $p_{k-1}(o)$ does not reach a good of $\bigcup_{i \in \mathcal{I}} e_i$. Therefore, the proof of this lemma is omitted.

Finally, the end of this appendix is dedicated to the proof of Proposition 2. We assume here that $\boldsymbol{\tau}$ is the true preference profile and $\boldsymbol{\tau'}$ is the preference profile revealed when agent $i$ tries to manipulate. Agent $i$ is the only agent who changes her preference from $\boldsymbol{\tau}$ to $\boldsymbol{\tau'}$, and therefore $\mathcal{I} = \{i\}$. Let $o^*$ denote the first good acquired by agent $i$ when she reveals her true preference $\tau_i$. The following proposition is a reformulation of Proposition 2.

**Proposition 5.** *For any good $o \in \varphi_i(\boldsymbol{e}, \boldsymbol{\tau'})$, $\{o^*\} \succsim_{\tau_i} \{o\}$ holds.*

*Proof.* By contradiction, assume that there exists $o \in \varphi_i(\boldsymbol{e}, \boldsymbol{\tau'})$ such that $\{o\} \succ_{\tau_i} \{o^*\}$. By definition of $\tau_i$, this means that $g(\tau_i(\{o^*\}, \{o\})) = o$ and $o \neq o^*$. Let $k$ denote the step of Algorithm 1 where $o^*$ is assigned to agent $i$ when she reveals $\tau_i$. By definition of 1, this implies that $g(\tau_i(X^{k-1}_i, X^{k-1}_i \cup V_k)) = o^*$ and $S_k(o^*) \cap e_i \neq \emptyset$ hold. Furthermore, by definition of $o^*$ we know that no good of $K \setminus V_k$ is assigned to agent $i$ during the $k-1$ first steps. Therefore, $X^{k-1}_i = \emptyset$ holds. We show first that $o \notin V_k$. Assume by contradiction that $o \in V_k$. Lemma 2 implies that $a(\tau_i(X^{k-1}_i, X^{k-1}_i \cup V_k)) \subseteq K \setminus V_k$ holds. Furthermore, $o^* \in V_k$ and $o \in V_k$ implies $\{o\} \oplus X^{k-1}_i = K \setminus \{o\}$ and $\{o^*\} \oplus (X^{k-1}_i \cup V_k) = K \setminus (V_k \setminus \{o^*\})$. Therefore both $\{o\} \oplus X^{k-1}_i$ and $\{o^*\} \oplus (X^{k-1}_i \cup V_k)$ are supersets of $a(\tau_i(X^{k-1}_i, X^{k-1}_i \cup V_k)) \cup g(\tau_i(X^{k-1}_i, X^{k-1}_i \cup V_k))$, and Lemma 1 implies that $\tau_i(X^{k-1}_i, X^{k-1}_i \cup V_k) = \tau_i(\{o^*\}, \{o\})$ holds. But this leads to a contradiction with $o \neq o^*$ since $g(\tau_i(X^{k-1}_i, X^{k-1}_i \cup V_k)) = o^*$ and $g(\tau_i(\{o^*\}, \{o\})) = o$. Therefore, we have proved that $o \notin V_k$.

Now, $o \notin V_k$ means that there is a step $l < k$ where $p_l(o)$ forms a cycle including $o$ in $G_l$. But on the other hand, $e_i \subseteq V_k$ implies that no good belonging to agent $i$ should be contained in a cycle in $G_l$, and $S_l(o) \cap e_i = \emptyset$ holds. This implies by Lemma 9 that $o \in V'_l$ and $p'_l(o) = p_l(o)$

hold. Hence, good $o$ is assigned to the same agent in $G_l$ and $G'_l$, and this agent is not agent $i$ since $S_l(o) \cap e_i = \emptyset$. This leads to a contradiction with $o$ assigned to agent $i$ when she reveals $\tau'_i$.

□

## Appendix E. Omitted Proof for Subsection 8.2

In this appendix we study the impact of removing set $B$ of goods from the initial endowment in an assignment problem. The removal of these goods may be due to hiding manipulation or for the requirement of some proofs. Furthermore, the goods of $B$ may belong to the initial endowment of a single agent or multiples agents. The results provided in this appendix applies in all these cases.

We assume in this appendix that, from an assignment problem $(\boldsymbol{e}, \boldsymbol{\tau})$, we are able to construct a truncated assignment problem $(\boldsymbol{e'}, \boldsymbol{\tau})$ on the same set of agents or identities $N$ and with the same preference profile $\boldsymbol{\tau}$, but where the initial endowment $\boldsymbol{e'}$ results from the removal of the goods of $B$ from $\boldsymbol{e}$ . We denote by $G'_k = (V'_k, E'_k)$ (resp. $G_k = (V_k, E_k)$) the ATTC graph during step $k$ of Algorithm 1 applied to $(\boldsymbol{e'}, \boldsymbol{\tau})$ (resp. $(\boldsymbol{e}, \boldsymbol{\tau})$), and let $Y_i^k$ (resp. $X_i^k$) denote the set of goods assigned to agent $i$ after the $k$ first steps in that case. Let $p'_k(o)$ (resp. $p_k(o)$) denote the directed path in $G'_k$ (resp. $G_k$) which starts from good $o$, and let $S'_k(o)$ (resp. $S_k(o)$) denote the set of goods visited by $p'_k(o)$ (resp. $p_k(o)$).

Note that the definition of $\boldsymbol{e'}$ may lead to an empty endowment for some agents of $N$. However, in that case these agents are ignored during Algorithm 1 applied to $(\boldsymbol{e'}, \boldsymbol{\tau})$, and no good is assigned to them. Note also that the LP-trees of $\boldsymbol{\tau}$ contain vertices labeled by the goods of $B$. Therefore, in order to compare sets of goods through these LP-trees, we initially assign the goods of $B$ to the agents who receive them during Algorithm 1 applied to $(\boldsymbol{e}, \boldsymbol{\tau})$. More formally, we assume that $Y_i^0$, which is the set of goods assigned to agent $i$ before the first step of Algorithm 1 applied to $(\boldsymbol{e'}, \boldsymbol{\tau})$, contains the subset of goods of $B$ assigned to agent $i$ during Algorithm 1 applied to $(\boldsymbol{e}, \boldsymbol{\tau})$ i.e., we have $Y_i^0 = \varphi_i(\boldsymbol{e}, \boldsymbol{\tau}) \cap B$ for any agent $i \in N$. Hence, by definition of Algorithm 1 we have $Y_i^0 = \varphi_i(\boldsymbol{e}, \boldsymbol{\tau}) \cap B \subseteq Y_i^k$ for any step $k \geq 1$ and for any agent $i \in N$. However, the goods of $Y_i^0$ are not really assigned to agent $i$ during Algorithm 1 because no agent reveals these goods as part of her initial endowment in $\boldsymbol{e'}$.

The following lemma is similar to Lemma 9, but applied to the context of changes in the initial endowment rather than changes in the revealed preference profile.

**Lemma 11.** *For any step $k \geq 1$ of Algorithm 1 such that $B \subseteq V_k$, and for any good $o \in V_k$, if $S_k(o) \cap B = \emptyset$ then $o \in V'_k$, $X^k_{\delta(o)} = Y^k_{\delta(o)} \setminus B$ and $p_k(o) = p'_k(o)$ hold.*

The proof of this Lemma is essentially the same as the proof of Lemma 9. Therefore, we chose to omit the proof of this lemma.

The following lemma shows that if we remove the goods visited by a cycle appearing during Algorithm 1 then the sequence of cycles occurring during prior steps is not affected by this removal.

**Lemma 12.** *If $B$ corresponds to the set of goods visited by a cycle $\mathcal{C}$ occuring in $G_k$ then any cycle $\mathcal{C}' \neq \mathcal{C}$, occuring in $G_l$ during step $l \leq k$, appears also in $G'_l$. Furthermore, for any good $o$ visited by $\mathcal{C}'$, $\tau_{\delta(o)}(Y^{l-1}_{\delta(o)}, Y^{l-1}_{\delta(o)} \cup V'_l) = \tau_{\delta(o)}(X^{l-1}_{\delta(o)}, X^{l-1}_{\delta(o)} \cup V_l)$ holds.*

*Proof.* Let $o$ be a good visited by cycle $\mathcal{C}' \neq \mathcal{C}$ during step $l \leq k$. We know that $\mathcal{C}$ occurs during step $k$ and visits all the goods of $B$. Therefore, $B \subseteq V_l$ since $V_k \subseteq V_l$. Furthermore, by definition

of Algorithm 1 no good of $B$ can be visited by $\mathcal{C}' = p_l(o)$ since $\mathcal{C}' \neq \mathcal{C}$. Therefore, $S_l(o) \cap B = \emptyset$ holds. This implies by Lemma 11 that $p_l(o) = p'_l(o)$. Therefore, cycle $C'$ occurs in $G'_l$ under the form $p'_l(o)$, and we have established the first part of the lemma.

As argued above, the sequences of cycles occurring in Algorithm 1 applied to both $(\boldsymbol{e}, \boldsymbol{\tau})$ and $(\boldsymbol{e'}, \boldsymbol{\tau})$ are the same during the $k$ first steps (except for $\mathcal{C}$ during step $k$). Therefore, for any step $l \leq k$, $V_l \setminus B = V'_l$ holds since $V'_1 = V_1 \setminus B$. Let $o$ be a good visited by cycle $\mathcal{C}' \neq \mathcal{C}$ during step $l \leq k$. We know by Lemma 11 that $Y^{l-1}_{\delta(o)} \setminus B = X^{l-1}_{\delta(o)}$ holds since $S_l(o) \cap B = \emptyset$ (as shown above). Hence, $K \setminus B \subseteq Y^{l-1}_{\delta(o)} \oplus X^{l-1}_{\delta(o)}$ and $K \setminus B \subseteq (Y^{l-1}_{\delta(o)} \cup V'_l) \oplus (X^{l-1}_{\delta(o)} \cup V_l)$ trivially hold. Moreover, Lemma 2 implies $a(\tau_{\delta(o)}(X^{l-1}_{\delta(o)}, X^{l-1}_{\delta(o)} \cup V_l)) \subseteq K \setminus V_l \subseteq K \setminus B$, where the last inclusion is due to $B \subseteq V_k \subseteq V_l$. Therefore, both $Y^{l-1}_{\delta(o)} \oplus X^{l-1}_{\delta(o)}$ and $(Y^{l-1}_{\delta(o)} \cup V'_l) \oplus (X^{l-1}_{\delta(o)} \cup V_l)$ are supersets of $a(\tau_{\delta(o)}(X^{l-1}_{\delta(o)}, X^{l-1}_{\delta(o)} \cup V_l))$. Finally, we know that $g(\tau_{\delta(o)}(X^{l-1}_{\delta(o)}, X^{l-1}_{\delta(o)} \cup V_l)) \notin B$ holds since no good of $B$ is visited by $\mathcal{C}'$ (as mentioned above). Therefore, $g(\tau_{\delta(o)}(X^{l-1}_{\delta(o)}, X^{l-1}_{\delta(o)} \cup V_l))$ belongs to both $Y^{l-1}_{\delta(o)} \oplus X^{l-1}_{\delta(o)}$ and $(Y^{l-1}_{\delta(o)} \cup V'_l) \oplus (X^{l-1}_{\delta(o)} \cup V_l)$. Hence, by Lemma 1 we can assert that $\tau_{\delta(o)}(Y^{l-1}_{\delta(o)}, Y^{l-1}_{\delta(o)} \cup V'_l) = \tau_{\delta(o)}(X^{l-1}_{\delta(o)}, X^{l-1}_{\delta(o)} \cup V_l)$. □

Lemma 12 can be applied to remove the goods visited by a cycle occurring during the last step of Algorithm 1 without changing the outcome. We will use Lemma 12 to show that we can also remove the goods visited by a cycle occurring during an arbitrary step of Algorithm 1 without changing the outcome. The sketch of proof for a given cycle $\mathcal{C}$ occurring during step $k$ is as follow:

- Remove one by one the cycles occurring during the last step of Algorithm 1 until $k$ becomes the last step of Algorithm 1 applied to the truncated problem $(\tilde{\boldsymbol{e}}, \boldsymbol{\tau})$, where $\tilde{\boldsymbol{e}}$ is the initial endowment resulting after removal,
- Remove from $(\tilde{\boldsymbol{e}}, \boldsymbol{\tau})$ the goods visited by $\mathcal{C}$ to obtain assignment problem $(\boldsymbol{e''}, \boldsymbol{\tau})$,
- Put back one by one the goods of the cycles removed in the first bullet until all goods, except for those removed in the second bullet, are back.

We will show that all these transformations preserve the outcome of Algorithm 1 for the remaining goods. Lemma 12 can be used to show that the outcome of Algorithm 1 is preserved by the transformations of the first and second bullets. But in the third bullet, we insert goods instead of removing them. Therefore, we need to show that, under some conditions, we can add some goods in an assignment problem without changing the outcome of Algorithm 1 for the goods which were already part of the problem.

To be as close as possible to the notations used in the sketch of proof, we assume here that the assignment problem $(\boldsymbol{e'}, \boldsymbol{\tau})$ results from the assignment problem $(\boldsymbol{e''}, \boldsymbol{\tau})$ by adding the goods of a given set $D$ i.e., each good $o$ of $D$ is reassigned to the initial endowment $e'_{\delta(o)}$ of its owner $\delta(o)$ in the original problem $(\boldsymbol{e}, \boldsymbol{\tau})$ (where $o \in e_{\delta(o)}$). We denote by $G''_k = (V''_k, E''_k)$ the ATTC graph during step $k$ of Algorithm 1 applied to $(\boldsymbol{e''}, \boldsymbol{\tau})$, and let $Z^k_i$ denote the set of goods assigned to agent $i$ after the $k$ first steps in that case. Let $p''_k(o)$ denote the directed path in $G''_k$ which starts from good $o$, and let $S''_k(o)$ denote the set of goods visited by $p''_k(o)$.

Note that during the transformations of the first and second bullets, removed goods were assigned to agents before the beginning of the ATTC procedure applied to $(\boldsymbol{e''}, \boldsymbol{\tau})$. We assume that we keep this initial assignment in $(\boldsymbol{e'}, \boldsymbol{\tau})$ except for the goods of $D$ which are reinjected into the

problem. More formally, we assume that for any agent $i \in N$, $Y_i^0 = Z_i^0 \setminus D$, where $Y_i^0$ are the goods assigned before the first step of Algorithm 1 applied to $(\boldsymbol{e'}, \boldsymbol{\tau})$. The following lemma shows that if for any agent the goods of $D$ are less worthy than the goods assigned to them by Algorithm 1 applied to $(\boldsymbol{e''}, \boldsymbol{\tau})$ then the outcome of Algorithm 1 is exactly the same for the goods in common to $\boldsymbol{e}$ and $\boldsymbol{e'}$.

**Lemma 13.** *If for any step $k \geq 1$ of Algorithm 1 applied to $(\boldsymbol{e''}, \boldsymbol{\tau})$, and for any agent $i$ such that $e_i'' \cap V_k'' \neq \emptyset$, $a(\tau_i(Z_i^{k-1}, Z_i^{k-1} \cup V_k'')) \subseteq K \setminus D$ holds then $V_l'' = V_l' \setminus D$ and $E_l'' \subseteq E_l'$ holds for any step $l$. Furthermore, with the same conditions, for any cycle $\mathcal{C}'$ in $G_l''$ and for any good $o$ visited by $\mathcal{C}'$, $\mathcal{C}'$ occurs also in $G_l'$ and $\tau_{\delta(o)}(Z_{\delta(o)}^{l-1}, Z_{\delta(o)}^{l-1} \cup V_l'') = \tau_{\delta(o)}(Y_{\delta(o)}^{l-1}, Y_{\delta(o)}^{l-1} \cup V_l')$.*

*Proof.* By contradiction let $l$ be the first step of Algorithm 1 such that $V_l'' \neq V_l' \setminus D$ or $E_l'' \not\subseteq E_l'$ hold. This means that during any ulterior step $r < l$, $V_r'' = V_r' \setminus D$ and $E_r'' \subseteq E_r'$ hold. This implies that for any good $o \in V_r''$, $p_r''(o) = p_r'(o)$ because $S_r''(o) \subseteq V_r'$ holds and every outgoing edges of the goods of $S_r''(o)$ are the same in $G_l'$ and $G_l''$. Therefore, $V_l'' = V_l' \setminus D$ has to hold, and this implies by assumption that $E_l'' \not\subseteq E_l'$ holds. Furthermore, any good of $V_r'' \setminus V_{r+1}''$, assigned during step $r < l$ of the Algorithm 1 procedure applied to $(\boldsymbol{e''}, \boldsymbol{\tau})$, should be assigned to the same agent during step $r$ of Algorithm 1 applied to $(\boldsymbol{e'}, \boldsymbol{\tau})$. Therefore, $Y_i^{l-1} \setminus D = Z_i^{l-1} \setminus D$ holds for any agent $i \in N$.

Let $o$ be a good of $V_l''$ such that the outgoing edge from $o$ in $E_l''$ does not belong to $E_l'$, and let $i = \delta(o)$ be the owner of $o$. By definition of Algorithm 1, this means that $\tau_i(Z_i^{l-1}, Z_i^{l-1} \cup V_l'') \neq \tau_i(Y_i^{l-1}, Y_i^{l-1} \cup V_l')$. We know that $g(\tau_i(Z_i^{l-1}, Z_i^{l-1} \cup V_l'')) \in V_l'' \subseteq K \setminus D$. By assumption we also know that $a(\tau_i(Z_i^{l-1}, Z_i^{l-1} \cup V_l'')) \subseteq K \setminus D$. Furthermore, $Y_i^{l-1} \setminus D = Z_i^{l-1} \setminus D$ and $V_l'' = V_l' \setminus D$ implies that $K \setminus D \subseteq Y_i^{l-1} \oplus Z_i^{l-1}$ and $K \setminus D \subseteq (Z_i^{l-1} \cup V_l'') \oplus (Y_i^{l-1} \cup V_l')$ hold. All in all, we have $g(\tau_i(Z_i^{l-1}, Z_i^{l-1} \cup V_l'')) \cup a(\tau_i(Z_i^{l-1}, Z_i^{l-1} \cup V_l''))$ subset of both $Z_i^{l-1} \oplus Y_i^{l-1}$ and $(Z_i^{l-1} \cup V_l'') \oplus (Y_i^{l-1} \cup V_l')$. This implies by Lemma 2 that $\tau_i(Z_i^{l-1}, Z_i^{l-1} \cup V_l'') = \tau_i(Y_i^{l-1}, Y_i^{l-1} \cup V_l')$, leading to a contradiction. Hence, both $V_l'' = V_l \setminus D$ and $E_l'' \subseteq E_l'$ hold and the first part of Lemma 13 is established.

Let $\mathcal{C}'$ be a cycle occurring in $G_l''$ and let $o$ be a good visited by $\mathcal{C}'$. According to the first part of Lemma 13, we know that $V_l'' \subseteq V_l'$ and $E_l'' \subseteq E_l'$. Therefore, the outgoing edge of any good in $S_l''(o) \subseteq V_l''$ is the same in $G_l'$ and $G_l''$, and $\mathcal{C}' = p_l''(o) = p_l'(o)$ holds. In other words, cycle $\mathcal{C}'$ occurs also in $G_l'$ under the form $p_k'(o)$. Furthermore, by following the same reasoning as in the previous paragraph, it is easy to show that $\tau_i(Z_i^{l-1}, Z_i^{l-1} \cup V_l'') = \tau_i(Y_i^{l-1}, Y_i^{l-1} \cup V_l')$ holds. □

We are now ready to prove that the removal of $B$, the set of goods visited by a cycle $\mathcal{C}$, does not change the outcome of Algorithm 1 for the remaining goods. In the following lemma, we assume that $(\boldsymbol{e}, \boldsymbol{\tau})$ is the initial problem and $(\boldsymbol{e'}, \boldsymbol{\tau})$ is the problem resulting from the removal of $B$ from $(\boldsymbol{e}, \boldsymbol{\tau})$. We assume also that $\mathcal{C}$ occurs during step $k$ of the Algorithm 1 procedure applied to $(\boldsymbol{e}, \boldsymbol{\tau})$.

**Lemma 14.** *If $\mathcal{C}' \neq \mathcal{C}$ is a cycle occurring in $G_l$ during step $l$ of Algorithm 1 then $\mathcal{C}'$ occurs also in $G_r'$ during a step $r \leq l$. Furthermore, if $\mathcal{C}'' \neq C$ is another cycle appearing in $G_s$ during a step $s < l$, and both $\mathcal{C}'$ and $\mathcal{C}''$ visit goods belonging to the initial endowment of a given agent then $\mathcal{C}''$ occurs in $G_t'$ during a step $t < r$.*

The first part of the Lemma 14 states that a cycle occurring in Algorithm 1 applied to $(\boldsymbol{e}, \boldsymbol{\tau})$ occurs also in the Algorithm 1 applied to $(\boldsymbol{e'}, \boldsymbol{\tau})$, and this cycle does not occur later during Algorithm 1 for $(\boldsymbol{e'}, \boldsymbol{\tau})$ than $(\boldsymbol{e}, \boldsymbol{\tau})$. The second part of Lemma 14 states that the sequence of cycles

involving the goods of an agent is the same for Algorithm 1 applied to $(\boldsymbol{e}, \boldsymbol{\tau})$ or $(\boldsymbol{e'}, \boldsymbol{\tau})$, even if cycles may occurs earlier when Algorithm 1 is applied to the truncated problem $(\boldsymbol{e'}, \boldsymbol{\tau})$. We follow the sketch of proof presented earlier and its notations.

*Proof.* In addition to Lemma 14, we show that for any cycle $\mathcal{C}' \neq \mathcal{C}$ occurring in $G_l$ and occurring in $G'_r$ during step $r \leq l$ (as in the statement of Lemma 14) and for any good $o$ visited by $\mathcal{C}'$, $\tau_{\delta(o)}(Y^{r-1}_{\delta(o)}, Y^{r-1}_{\delta(o)} \cup V'_r) = \tau_{\delta(o)}(X^{l-1}_{\delta(o)}, X^{l-1}_{\delta(o)} \cup V_l)$ holds. Let $h$ denotes the last step of Algorithm 1 applied to $(\boldsymbol{e}, \boldsymbol{\tau})$. The proof is by induction on $h - k$.

*Base case*: When $h = k$ i.e., $k$ is the last step of Algorithm 1, Lemma 12 states that we can remove the goods of $B$ without changing the behavior of Algorithm 1 for the remaining goods. Hence, any cycle $\mathcal{C}' \neq \mathcal{C}$ occurring in $G_l$ occurs also in $G'_l$. Furthermore, for the same reason if $\mathcal{C}'' \neq \mathcal{C}$ is another cycle appearing in $G_s$ during a step $s < l$ then $\mathcal{C}''$ appears in $G'_s$, and $s < l$. Finally, Lemma 12 states that for any good $o$ visited by $\mathcal{C}'$, $\tau_{\delta(o)}(Y^{l-1}_{\delta(o)}, Y^{l-1}_{\delta(o)} \cup V'_l) = \tau_{\delta(o)}(X^{l-1}_{\delta(o)}, X^{l-1}_{\delta(o)} \cup V_l)$ holds.

*Induction step*: Assume now that $h > k$. Let $(\tilde{\boldsymbol{e}}, \boldsymbol{\tau})$ be the assignment problem obtained by removing the goods of $V_h$ i.e., the goods assigned during the last step of Algorithm 1 applied to $(\boldsymbol{e}, \boldsymbol{\tau})$. We denote by $\tilde{G}_l = (\tilde{V}_l, \tilde{E}_l)$ the ATTC graph during step $l$ of the ATTC procedure applied to $(\tilde{\boldsymbol{e}}, \boldsymbol{\tau})$, and $\tilde{Z}^i_l$ the set of goods assigned to agent $i$ at the end of this step. For any agent $i \in N$, the set of goods $\tilde{Z}^i_0$ is the set of goods assigned to agent $i$ before the beginning of Algorithm 1 applied to $(\tilde{\boldsymbol{e}}, \boldsymbol{\tau})$, and it contains the good of $V_h$ assigned to agent $i$ during the last step of Algorithm 1 applied to $(\boldsymbol{e}, \boldsymbol{\tau})$, if such a good exists. Note that $\tilde{Z}^i_0$ contains at most one good since at most one good can be assigned to an agent during one step of Algorithm 1.

By assuming that the goods of $V_h$ were removed cycle by cycle, we can state by Lemma 12 that any cycle $\mathcal{C}'$ occurring in $G_l$ during step $l < h$ occurs also in $\tilde{G}_l$, and for any good $o$ visited by $\mathcal{C}'$, we have $\tau_{\delta(o)}(\tilde{Z}^{l-1}_{\delta(o)}, \tilde{Z}^{l-1}_{\delta(o)} \cup \tilde{V}_l) = \tau_{\delta(o)}(X^{l-1}_{\delta(o)}, X^{l-1}_{\delta(o)} \cup V_l)$. Hence, Algorithm 1 applied to $(\tilde{\boldsymbol{e}}, \boldsymbol{\tau})$ has exactly $h - 1$ steps. Now let $(\boldsymbol{e''}, \boldsymbol{\tau})$ be the assignment problem obtained by removing the goods of $B$ to the assignment problem $(\tilde{\boldsymbol{e}}, \boldsymbol{\tau})$. The notations used to describe the different steps of Algorithm 1 applied to $(\boldsymbol{e''}, \boldsymbol{\tau})$ are the same than the ones described before Lemma 13. Note that for any agent $i \in N$, the set of goods $Z^i_0$ contains the good of $\tilde{Z}^i_0$ and the goods of $B$ assigned to agent $i$ during Algorithm 1 applied to $(\tilde{\boldsymbol{e}}, \boldsymbol{\tau})$, if such good exists. By induction hypothesis we know that any cycle $\mathcal{C}' \neq \mathcal{C}$, occurring in $\tilde{G}_l$ (and equivalently in $G_l$) occurs also in $G''_r$ during a step $r \leq l$. Furthermore, if $\mathcal{C}'' \neq \mathcal{C}$ is another cycle appearing in $\tilde{G}_s$ during a step $s < l$ (and equivalently in $G_s$), and both $\mathcal{C}'$ and $\mathcal{C}''$ visit goods belonging to the initial endowment of a given agent, then $\mathcal{C}''$ appears in $G''_t$ during a step $t < r$. Finally, for any good $o$ visited by $\mathcal{C}'$, $\tau_{\delta(o)}(Z^{r-1}_{\delta(o)}, Z^{r-1}_{\delta(o)} \cup V''_r) = \tau_{\delta(o)}(\tilde{Z}^{l-1}_{\delta(o)}, \tilde{Z}^{l-1}_{\delta(o)} \cup \tilde{V}_l)$ holds. Note that this implies $\tau_{\delta(o)}(Z^{r-1}_{\delta(o)}, Z^{r-1}_{\delta(o)} \cup V''_r) = \tau_{\delta(o)}(X^{l-1}_{\delta(o)}, X^{l-1}_{\delta(o)} \cup V_l)$ because we have shown that $\tau_{\delta(o)}(\tilde{Z}^{l-1}_{\delta(o)}, \tilde{Z}^{l-1}_{\delta(o)} \cup \tilde{V}_l) = \tau_{\delta(o)}(X^{l-1}_{\delta(o)}, X^{l-1}_{\delta(o)} \cup V_l)$ holds.

We show now that for any step $t$ of Algorithm 1 applied to $(\boldsymbol{e''}, \boldsymbol{\tau})$ and for any agent $i$ such that $e''_i \cap V''_t \neq \emptyset$, $a(\tau_i(Z^{t-1}_i, Z^{t-1}_i \cup V''_t)) \subseteq K \setminus V_h$ holds. By contradiction let $t$ be the first step where there is an agent $i$ with $e''_i \cap V''_t \neq \emptyset$ and $a(\tau_i(Z^{t-1}_i, Z^{t-1}_i \cup V''_t)) \cap V_h \neq \emptyset$. Let $r \geq t$ be the earliest step of Algorithm 1 applied to $(\boldsymbol{e''}, \boldsymbol{\tau})$ such that a good $o$ belonging to $e''_i$ (i.e., $\delta(o) = i$) is visited by a cycle $\mathcal{C}'$ in $G''_r$. By Corollary 4 we know that $a(\tau_i(Z^{t-1}_i, Z^{t-1}_i \cup V''_t)) \subseteq a(\tau_i(Z^{r-1}_i, Z^{r-1}_i \cup V''_r))$ holds since $r \geq t$. Furthermore, as stated in the previous paragraph

we know that there exists a step $l \geq r$ such that $\mathcal{C}'$ occurs in $\tilde{G}_l$ (and equivalently in $G_l$), and $\tau_i(Z_i^{r-1}, Z_i^{r-1} \cup V_r'') = \tau_i(X_i^{l-1}, X_i^{l-1} \cup V_l)$. By definition of $\tau_i$, this implies that $a(\tau_i(Z_i^{r-1}, Z_i^{r-1} \cup V_r'')) = a(\tau_i(X_i^{l-1}, X_i^{l-1} \cup V_l))$, and therefore $a(\tau_i(Z_i^{t-1}, Z_i^{t-1} \cup V_t'')) \subseteq a(\tau_i(X_i^{l-1}, X_i^{l-1} \cup V_l))$ holds. Hence, we can conclude that $a(\tau_i(X_i^{l-1}, X_i^{l-1} \cup V_l)) \cap V_h \neq \emptyset$ holds. But on the other hand, Lemma 2 implies $a(\tau_i(X_i^{l-1}, X_i^{l-1} \cup V_l)) \subseteq K \setminus V_l$. Furthermore, $K \setminus V_l \subseteq K \setminus V_h$ holds since $h > l$. This leads to contradiction with $a(\tau_i(X_i^{l-1}, X_i^{l-1} \cup V_l)) \cap V_h \neq \emptyset$.

Note first that assignment problem $(\boldsymbol{e'}, \boldsymbol{\tau})$ can be obtained from $(\boldsymbol{e''}, \boldsymbol{\tau})$ by reinserting the goods of $V_h$ into the initial endowment of their owners (instead of assigning these goods to their owner before starting Algorithm 1). Notations used to describe Algorithm 1 applied to $(\boldsymbol{e'}, \boldsymbol{\tau})$ are the same than the ones used at the beginning of this appendix. The property shown in the previous paragraph implies that we can apply Lemma 13. Hence, $V_r'' = V_r' \setminus V_h$ and $E_r'' \subseteq E_r'$ hold for any step $r$, any cycle $\mathcal{C}'$ occurring in $G_r''$ will occur also in $G_r'$, and for any good $o$ visited by $\mathcal{C}'$, $\tau_{\delta(o)}(Z_{\delta(o)}^{r-1}, Z_{\delta(o)}^{r-1} \cup V_r'') = \tau_{\delta(o)}(Y_{\delta(o)}^{r-1}, Y_{\delta(o)}^{r-1} \cup V_r')$ holds. Furthermore, we we have shown earlier that $\mathcal{C}'$ also occurs in $\tilde{G}_l$ (and equivalently in $G_l$) during a step $l \geq r$, and $\tau_{\delta(o)}(Z_{\delta(o)}^{r-1}, Z_{\delta(o)}^{r-1} \cup V_r'') = \tau_{\delta(o)}(X_{\delta(o)}^{l-1}, X_{\delta(o)}^{l-1} \cup V_l)$. To summarize, for any cycle $\mathcal{C}' \neq \mathcal{C}$ occuring in $G_l$ during a step $l < h$, $\mathcal{C}'$ occurs in $G_r'$ during a step $r \leq l$. Furthermore, if $\mathcal{C}'' \neq \mathcal{C}$ is another cycle appearing in $G_s$ during a step $s < l$ and both $\mathcal{C}'$ and $\mathcal{C}''$ visit goods belonging to the initial endowment of a given agent then $\mathcal{C}''$ occurs in $G_t'$ during a step $t < r$ since $\mathcal{C}''$ occurs in $G_t''$ (and therefore in $G_t'$).

It remains to consider the case of goods belonging to $V_h$. Let $o$ be a good belonging to $V_h$ and let $i = \delta(o)$ be the owner of $o$. Let $l$ be the step of Algorithm 1 applied to $(\boldsymbol{e'}, \boldsymbol{\tau})$ where $o$ is visited by a cycle $\mathcal{C}' = p_l'(o)$. It is obvious that $S_l'(o) \subseteq V_h$ since we have already described all cycles containing the goods of $V \setminus V_h$ when Algorithm 1 is applied to $(\boldsymbol{e'}, \boldsymbol{\tau})$ and all of them appear also during Algorithm 1 applied to $(\boldsymbol{e''}, \boldsymbol{\tau})$ where no good of $V_h$ are present. This implies by definition of Algorithm 1 that $g(\tau_i(Y_i^{l-1}, Y_i^{l-1} \cup V_r')) \in V_h$ holds. We claim that $e_i' \cap V_l' = \{o\}$ holds i.e., $o$ is the last remaining good of agent $i$ in $G_l'$.Assume by contradiction that $e_i' \cap V_l'$ contains another good $o' \neq o$. Note that $o'$ cannot belong to $V_h$ because no more than one good can be assigned to agent $i$ during step $h$ of $e_i' \cap V_l' = \{o\}$ applied to $(\boldsymbol{e}, \boldsymbol{\tau})$. Moreover, it was shown in the previous paragraph that $V_l' \setminus V_h = V_l''$ and $E_l'' \subseteq E_l'$ hold. Hence, $o' \in V_l' \setminus V_h$ implies that $o' \in V_l''$ holds. Furthermore, the outgoing edge of $o'$ in $G_l''$ must be the same as the outgoing edge of $o'$ in $G_l'$. Let $(o', f')$ be this outgoing edge. By definition of Algorithm 1, if $(o', f')$ belongs to $E_l'$ then $f' = g(\tau_i(Y_i^{l-1}, Y_i^{l-1} \cup V_l'))$. On the other hand, $f'$ does not belong to $V_h$ because $(o', f') \in E_l''$ implies $f' \in V_l'' = V_l' \setminus V_h$. This leads to a contradiction with $g(\tau_i(Y_i^{l-1}, Y_i^{l-1} \cup V_l')) \in V_h$. Therefore, we have shown that $e_i' \cap V_l' = \{o\}$ holds. This implies that $\mathcal{C}'$ must be the last cycle of Algorithm 1 applied to $(\boldsymbol{e'}, \boldsymbol{\tau})$ which contains a good belonging to agent $i$.

The latest result means that cycles including goods of $e_i' \setminus \{o\}$ must occur during earlier steps than $l$ in Algorithm 1 applied to $(\boldsymbol{e'}, \boldsymbol{\tau})$. Furthermore, $e_i' \setminus \{o\} \subseteq K \setminus V_h$ implies that these cycles must also occur during Algorithm 1 applied to $(\boldsymbol{e}, \boldsymbol{\tau})$, and during earlier steps than $l$. Finally, any good of $B$ that is assigned to agent $i$ during Algorithm 1 applied to $(\boldsymbol{e}, \boldsymbol{\tau})$ would be part of $Y_i^0$. Hence, $Y_i^{l-1} = X_i^{h-1}$ holds i.e., the same set of goods is assigned to agent $i$ at step $l$ of Algorithm 1 applied to $(\boldsymbol{e}, \boldsymbol{\tau})$ then at step $h-1$ of Algorithm 1 applied to $(\boldsymbol{e'}, \boldsymbol{\tau})$. This implies that $Y_i^{l-1} \oplus X_i^{h-1} = K$ and $K \setminus ((V_h \setminus V_l') \cup (V_l' \setminus V_h)) \subseteq (Y_i^{l-1} \cup V_l') \oplus (X_i^{h-1} \cup V_h)$ hold. We show now by contradiction that $a(\tau_i(X_i^{h-1}, X_i^{h-1} \cup V_h)) \cap (V_l' \setminus V_h) = \emptyset$. For the sake of contradiction, assume that $a(\tau_i(X_i^{h-1}, X_i^{h-1} \cup V_h)) \cap (V_l' \setminus V_h) \neq \emptyset$. This implies by Lemma 5 that $g(\tau_i(X_i^{h-1}, X_i^{h-1} \cup V_h \cup V_l')) \in V_l' \setminus V_h$ since $V_l' \subseteq K \setminus Y_i^{l-1} = K \setminus X_i^{h-1}$. But on the other

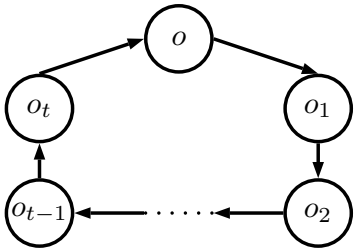



Figure 10: Cycle $\mathcal{C}'$ in $G_h$

hand, we know that $K \setminus (V_h \setminus V_l') \subseteq (X_i^{h-1} \cup V_h \cup V_l') \oplus (Y_i^{l-1} \cup V_l')$ and $a(\tau_i(X_i^{h-1}, X_i^{h-1} \cup V_h \cup V_l')) \subseteq K \setminus (V_h \cup V_l')$ hold (proof similar to Lemma 2). All in all, $g(\tau_i(X_i^{h-1}, X_i^{h-1} \cup V_h \cup V_l')) \cup a(\tau_i(X_i^{h-1}, X_i^{h-1} \cup V_h \cup V_l')) \subseteq (X_i^{h-1} \cup V_h \cup V_l') \oplus (Y_i^{l-1} \cup V_l')$ holds. This implies by Lemma 1 that $\tau_i(Y_i^{l-1}, Y_i^{l-1} \cup V_l') = \tau_i(X_i^{h-1}, X_i^{h-1} \cup V_h \cup V_l')$. But this also implies that $g(\tau_i(Y_i^{l-1}, Y_i^{l-1} \cup V_l')) = g(\tau_i(X_i^{h-1}, X_i^{h-1} \cup V_h \cup V_l'))$ belongs to $V_l' \setminus V_h$, which contradicts the fact that $g(\tau_i(Y_i^{l-1}, Y_i^{l-1} \cup V_l')) \in V_h$. Hence, we have shown that $a(\tau_i(X_i^{h-1}, X_i^{h-1} \cup V_h)) \cap (V_l' \setminus V_h) = \emptyset$

A direct consequence of Lemma 2 is that $a(\tau_i(X_i^{h-1}, X_i^{h-1} \cup V_h)) \subseteq K \setminus V_h$ holds. This implies that $a(\tau_i(X_i^{h-1}, X_i^{h-1} \cup V_h)) \subseteq (Y_i^{l-1} \cup V_l') \oplus (X_i^{h-1} \cup V_h)$ holds since we have shown that both $a(\tau_i(X_i^{h-1}, X_i^{h-1} \cup V_h)) \subseteq K \setminus (V_l' \setminus V_h)$ and $K \setminus ((V_h \setminus V_l') \cup (V_l' \setminus V_h)) \subseteq (Y_i^{l-1} \cup V_l') \oplus (X_i^{h-1} \cup V_h)$ hold. If $g(\tau_i(X_i^{h-1}, X_i^{h-1} \cup V_h)) \in V_l' \cap V_h$ was to hold then this would imply by Lemma 1 that $\tau_i(X_i^{h-1}, X_i^{h-1} \cup V_h) = \tau_i(Y_i^{l-1}, Y_i^{l-1} \cup V_l')$ holds since inclusion $g(\tau_i(X_i^{h-1}, X_i^{h-1} \cup V_h)) \cup a(\tau_i(X_i^{h-1}, X_i^{h-1} \cup V_h)) \subseteq (Y_i^{l-1} \cup V_l') \oplus (X_i^{h-1} \cup V_h)$ would hold. Note that this reasoning would be true for any agent $j$ that receives a good during the last step of Algorithm 1 applied to $(\boldsymbol{e}, \boldsymbol{\tau})$. All in all, we have shown that for any good $o' \in V_h$ belonging to the initial endowment $e_j$ of agent $j$, the following implication holds:

$$g(\tau_j(X_j^{h-1}, X_j^{h-1} \cup V_h)) \in V_t' \cap V_h \Rightarrow \tau_j(X_j^{h-1}, X_j^{h-1} \cup V_h) = \tau_j(Y_j^{t-1}, Y_j^{t-1} \cup V_t') \quad (1)$$

where $t$ is the step of Algorithm 1 applied to $(\boldsymbol{e'}, \boldsymbol{\tau})$ where $o'$ is visited by a cycle in $G_t'$. Using this property, we show by contradiction that $g(\tau_i(X_i^{h-1}, X_i^{h-1} \cup V_h)) \in V_l'$ holds. We denotes by $o_1, o_2, \ldots, o_c$ the goods of $S_l'(o) \setminus \{o\}$ visited by $\mathcal{C}'$. $\mathcal{C}'$ is described in Figure 10, where edge $(o', f')$ means that $f'$ is assigned to agent $\delta(o')$ at the end of step $h$ of Algorithm 1 applied to $(\boldsymbol{e}, \boldsymbol{\tau})$, and therefore $g(\tau_{\delta(o')}(X_{\delta(o')}^{h-1}, X_{\delta(o')}^{h-1} \cup V_h)) = f'$ holds by definition of Algorithm 1. We denote by $i_j = \delta(o_j)$ the owner of $o_j$. By contradiction, assume that $g(\tau_i(X_i^{h-1}, X_i^{h-1} \cup V_h)) = o_1 \in V_h \setminus V_l'$ holds. This implies that there exists a step $l_1 < l$ such that $o_1$ is visited by a cycle, say $\mathcal{C}_1$, in $G_{l_1}'$. Consider now step $l_1$ and cycle $\mathcal{C}_1$. If $o_2 \notin V_{l_1}'$ then there must be a step $l_2 < l_1$ where $o_2$ is visited by a cycle, say $\mathcal{C}_2$, in $G_{l_2}'$. On the other hand, if $o_2 \in V_{l_1}'$ then $g(\tau_{i_1}(X_{i_1}^{h-1}, X_{i_1}^{h-1} \cup V_h)) = o_2 \in V_{l_1}' \cap V_h$ holds and (1) implies $\tau_{i_1}(X_{i_1}^{h-1}, X_{i_1}^{h-1} \cup V_h) = \tau_{i_1}(Y_{i_1}^{l_1-1}, Y_{i_1}^{l_1-1} \cup V_{l_1}')$. In other words, $\mathcal{C}_1$ visits also $o_2$ in $G_{l_1}'$. To simplify notation, we assume in that case that $l_2 = l_1$ and $\mathcal{C}_2 = \mathcal{C}_1$. Note that in both cases we have $l_2 \leq l_1$. Consider now step $l_2$ and cycle $\mathcal{C}_2$ ... If $o_t \notin V_{l_{t-1}}'$ then there must be a step $l_t < l_{t-1}$ where $o_t$ is visited by a cycle, say $\mathcal{C}_t$, in $G_{l_t}'$. On the other hand, if $o_t \in V_{l_{t-1}}'$ then $g(\tau_{i_{t-1}}(X_{i_{t-1}}^{h-1}, X_{i_{t-1}}^{h-1} \cup V_h)) = o_t \in V_{l_{t-1}}' \cap V_h$ holds and (1) implies $\tau_{i_{t-1}}(X_{i_{t-1}}^{h-1}, X_{i_{t-1}}^{h-1} \cup V_h) = \tau_{i_{t-1}}(Y_{i_{t-1}}^{l_{t-1}-1}, Y_{i_{t-1}}^{l_{t-1}-1} \cup V_{l_{t-1}}')$. In other words, $\mathcal{C}_{t-1}$ visits also $o_t$

in $G'_{l_{t-1}}$. We assume in that case that $l_t = l_{t-1}$ and $\mathcal{C}_t = \mathcal{C}_{t-1}$. Note that in both cases we have $l_t \leq l_{t-1}$. All in all, we have $l > l_1 \geq \ldots \geq l_t$. This implies that $o \in V'_{l_t}$ holds since we have assumed $o \in V'_l$. Therefore, (1) implies that $\tau_{i_t}(X_{i_t}^{h-1}, X_{i_t}^{h-1} \cup V_h) = \tau_{i_t}(Y_{i_t}^{l_t-1}, Y_{i_t}^{l_t-1} \cup V'_{l_t})$ holds. But this implies $g(\tau_{i_t}(Y_{i_t}^{l_t-1}, Y_{i_t}^{l_t-1} \cup V'_{l_t})) = g(\tau_{i_t}(X_{i_t}^{h-1}, X_{i_t}^{h-1} \cup V_h)) = o$, and therefore we know by definition of Algorithm 1 that $o$ is visited $\mathcal{C}_t$ in $G'_{l_t}$. This leads to a contradiction with $o \in V'_l$ since $l > l_t$.

We have shown in the previous paragraph that $g(\tau_i(X_i^{h-1}, X_i^{h-1} \cup V_h)) \in V'_l \cap V_h$ holds. This implies by (1) that $\tau_i(X_i^{h-1}, X_i^{h-1} \cup V_h) = \tau_i(Y_i^{l-1}, Y_i^{l-1} \cup V'_l)$ holds. Actually, this property is true for each good visited by $\mathcal{C}'$ since they all belong to $V_h$. Therefore, cycle $p'_l(o)$ that visits $o$ in $G'_l$ has to be the same as $\mathcal{C}'$. This concludes the proof since we have already shown that $\mathcal{C}'$ is the last cycle of the ATTC procedure applied to $(\boldsymbol{e'}, \boldsymbol{\tau})$ visiting a good belonging to agent $i$. □

The proof of Proposition 4 follows directly from Lemma 14. Let $(\boldsymbol{e'}, \boldsymbol{\tau})$ denote the assignment problem when the manipulator use hiding manipulation, and let $(\boldsymbol{e}, \boldsymbol{\tau})$ denote the assignment problem when the manipulator use the splitting manipulation computed by Algorithm 3. We can assume without loss of generality that $\boldsymbol{\tau}$ is the preference profile computed by Algorithm 3 since it differs from the input preference profile only by its components corresponding to the goods of $e_i^w$. Each identity associated with a good of $e_i^w$ prefers her good to any other good. Therefore, during the first step of Algorithm 1 applied to $(\boldsymbol{e}, \boldsymbol{\tau})$, each good of $e_i^w$ is assigned to its owner since it belongs to a cycle of size 1. Therefore, we can state by Lemma 14 that the outcome of Algorithm 1 is the same for both $(\boldsymbol{e}, \boldsymbol{\tau})$ and $(\boldsymbol{e'}, \boldsymbol{\tau})$.

## Appendix F. Omitted Proof for Subsection 8.1

This appendix is dedicated to the study of Algorithm 2. We show that this algorithm is able to transform any splitting manipulation into a misreport manipulation with the same outcome for the manipulator.

### F.1 Initialization

We start our study of Algorithm 2 by the first **while** loop (lines 1–6). We show that modifications performed on preference profile $\boldsymbol{\tau}$ do not change the outcome of Algorithm 1 for the manipulator. To this end, we denote by $\boldsymbol{\tau}$ and $\boldsymbol{\tau'}$ the preference profiles before and after one loop i.e., modification performed in lines 4–5 of Algorithm 2. We denote by $G'_k = (V'_k, E'_k)$ (resp. $G_k = (V_k, E_k)$) the ATTC graph during step $k$ of Algorithm 1 in the specific case where $\boldsymbol{\tau'}$ (resp. $\boldsymbol{\tau}$) is revealed, and let $Y_i^k$ (resp. $X_i^k$) denote the set of goods assigned to agent $i$ after the $k$ first steps in that case. Let $p'_k(o)$ (resp. $p_k(o)$) denote the directed path in $G'_k$ (resp. $G_k$) which starts from good $o$, and let $S'_k(o)$ (resp. $S_k(o)$) denote the set of goods visited by $p'_k(o)$ (resp. $p_k(o)$). Finally, for any good $o' \in S'_k(o)$ (resp. $o'' \in S_k(o)$), let $[p'_k]_o^{o'}$ (resp. $[p_k]_o^{o''}$) denote the subpath of $p'_k(o)$ (resp. $p_k(o)$) which starts from good $o$ and finishes at good $o'$ (resp. $o''$).

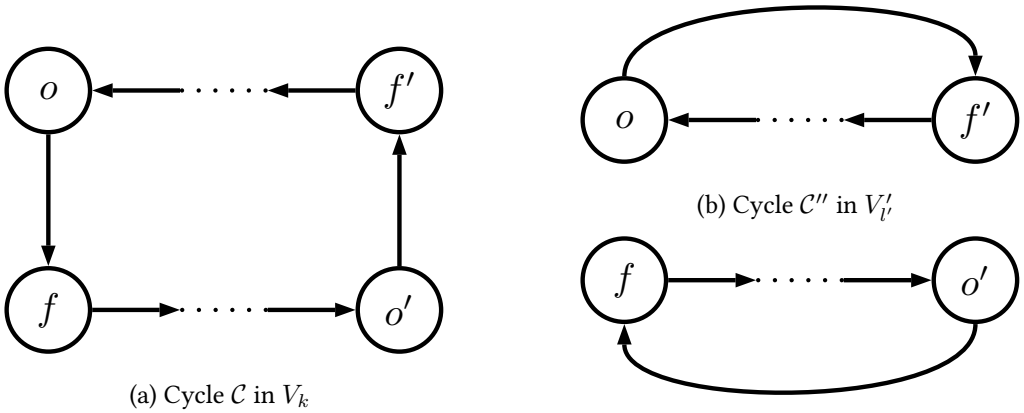
Note that $\boldsymbol{\tau'}$ only differs from $\boldsymbol{\tau'}$ by its components $\delta(o)$ and $\delta(o')$, corresponding to the preferences revealed by the manipulator for goods $o$ and $o'$, respectively. We denote by $i_o$ and $i_{o'}$ the identities used by the manipulator for goods $o$ and $o'$, respectively. Therefore, $\mathcal{I} = \{i_o, i_{o'}\}$ is the set of identities that do not reveal the same preferences in $\boldsymbol{\tau}$ and $\boldsymbol{\tau'}$, as defined in Annexe D. The following lemma shows that replacing $\boldsymbol{\tau}$ with $\boldsymbol{\tau'}$ do not change the outcome of Algorithm 1.

**Lemma 15.** *For any agent $j \in N$, $\varphi_j(\boldsymbol{e}, \boldsymbol{\tau}) = \varphi_j(\boldsymbol{e}, \boldsymbol{\tau'})$ holds.*

*Proof.* Let $k$ denote the step of Algorithm 1 where $f$, $f'$, $o$ and $o'$ arel visited by cycle $\mathcal{C}$ in $G_k$ (these elements are described in lines 1–2 of Algorithm 2). Let $l$ be the first step such that $o' \in S_l(o)$, i.e., $p_l(f)$ reaches $o'$ in $G_l$. Note that such step should exist since $o' \in S_k(f)$ by definition. Furthermore, $k \geq l$ trivially holds, and Corollary 5 implies that $[p_l(f)]_f^{o'} = [p_k(f)]_f^{o'}$ holds. Therefore, $o$ is not visited by $[p_l(f)]_f^{o'}$, as illustrated by Figure 11a, and $o'$ is the first good of $\bigcup_{j \in \mathcal{I}} e_j = \{o, o'\}$ visited by $p_l(f)$. This implies by Lemma 10 that $f \in V_l'$ and $[p_l'(f)]_f^{o'} = [p_l(f)]_f^{o'}$ hold. Furthermore, we know that $(o', f) \in E_l'$ since $f$ labels the root of $\tau'_{\delta(o')}$. Therefore, the concatenation of $[p_l'(f)]_f^{o'}$ and $(o', f)$ forms a cycle in $G_l'$, as illustrated in Figure 11c, and $f$ is assigned to $\delta(o')$. Furthermore, for any good $o''$ visited by this cycle, if $(o'', f'')$ belongs to $[p_l(f)]_f^{o'}$ then it also belongs to $[p_l'(f)]_f^{o'}$, and $\delta(o'')$ receives $f''$ as in the case where Algorithm 1 is applied to $(\boldsymbol{e}, \boldsymbol{\tau})$ since $[p_l'(f)]_f^{o'} = [p_l(f)]_f^{o'}$ holds. By using similar arguments, it is easy to check that $f'$ is assigned to $i_o$, and for each other good $o''$ visited by $[p_{l'}(f')]_{f'}^{o}$, where $l'$ is the first step of Algorithm 1 where $o \in S_{l'}(f')$, if edge $(o'', f'')$ belongs to $[p_{l'}(f')]_{f'}^{o}$ then it also belongs to $[p_{l'}'(f)]_f^{o'}$, and $\delta(o'')$ receives $f''$ as in the case where Algorithm 1 is applied to $(\boldsymbol{e}, \boldsymbol{\tau})$.

Finally, let $B$ denote the set of goods visited by $\mathcal{C}$. In order to show that the outcome of Algorithm 1 is preserved for the goods outside of $B$, we remove the goods of $B$ from initial endowment $\boldsymbol{e}$ and assign these goods to their assignee described in the previous paragraph. Let $\boldsymbol{e'}$ denote the initial endowment after removing the goods of $B$. We consider now the outcomes of Algorithm 1 applied to both $(\boldsymbol{e'}, \boldsymbol{\tau})$ and $(\boldsymbol{e'}, \boldsymbol{\tau'})$. Note that the goods of $B$ are assigned before starting Algorithm 1, as described in Appendix E. This initial assignment ensures that preference profile $\boldsymbol{\tau}$ remains consistent after removing the goods of $B$ from $\boldsymbol{e}$. Lemma 14 implies that the outcome of Algorithm 1 applied to $(\boldsymbol{e}, \boldsymbol{\tau})$ and $(\boldsymbol{e}, \boldsymbol{\tau'})$ (respectively, applied to $(\boldsymbol{e}, \boldsymbol{\tau'})$ and $(\boldsymbol{e'}, \boldsymbol{\tau'})$) is the same for the goods of $K \setminus B$. Therefore, it is enough to show that the goods of $K \setminus B$ are assigned to the same identities for $(\boldsymbol{e'}, \boldsymbol{\tau})$ and $(\boldsymbol{e'}, \boldsymbol{\tau'})$ to conclude the proof. We have shown in the previous paragraph that the goods of $B \setminus \{f, f'\}$ are assigned to the same agents by Algorithm 1 applied to $(\boldsymbol{e}, \boldsymbol{\tau})$ and $(\boldsymbol{e}, \boldsymbol{\tau'})$. Therefore, these goods are initially assigned to the same agents before applying Algorithm 1 to $(\boldsymbol{e'}, \boldsymbol{\tau})$ and $(\boldsymbol{e'}, \boldsymbol{\tau'})$. Furthermore, by assumption on splitting manipulations we know that the initial endowment of identities $i_o$ and $i_{o'}$ contain only goods $o$ and $o'$, respectively. Therefore, their initial endowments are empty in $\boldsymbol{e'}$ and they are ignored during Algorithm 1 applied to $(\boldsymbol{e'}, \boldsymbol{\tau})$ and $(\boldsymbol{e'}, \boldsymbol{\tau'})$. All in all, assignments problems $(\boldsymbol{e'}, \boldsymbol{\tau})$ and $(\boldsymbol{e'}, \boldsymbol{\tau'})$ only differ by the preference revealed by agents $i_o$ and $i_{o'}$ which are ignored during Algorithm 1. Therefore, Algorithm 1 provides the same outcome for both $(\boldsymbol{e'}, \boldsymbol{\tau})$ and $(\boldsymbol{e'}, \boldsymbol{\tau'})$. □

As illustrated in Figure 11, cycle $C$ gives rise to two different cycles when $\boldsymbol{\tau'}$ is revealed instead of $\boldsymbol{\tau}$. Furthermore, as mentioned in Lemma 14, any other cycle occurring during Algorithm 1 when $\boldsymbol{\tau}$ is revealed will be also occur during Algorithm 1 when $\boldsymbol{\tau'}$ is revealed. Hence, the number of cycles including at least of the goods belonging to the manipulator is increased by one after each loop of the initialization phase. Therefore, the **while** loop stop after at most $|e_i|$ steps. Lemma 15 means that the splitting manipulation obtained at the end of initialization phase provides the same outcome as to the initial splitting manipulation. Furthermore, no cycle containing more than two goods of the manipulator appears during Algorithm 1 applied to the resulting preference profile.




(a) Cycle $\mathcal{C}$ in $V_k$

(b) Cycle $\mathcal{C}''$ in $V'_{l'}$

(c) Cycle $\mathcal{C}'$ in $V'_l$

Figure 11: Differences between a path in $V_k$ an $V'_k$

### F.2 Main Algorithm

Let $\boldsymbol{\tau}$ denote the preference profile obtained at the end of initialization phase, and let $\boldsymbol{\tau'}$ denote the preference profile where the manipulator reveals the misreport manipulation $\tau'_i$ computed by Algorithm 2. For sake of simplicity, we assume in $(\boldsymbol{e}, \boldsymbol{\tau'})$ that the manipulator use her fictitious identities when she reveals her preferences. This is without loss of generality because $\tau'_i$ is a path, and therefore the corresponding preferences are not conditional to the goods assigned to the manipulator during Algorithm 1. All other notations remain the same as in the previous subsection, with respect to the new definition of $\boldsymbol{\tau}$ and $\boldsymbol{\tau'}$. The following lemma shows that the outcome of Algorithm 1 is the same for the manipulator when $\boldsymbol{\tau'}$ is revealed instead of $\boldsymbol{\tau}$.

**Lemma 16.** *The goods assigned to the identities of the manipulator during Algorithm 1 are the same when $\boldsymbol{\tau'}$ is revealed instead of $\boldsymbol{\tau}$.*

*Proof.* We prove this statement by induction on the number of identities used by the manipulator i.e., on the size of the initial endowment of the manipulator.

*Base case*: Assume first that $e_i$ contains a single good $o$ and $\mathcal{I} = \{i\}$. Let $f$ be the good assigned to the manipulator when $\boldsymbol{\tau}$ is revealed, and $l$ be the first step of Algorithm 1 when $o$ belongs to $S_l(f)$, i.e. $o$ is reached by the directed path $p_l(f)$. We know by Lemma 10 that $f \in V'_l$ and $[p'_l(f)]^o_f = [p_l(f)]^o_f$. Furthermore, we know that $(o, f)$ belongs to $E'_l$ because $f$ labels the root of $\tau'_i$. Therefore, during step $l$ of Algorithm 1 when $\boldsymbol{\tau'}$ is revealed, the concatenation of $[p'_l(f)]^o_f$ and $(o, f)$ forms a cycle containing $o$. Hence, the same good $f$ is assigned to agent $i$ during Algorithm 1 applied to $(\boldsymbol{e}, \boldsymbol{\tau})$ and $(\boldsymbol{e}, \boldsymbol{\tau'})$.

*Induction step*: Let $r$ be the number of goods belonging to the initial endowment of the manipulator. Let $\mathcal{C}$ be the cycle chosen in line 7 of Algorithm 2 during the last **while** loop. Notations $o, f$ and $B$ are defined as in line 10–11 of Algorithm 2. Note that $\delta(o)$ is an identity of the manipulator, and therefore $\tau'_{\delta(o)}$ is the LP-tree computed by Algorithm 2. Hence, $f$ labels the root of $\tau'_{\delta(o)}$ since by definition this is the last element inserted by function *add_root* in the second loop **while** of Algorithm 2. Let $k$ denote the step where $\mathcal{C}$ occurs in $G_k$, and let $l$ be the first step of Algorithm 1 where $o$ belongs to $S_l(f)$. Since $k$ is the last step where $o \in V_k$, inequality $k \geq l$

trivially holds. This implies by Corollary 5 that $[p_l(f)]^o_f = [p_k(f)]^o_f$ holds. Therefore, no good belonging to $\bigcup_{j\in\mathcal{I}} e_j \setminus \{o\}$ is visited by $[p_l(f)]^o_f$ because $o$ is the only good of the manipulator visited by $\mathcal{C}$ (as a result of initialization phase). Therefore, $o$ is the first good of $\bigcup_{j\in\mathcal{I}} e_j$ visited by $p_l(f)$, and this implies by Lemma 10 that $[p'_l(f)]^o_f = [p_l(f)]^o_f$ holds. Furthermore, we know that $(o, f)$ belongs to $E'_l$ since $o$ labels the root of $\tau'_{\delta(o)}$. Therefore, the concatenation of $[p'_l(f)]^o_f$ and $(o, f)$ forms the same cycle as $\mathcal{C}$ in $G'_l$. The goods visited by $\mathcal{C}$ are assigned to the same agents as they are when $\boldsymbol{\tau}$ is revealed.

Finally, let $B$ denotes the set of goods visited by $\mathcal{C}$. In order to show that the outcome is preserved for the goods outside of $B$, we remove the goods of $B$ from initial endowment $\boldsymbol{e}$ and we assign these goods to their assignee as described in the previous paragraph. Let $\boldsymbol{e'}$ denote the initial endowment after removing the goods of $B$. We consider now the outcome of the ATTC procedure applied to both $(\boldsymbol{e'}, \boldsymbol{\tau})$ and $(\boldsymbol{e'}, \boldsymbol{\tau'})$. Note that the goods of $B$ assigned before starting Algorithm 1, as described in Appendix E. Let $\tau''_i$ denote the outcome of Algorithm 2 applied to $(\boldsymbol{e'}, \boldsymbol{\tau})$, and let $\boldsymbol{\tau''}$ denote the preference profile where the identities belonging to the manipulator reveal $\tau''_i$ instead of $\tau'_i$ (other agents reveals the same preference as in $\boldsymbol{\tau}$). Note first that the initial endowment of $\delta(o)$ is empty in $\boldsymbol{e'}$. Therefore this identity is ignored during Algorithm 1 applied to $(\boldsymbol{e'}, \boldsymbol{\tau})$. This means also that the number of identities used by the manipulator during Algorithm 1 applied to $(\boldsymbol{e'}, \boldsymbol{\tau})$, or equivalently the number of goods belonging to him in $\boldsymbol{e'}$, is equal to $r - 1$. Therefore, by induction hypothesis we can state that the outcomes of Algorithm 1 applied to $(\boldsymbol{e'}, \boldsymbol{\tau})$ and $(\boldsymbol{e'}, \boldsymbol{\tau''})$ are the same for each identity in $\mathcal{I}$ belonging to the manipulator. On the other hand, we know by Lemma 14 that the outcomes of Algorithm 1 applied to $(\boldsymbol{e}, \boldsymbol{\tau})$ and $(\boldsymbol{e'}, \boldsymbol{\tau})$ (respectively for $(\boldsymbol{e}, \boldsymbol{\tau'})$ and $(\boldsymbol{e'}, \boldsymbol{\tau'})$) are the same for the goods of $K \setminus B$. Furthermore, Lemma 14 states that the sequence of cycles visiting goods of $\bigcup_{j\in\mathcal{I}} e_j \setminus \{o\}$ is the same for Algorithm 1 applied to $(\boldsymbol{e}, \boldsymbol{\tau})$ and $(\boldsymbol{e'}, \boldsymbol{\tau})$. Therefore, LP-trees $\tau''_i$ and $\tau'_i$, computed by Algorithm 2 with parameters $(\boldsymbol{e}, \boldsymbol{\tau})$ and $(\boldsymbol{e'}, \boldsymbol{\tau})$, respectively, only differs by the position of $f$ i.e., the vertex with label $f$ is not the root of $\tau''_i$. However, since $f$ does not belong to $\boldsymbol{e'}$ and $\tau'_i$ and $\tau''_i$ are paths (and therefore represent lexicographic preferences), the outcomes of Algorithm 1 applied to $(\boldsymbol{e'}, \boldsymbol{\tau'})$ and $(\boldsymbol{e'}, \boldsymbol{\tau''})$ are the same. All in all, each identity of the manipulator receives the same good during Algorithm 1 applied to $(\boldsymbol{e}, \boldsymbol{\tau})$ and $(\boldsymbol{e}, \boldsymbol{\tau'})$. □

## References


Afacan, M. (2014). Fictitious students creation incentives in school choice problems. *Economic Theory*, *56*(3), 493–514.

Atlamaz, M., & Klaus, B. (2007). Manipulation via endowments in exchange markets with indivisible goods. *Social Choice and Welfare*, *28*(1), 1–18.

Aziz, H., Bachrach, Y., Elkind, E., & Paterson, M. (2011). False-name manipulations in weighted voting games. *Journal of Artificial Intelligence Research*, *40*(1), 57–93.

Aziz, H., Biró, P., Lang, J., Lesca, J., & Monnot, J. (2016). Optimal reallocation under additive and ordinal preferences. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems*, pp. 402–410.

Aziz, H., Bouveret, S., Lang, J., & Mackenzie, S. (2017). Complexity of manipulating sequential allocation. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'17)*, pp. 328–334.

Aziz, H., Kalinowski, T., Walsh, T., & Xia, L. (2016). Welfare of sequential allocation mechanisms for indivisible goods. In *Proceedinggs of the 22nd European Conference on Artificial Intelligence (ECAI'16)*, pp. 787–794.

Bartholdi, J.J., I., Tovey, C., & Trick, M. (1989). The computational difficulty of manipulating an election. *Social Choice and Welfare*, *6*(3), 227–241.

Booth, R., Chevaleyre, Y., Lang, J., Mengin, J., & Sombattheera, C. (2010). Learning conditionally lexicographic preference relations. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI'10)*, pp. 269–274.

Boutilier, C., Brafman, R. I., Domshlak, C., Hoos, H. H., & Poole, D. (2004). CP-Nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, *21*, 135–191.

Cechlárová, K. (2009). On the complexity of the Shapley-Scarf economy with several types of goods. *Kybernetika*, *45*(5), 689–700.

Cechlárová, K., & Lacko, V. (2012). The kidney exchange problem: How hard is it to find a donor?. *Annals of Operations Research*, *193*(1), 255–271.

Chen, Y., & Sönmez, T. (2002). Improving efficiency of on-campus housing: An experimental study. *American Economic Review*, *92*(5), 1669–1686.

Chevaleyre, Y., Endriss, U., & Maudet, N. (2007). Allocating goods on a graph to eliminate envy. In *Proceedings of the 22th International Joint Conference on Artificial Intelligence (IJCAI'07)*, pp. 700–705.

Conitzer, V. (2008). Anonymity-proof voting rules. In *Proceedings of the 4th Workshop on Internet and Network Economics (WINE'08)*, pp. 295–306.

Conitzer, V., & Xia, L. (2012). Paradoxes of multiple elections: An approximation approach. In *Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR'12)*, pp. 179–187.

de Keijzer, B., Bouveret, S., Klos, T., & Zhang, Y. (2009). On the complexity of efficiency and envy-freeness in fair division of indivisible goods with additive preferences. In *Proceedings of the First International Conference on Algorithmic Decision Theory (ADT'09)*, pp. 98–110. Springer.

Endriss, U., Maudet, N., Sadri, F., & Toni, F. (2006). Negotiating socially optimal allocations of resources. *Journal of Artificial Intelligence Research*, *25*, 315–348.

Faliszewski, P., Hemaspaandra, E., & Hemaspaandra, L. A. (2010). Using complexity to protect elections. *Communications of the ACM*, *53*(11), 74–82.

Faliszewski, P., & Procaccia, A. D. (2010). Ai's war on manipulation: Are we winning?. *AI Magazine*, *31*(4), 53–64.

Fujita, E., Lesca, J., Sonoda, A., Todo, T., & Yokoo, M. (2015). A complexity approach for core-selecting exchange with multiple indivisible goods under lexicographic preferences. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*, pp. 907–913.

Gigerenzer, G., & Goldstein, D. (1996). Reasoning the fast and frugal way: Models of bounded rationality. *Psychological review*, *103*(4), 650.

Gold, E. M. (1978). Complexity of automaton identification from given data. *Information and Control*, *37*(3), 302 – 320.

Gourvès, L., Lesca, J., & Wilczynski, A. (2017). Object allocation via swaps along a social network. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*, pp. 213–219.

Lang, J., Mengin, J., & Xia, L. (2012). Aggregating conditionally lexicographic preferences on multi-issue domains. In *Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming (CP'12)*, pp. 973–987.

Lesca, J., Minoux, M., & Perny, P. (2013). Compact versus noncompact lp formulations for minimizing convex choquet integrals. *Discrete Applied Mathematics*, *161*(1), 184 – 199.

Lesca, J., & Perny, P. (2010). LP solvable models for multiagent fair allocation problems.. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI'10)*, pp. 393–398.

Luo, S., & Tang, P. (2015). Mechanism design and implementation for lung exchange. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pp. 209–215.

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

Moulin, H. (2008). Proportional scheduling, split-proofness, and merge-proofness. *Games and Economic Behavior*, *63*(2), 567–587.

Ohta, N., Conitzer, V., Satoh, Y., Iwasaki, A., & Yokoo, M. (2008). Anonymity-proof Shapley value: extending Shapley value for coalitional games in open environments. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'08)*, pp. 927–934.

Pápai, S. (2003). Strategyproof exchange of indivisible goods. *Journal of Mathematical Economics*, *39*(8), 931 – 959.

Pápai, S. (2007). Exchange in a general market with indivisible goods. *Journal of Economic Theory*, *132*(1), 208 – 235.

Pini, M. S., Rossi, F., Venable, K. B., & Walsh, T. (2011). Manipulation complexity and gender neutrality in stable marriage procedures. *Autonomous Agents and Multi-Agent Systems*, *22*(1), 183–199.

Roth, A. E., & Postlewaite, A. (1977). Weak versus strong domination in a market with indivisible goods. *Journal of Mathematical Economics*, *4*(2), 131–137.

Roth, A. E., Sönmez, T., & Ünver, M. U. (2004). Kidney exchange. *The Quarterly Journal of Economics*, *119*(2), 457–488.

Saban, D., & Sethuraman, J. (2014). A note on object allocation under lexicographic preferences. *Journal of Mathematical Economics*, *50*, 283 – 289.

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

Sikdar, S., Adalı, S., & Xia, L. (2017). Mechanism design for multi-type housing market. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI-17)*, pp. 684–690.

Sikdar, S., Adalı, S., & Xia, L. (2018). Top-trading-cycles mechanisms with acceptable bundles. In *Proceedings of the 11th Multidisciplinary Workshop on Advances in Preference Handling (M-PREF18)*.

Sönmez, T. (1999). Strategy-proofness and essentially single-valued cores. *Econometrica*, *67*(3), 677–689.

Sonoda, A., Fujita, E., Todo, T., & Yokoo, M. (2014). Two case studies for trading multiple indivisible goods with indifferences. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI'14)*, pp. 791–797.

Sun, Z., Hata, H., Todo, T., & Yokoo, M. (2015). Exchange of indivisible objects with asymmetry. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 97–103.

Teo, C.-P., Sethuraman, J., & Tan, W.-P. (2001). Gale-Shapley stable marriage problem revisited: Strategic issues and applications. *Management Science*, *47*(9), 1252–1267.

Todo, T., & Conitzer, V. (2013). False-name-proof matching. In *Proceedings of the 12th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'13)*, pp. 311–318.

Todo, T., Iwasaki, A., & Yokoo, M. (2011). False-name-proof mechanism design without money. In *Proceedings of the 10th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'11)*, pp. 651–658.

Todo, T., Sun, H., & Yokoo, M. (2014). Strategyproof exchange with multiple private endowments. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI'14)*, pp. 805–811.

Yokoo, M., Conitzer, V., Sandholm, T., Ohta, N., & Iwasaki, A. (2005). Coalitional games in open anonymous environments. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, pp. 1668–1669.

Yokoo, M., Sakurai, Y., & Matsubara, S. (2004). The effect of false-name bids in combinatorial auctions: New fraud in internet auctions. *Games and Economic Behavior*, *46*(1), 174–188.