Online Housing Market

Julien Lesca
Université Paris-Dauphine, PSL, CNRS, LAMSADE
julien.lesca@dauphine.fr

Abstract

We study an online variant of the celebrated housing market problem, where each agent owns a single house and seeks to exchange it based on her preferences. In this online setting, agents may arrive and depart at any time, meaning not all agents are present in the housing market simultaneously. We extend the well-known serial dictatorship and top trading cycle mechanisms to the online scenario, aiming to retain their desirable properties, such as Pareto efficiency, individual rationality, and strategy-proofness. These extensions also seek to prevent agents from strategically delaying their arrivals or advancing their departures. We demonstrate that achieving all these properties simultaneously is impossible and present several variants that achieve different subsets of these properties.

1 Introduction

Allocating indivisible resources to agents is a fundamental problem in computational social choice, situated at the intersection of economics [Thomson, 2011] and computer science [Klaus *et al.*, 2016; Manlove, 2013]. The decision-maker in such problems must account for both the preferences of the agents over the resources and their strategic behavior. In this paper, we focus on a specific problem known as the housing market in the matching theory literature [Shapley and Scarf, 1974], where each agent is endowed with a single resource they are willing to exchange for another. Additionally, we assume that no monetary compensation is allowed to offset any unfavorable exchanges during the process. Despite its simplicity, this problem has numerous applications, including the exchange of dormitory rooms among students [Chen and Sönmez, 2002], the trade of used items [Swapz, 2025], kidney exchanges where incompatible donor/recipient pairs exchange organs for transplants [Ferrari *et al.*, 2015], and more (see, e.g., [Biró, 2017] for additional applications).

In this paper, we assume that preferences over resources are ordinal. The procedure for reallocating resources among agents is centralized, with the decision-maker aiming to produce an allocation that is as efficient as possible. Under ordinal preferences, Pareto optimality serves as an appropriate efficiency measure, seeking allocations where no agent can be made better off without disadvantaging another. In the standard offline setting, this requirement can be met using the serial dictatorship procedure, also known as the picking sequence [Bouveret and Lang, 2014]. In this procedure, agents are arranged in a specific order, and each agent, one by one, selects their most preferred resource from the pool of remaining resources. However, this process is not individually rational, as some agents may end up with resources less desirable than their initial endowments. This is problematic, as agents might be discouraged from participating in the trade to avoid being worse off. The well-known Gale's top trading cycle (TTC) procedure [Shapley and Scarf, 1974] addresses this issue by ensuring individual rationality while producing a Pareto-optimal allocation.

Since the procedure is centralized, the decision maker must interact with agents to gather their preferences over resources. However, during this process, agents may misreport their preferences in an attempt to manipulate the procedure and achieve a more favorable outcome. Mechanism design seeks to create procedures where truthfully revealing preferences is a dominant strategy for agents[Hurwicz and Reiter, 2006]. Both the serial dictatorship and TTC procedures possess this property. Moreover, it has been proven that the TTC procedure is the only mechanism that is simultaneously individually rational, Pareto efficient, and strategy-proof[Ma, 1994].

The standard offline setting assumes that all agents participating in the exchange are available at the same time, and that the exchange procedure is conducted during this period. However, in many contexts, this requirement is unrealistic or too demanding. In more realistic scenarios, agents are only available during restricted time periods, meaning that some may not be able to participate in the exchange simultaneously. This is the case, for example, with online exchange websites[Swapz, 2025], where agents do not arrive at the marketplace at the same time and cannot remain indefinitely before their swaps take place. This type of scenario is referred to as online[Albers, 2003], or dynamic, in the context of matching[Baccara and Yariv, 2021].

Outline. Section 2 lists related works connected to the online problem under consideration. Section 3 defines the main properties we aim to achieve. Section 4 presents online mechanisms based on the serial dictatorship procedure, while Section 5 introduces multiple online variants of the TTC procedure.

2 Related Works

Multiple social choice problems have been examined through the lens of online procedures. In fair division, which aims to allocate resources fairly, several online variants have been explored[Aleksandrov and Walsh, 2020; Sankar *et al.*, 2021; Hosseini *et al.*, 2024]. For instance, strategy-proofness and Pareto efficiency have been studied in this context[Aleksandrov *et al.*, 2015], along with envy-freeness. Various extensions of the serial dictatorship procedure have been proposed to achieve these properties[Aleksandrov and Walsh, 2019]. The online electric vehicle charging problem[Gerding *et al.*, 2019], which focuses on scheduling charging for customers arriving in an online fashion, also shares similarities with our problem. However, in both cases, the online setting differs from ours, as resources do not arrive dynamically and are known to the procedure from the outset. Closer to our setting, as it addresses a one-to-one matching problem, an online version where preferences are elicited incrementally by querying agents has been considered[Hosseini *et al.*, 2021]. A procedure that elicits preferences to achieve Pareto-optimality while minimizing the number of queries has been proposed.

Numerous works have focused on dynamic kidney exchange[Ünver, 2010; Ashlagi and Roth, 2021], where donors and recipients arrive in an online fashion. Most of these studies consider compatibility (0-1 preferences) rather than ordinal preferences. Additionally, they often assume that the allocation process can be probabilistic, rely on probabilistic assumptions about arrivals and departures, and aim to reduce the expected waiting time before a transplant[Bloch and Cantala, 2017; Anderson *et al.*, 2017; Baccara *et al.*, 2020]. Other studies focus on maximizing the expected number of matched pairs[Awasthi and Sandholm, 2009; Dickerson *et al.*, 2012]. The design of strategy-proof and individually rational mechanisms has also been considered for transplant centers participating in national exchange programs[Ashlagi *et al.*, 2015; Hajaj *et al.*, 2015].

3 Preliminary

Let $N = \{1, \ldots, n\}$ denote the set of agents. Each agent i owns a single good[1] e_i, called her initial endowment. Let $\mathcal{T} = [t^-, t^+]$ denote the timeline during which the market is open, where t^- and t^+ represent the opening and closing times, respectively. For each agent i, a_i and d_i represent her (earliest) arrival and (latest) departure times in the market, with $a_i, d_i \in \mathcal{T}$ such that $t^- \leq a_i < d_i \leq t^+$. To simplify the setting, we assume that no two arrival or departure times occur simultaneously. Let $E = \{e_1, \ldots, e_n\}$ represent the set of items. Each agent i has a strict ordinal preference $\succ_i$ over the items in E, such that $e_j \succ_i e_k$ means agent i strictly prefers e_j over e_k. Furthermore, symbol $\succeq_i$ stands for $\succ_i$ or $=$. An instance of the online exchange problem is represented by the set of tuples $\{(i, e_i, a_i, d_i, \succ_i)\}_{i \in N}$. Without loss of generality, we assume that the agent indices are ordered by increasing arrival times, i.e., such that $a_1 < a_2 < \ldots < a_n$ hold. Let $\mathcal{I}$ denote the entire set of possible instances.

Our goal is to allocate to each agent a single item such that each item is assigned only once. In other words, we search for an allocation $M : N \rightarrow E$ such that $M(i) \neq M(j)$ for each $i \neq j$. The timeline constraints are such that each agent will leave the market with an item which belongs to an agent that arrived in the market earlier than her departure time. For any $t \in \mathcal{T}$, let $N_{<t} = \{i \in N : a_i < t\}$ denote the subset of agents arriving before t. For a given instance I, an allocation M is I-compatible if $M(i) \in E_{<d_i}$ holds $\forall i \in N$, where $E_{<t} = \{e_i : i \in N_{<t}\}$ for each $t \in \mathcal{T}$. For any instance I, the set of I-compatible allocations is denoted $\mathcal{M}(I)$.

[1]The resources are interchangeably referred to as goods or items

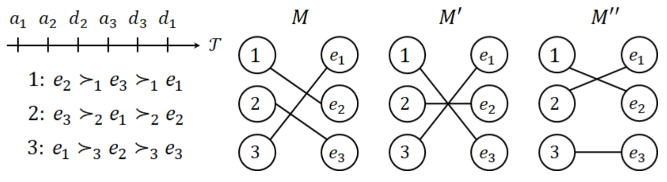

a_1 a_2 d_2 a_3 d_3 d_1
$\mathcal{T}$
1: $e_2 \succ_1 e_3 \succ_1 e_1$
2: $e_3 \succ_2 e_1 \succ_2 e_2$
3: $e_1 \succ_3 e_2 \succ_3 e_3$
M
M'
M''
1
2
3
e_1
e_2
e_3

Figure 1: Example instance (left) and allocations (right).

An exchange algorithm $\mathcal{A}$ returns an I-compatible allocation for each instance I. We assume that $\mathcal{A}_i(I)$ denotes the item allocated to agent i by algorithm $\mathcal{A}$ for instance I. In the online version of the problem, the algorithm should make a decision on the allocation for an agent when she leaves the market, without knowing the agents that arrive after her departure. In other words, the mechanism should make a decision only based on the agents that already visited the market by the departure time of the agent. For any $t \in \mathcal{T}$, let $I_{<t}$ denote a truncated copy of I restricted to the agents of $N_{<t}$.

Definition 1. *An exchange algorithm $\mathcal{A}$ is online if for each instance I and for each agent i, $\mathcal{A}_i(I) = \mathcal{A}_i(I_{<d_i})$ holds.*

The choice between online exchange algorithms should be guided by the properties fulfilled by the allocation that they compute. To compare these algorithms, we consider multiple standard desiderata properties. The first one is a standard definition of efficiency in multi-agent problems.

Definition 2. *Allocation M' Pareto-dominates allocation M if for each agent i, $M'(i) \succeq_i M(i)$, and for at least one agent j, $M'(j) \succ_j M(j)$. For a given instance I, let $\mathcal{S}$ denote a subset of $\mathcal{M}(I)$. Allocation M is $\mathcal{S}$-Pareto optimal ($\mathcal{S}$-PO), if there is no allocation M' of $\mathcal{S}$ that Pareto-dominates it.*

The standard notion of Pareto optimality corresponds to $\mathcal{M}(I)$-*PO*, as illustrated by the following example.

Example 1. *Consider instance I described in Figure 1. The timeline, containing the arrival and departure times of the agents, as well as their preferences are in the left part of the figure. Three different allocations are provided in the right part of the figure. For example, allocation M is such that $M(1) = e_2$, $M(2) = e_3$, and $M(3) = e_1$. Note that M is not I-compatible, as agent 2 receives an item from an agent who arrives later than d_2. Therefore, M does not belong to $\mathcal{M}(I)$. Allocations M' and M'' are I-compatible and $\mathcal{M}(I)$-PO.*

Restricting comparisons to I-compatible allocations allows for the existence of efficient solutions, even if better ones exist but are incompatible with the online setting.

Definition 3. *Exchange algorithm $\mathcal{A}$ is individually rational (IR), if for each instance I, $\mathcal{A}_i(I) \succeq_i e_i$ for any agent i.*

Individual rationality property incentivize agents to participate in the exchange algorithm, as no agent receive an item that is less desirable than her initial endowment. Another standard property related to the manipulative power of agents, through misreporting their preferences, is called incentive compatibility. We generalize this notion to take into account manipulations related to arrival and departure times.

Definition 4. *An exchange algorithm is strongly incentive compatible (SIC), if for each agent i and for each pair of instances I and I' such that $I' = I \setminus \{i, e_i, a_i, d_i, \succ_i\} \cup \{i, e_i, a'_i, d'_i, \succ'_i\}$ and $a_i \leq a'_i < d'_i \leq d_i$, $\mathcal{A}_i(I) \succeq_i \mathcal{A}_i(I')$ holds. It is a-IC (respectively d-IC) if the above inequality holds only when $d_i = d'_i$ (respectively $a_i = a'_i$).*[2] *Finally, it is weakly incentive compatible (WIC), if the above inequality holds only when both $a_i = a'_i$ and $d_i = d'_i$.*

The notion denoted WIC corresponds to the standard incentive compatibility in the offline case. Stronger notions should incentivize agents to arrive as early as possible (a-IC), as late as possible (d-IC), or both (SIC). We assume that a_i and d_i are agent i's earliest arrival and latest departure times, so she cannot arrive earlier than a_i or depart later than d_i. Definition 4 states that agent i cannot benefit by arriving later than a_i or departing earlier than d_i under an SIC mechanism.

4 Serial Dictatorship Procedure

In this section, we consider an exchange algorithm based on the serial dictatorship procedure, which is described in Algorithm 1. The standard version of the serial dictatorship procedure is based on a permutation of the agents defining the order in which each agent will choose her item among the remaining ones. We slightly generalize these permutations by considering permutation functions whose order depends on the instance. More formally, $\Pi : \mathcal{I} \rightarrow N^N$ denotes this permutation function, which is such that for any instance I and for any $i \in \{1, \ldots, n\}$, $\Pi_i(I)$ denotes the agent choosing at position i. We assume here that this permutation function only depends on the arrival and departure times of the agents. An example of such a permutation is the ascending departure permutation δ which ranks the agents according to their departure times,[3] and more specifically by increasing departure times, i.e., such that $d_{\delta(1)} < d_{\delta(2)} < \ldots < d_{\delta(n)}$.

In Algorithm 1, every iteration of the "for" loop corresponds to the departure of an agent, specifically agent $\delta(i)$, whose departure time is the i^{th} earliest. However, the order in which the agents choose their items is determined by the permutation function Π. Therefore, before agent $\delta(i)$ chooses her item, all agents that are ranked before her according to Π must choose their items. Once an agent has chosen an item, she is permanently matched to it, even if more desirable items arrive later. Note that the procedure $best_i$ returns the most preferred item of agent i from a given set of items.

Algorithm 1 is designed to be online, as only the part of the instance corresponding to agents who have already arrived, $I_{<d_{\delta(i)}}$, is used at iteration i, during the departure of agent $\delta(i)$. However, the permutation must be prefix-consistent i.e., the order of agents already matched to items must not change when new agents arrive. Otherwise, an agent might be matched multiple times or ignored by the algorithm.

[2]The notion of participation in [Mattei *et al.*, 2017] is close to a-IC, except that it does not require standard incentive compatibility.

[3]The input instance is omitted to simplify notations.

Algorithm 1 Online serial dictatorship procedure

```
Input: Permutation function $\Pi$.
 1: Initialize $M$ as an empty matching.
 2: $A \leftarrow \emptyset$. {Items already assigned.}
 3: $B \leftarrow \emptyset$ {Agents already matched.}
 4: $j \leftarrow 1$ {Position of the first remaining agent to choose.}
 5: for $i = 1$ to $n$ do
 6:     {Iteration occurring at $d_{\delta(i)}$.}
 7:     if $\delta(i) \notin B$ then
 8:         Let $\pi$ denotes $\Pi(I_{<d_{\delta(i)}})$.
 9:         while $\pi(j) \neq \delta(i)$ do
10:             $M(\pi(j)) \leftarrow best_{\pi(j)}(E_{<d_{\delta(i)}} \setminus A)$.
11:             $A \leftarrow A \cup \{M(\pi(j))\}$.
12:             $B \leftarrow B \cup \{\pi(j)\}$
13:             $j \leftarrow j + 1$.
14:         $M(\delta(i)) \leftarrow best_{\delta(i)}(E_{<d_{\delta(i)}} \setminus A)$.
15:         $A \leftarrow A \cup \{M(\delta(i))\}$.
16:         $B \leftarrow B \cup \{\delta(i)\}$
17:         $j \leftarrow j + 1$.
18: return $M$.
```

Definition 5. *Permutation function Π is prefix-stable (PS), if for each agent i, and for any positions i' and j' such that $\Pi_{i'}(I) = i$ and $j' \leq i'$, $\Pi_{j'}(I) = \Pi_{j'}(I_{<d_i})$ holds.*

The following proposition shows that Algorithm 1 is online and weakly incentive compatible with PS permutation.

Proposition 1. $\star$[4] *Algorithm 1 is both online and WIC if the input permutation function is PS.*

4.1 The Ascending Departure Permutation

The following proposition shows that the ascending departure permutation is a good candidate to be used with Algorithm 1.

Proposition 2. $\star$ *Ascending departure permutation δ is PS.*

Note that when Algorithm 1 is used with the ascending departure permutation δ, the algorithm allows each agent to choose the best remaining item upon leaving the market.

Example 2. *We run Algorithm 1 with the ascending departure permutation δ on the instance of Figure 1. At d_2 (departure of agent 2), she selects item e_1 that she prefers to e_2. Then, at d_3, agent 3 selects item e_2 that she prefers to e_3. Finally, agent 1 selects e_3, which is the only remaining item. The resulting allocation is M' described in Figure 1.*

The following proposition shows that the ascending departure permutation δ leads to a Pareto efficient outcome.

Proposition 3. $\star$ *For any $I \in \mathcal{I}$, Algo. 1 using the ascending departure permutation δ, returns a $\mathcal{M}(I)$-PO allocation.*

The following proposition shows that no other online exchange algorithm always returns a Pareto-efficient outcome.

[4]Propositions marked with $\star$ indicate that proofs are omitted due to space limitations, but are included in the supplementary material

Proposition 4. *Algorithm 1 using the ascending departure permutation δ is the only online exchange algorithm that returns for any instance I an $\mathcal{M}(I)$-PO allocation.*

Proof. By contradiction, let $\mathcal{A}$ denote an online exchange algorithm that behaves differently than Algorithm 1 with the ascending departure permutation δ. This means that there is an instance I and an agent j such that $\mathcal{A}_j(I)$ does not correspond to the assignment made by Algorithm 1 with permutation δ. In other words, agent j does not receive her most favorite item among those in $E_{<d_{\delta(j)}}$ minus the items already assigned to agents leaving earlier than her. Furthermore, we assume w.l.o.g. that, among the set of agents in this situation, agent j is the one who leaves the earliest. Let M denote the allocation obtained by Algorithm 1 with permutation δ applied to I. Let i denote the agent who receives $M(j)$ instead of agent j in $\mathcal{A}(I)$, i.e., such that $\mathcal{A}_i(I) = M(j)$. By assumption, $d_j < d_i$ holds since, otherwise, an agent arriving earlier than agent j would not receive the same item as in M.

To show the contradiction, we construct a new instance I' by adding dummy agents to I. For any agent k leaving after d_j, let k' be a dummy agent such that $a_{k'} = d_j + k\epsilon$, with $\epsilon > 0$ small enough for all these agents to arrive earlier than any other arrival or departure, and $d_{k'}$ large enough to leave after any other agent of I. The preferences of the agents in I' are as follows. For any agent k that was already in I and leaving after d_j, her favorite item is $e_{k'}$, followed by the items of I ranked in the same order, and finally all other items of I' ranked arbitrarily. Dummy agent i''s preferences are $\mathcal{A}_j(I) \succ_{i'} M(j) \succ_{i'} \ldots$, where all the other items are ranked arbitrarily. Finally, the favorite item of other dummy agent k' is $\mathcal{A}_k(I)$, and all other items are ranked arbitrarily.

Note first that since $\mathcal{A}$ is online, both $\mathcal{A}_j(I') = \mathcal{A}_j(I'_{<d_j})$ and $\mathcal{A}_j(I_{<d_j}) = \mathcal{A}_j(I)$ hold. Furthermore, since all dummy agents arrive later than d_j, $I_{<d_j} = I'_{<d_j}$, which implies with the former equalities that $\mathcal{A}_j(I') = \mathcal{A}_j(I)$. Concerning the assignment of the other agents, based on their preferences and since $\mathcal{A}$ is $\mathcal{M}(I')$-PO, we can say that for each dummy agent k', we have $\mathcal{A}_{k'}(I') = \mathcal{A}_k(I)$, and for each regular agent k leaving after d_j, we have $\mathcal{A}_k(I') = e_{k'}$. More precisely for i' we have $\mathcal{A}_{i'}(I') = \mathcal{A}_i(I) = M(j)$. Note that except for agents i' and j, any agent receives her favorite item.

But now consider the almost same allocation, say M', as the one constructed by $\mathcal{A}$, except that agent j receives $M(j)$ and agent i' receives $\mathcal{A}_j(I)$. Note that by construction, M' should belong to $\mathcal{M}(I')$ since $\mathcal{A}$ is an online exchange algorithm, and therefore $\mathcal{A}(I_{<d_j}) = \mathcal{A}(I'_{<d_j})$ belongs to $\mathcal{M}(I'_{<d_j})$ (for agent j and the ones leaving after her), $\mathcal{A}(I')$ belongs to $\mathcal{M}(I')$ (for all other agents except agent i'), and $a_{i'} < d_i$ holds (for agent i'). Agent j and dummy agent i' receive in M' a strictly better item than the one offered by algorithm $\mathcal{A}$. Therefore, M' Pareto dominates $\mathcal{A}(I')$, leading to a contradiction since $\mathcal{A}$ is $\mathcal{M}(I')$-PO. □

Propositions 3 and 4 suggest focusing on Algorithm 1 with input δ when we want an online algorithm that returns an efficient outcome. Furthermore, Proposition 1 and 2 attest that this algorithm is WIC. However, it is easy to check that Algorithm 1 using input δ is neither a-IC, d-IC nor IR.

Corollary 5. *There is no online exchange algorithm that both returns a $\mathcal{M}(I)$-Pareto optimal allocation for any instance $I \in \mathcal{I}$, and is either a-IC, d-IC or IR.*

4.2 The Ascending Arrival Permutation

Another candidate is the ascending arrival permutation α, which ranks the agents by increasing arrival time i.e., such that $a_{\alpha(1)} < a_{\alpha(2)} < \ldots < a_{\alpha(n)}$. The following proposition shows that this permutation α is consistent with Algorithm 1.

Proposition 6. $\star$ *Ascending arrival permutation α is PS.*

Example 3. *We run Algorithm 1 with the ascending arrival permutation α on the instance of Figure 1. At d_2, agent 1 selects first and picks e_2, which she prefers to e_1. Then, agent 2 selects the last remaining item at d_2, which is e_1. Finally, at d_3, agent 3 picks the last remaining item, which is e_3. The resulting allocation is M'' described in Figure 1.*

Furthermore, choosing the ascending arrival permutation α incentivizes agents to truthfully report their departure time.

Proposition 7. $\star$ *Algorithm 1 is online and d-IC if and only if it uses the ascending arrival permutation α as input.*

Note that this statement is not as strong as Proposition 4 since it does not rule out any other d-IC online exchange algorithm. The following proposition rules out the existence of a permutation that achieves a-IC with Algorithm 1.

Proposition 8. $\star$ *There is no online permutation function that makes Algorithm 1 a-IC.*

5 Top Trading Cycle Based Algorithms

We now explore the possibility to extend the TTC procedure to our online problem. This procedure is a natural candidate to achieve the properties mentioned in Section 3, since in the offline context it fulfills all of them, and even stronger requirements [Roth and Postlewaite, 1977; Ma, 1994]. Before providing an extension of TTC to the online setting, we provide a description of the standard TTC procedure in Algorithm 3, which will be used as a subroutine of our online algorithms. This procedure relies on the TTC-graph, which is an oriented graph where agents are vertices. As described in Algorithm 2, each vertex has an out-degree of one, and the single oriented edge starting at the vertex corresponding to an agent i is oriented to the vertex corresponding to agent j whose item e_j is the most favorite according to $\succ_i$ among the ones detained by the agents in the graph. Such a graph should contain at least one cycle, and items are assigned along these cycles. Each agent contained in the cycle is assigned the item belonging to the agent that she is pointing to and leaves the market with this item. The procedure continues repeatedly with the remaining agents and their initial endowment until each agent is assigned an item.

It is well known that the TTC procedure is IR in the offline setting, and any online extension of this algorithm should keep this property. However, concerning the other properties, in the online setting not all agents are present at the beginning of the algorithm, and some of them will not arrive before the departure of other agents. Therefore, Algorithm 3 should be customized to fit the online setting. We explore a procedure

Algorithm 2 TTC-graph construction

```
Input: Subset of agents $N'$ and their initial endowment.
1: $V \leftarrow N'$. {Set of vertices.}
2: $H \leftarrow \bigcup_{i \in N'} \{e_i\}$.{Houses owned by the agents of $N'$.}
3: $E \leftarrow \emptyset$. {Set of edges.}
4: for each $i$ in $N'$ do
5:     $E \leftarrow E \cup (i, best_i(H))$.
6: return $G = (V, E)$.
```

Algorithm 3 Top trading cycle algorithm

```
Input: Subset of agents $N'$ and their initial endowments.
1: while $N' \neq \emptyset$ do
2:     Construct the TTC-graph $G(N')$ using Algorithm 2.
3:     for each cycle $C = (c_1, c_2, \ldots, c_k)$ in $G(N')$ do
4:         for $i = 1$ to $k - 1$ do
5:             $M'(c_i) \leftarrow e_{c_{i+1}}$.
6:         $M'(c_k) \leftarrow e_{c_1}$.
7:         $N' \leftarrow N' \setminus C$.
8: return $M'$.
```

described in Algorithm 4, which applies TTC to subsets of agents already present in the market, according to an input partition function. Partition function $\mathcal{P}$ applied to I will partition the agents of I into multiple subsets. More formally, a partition function $\mathcal{P} : \mathcal{I} \to 2^{2^N}$ maps any instance $I \in \mathcal{I}$ to a partition $\mathcal{P}(I)$ of the agents in I. This means that $\mathcal{P}(I)$ is a collection of subsets of agents such that the union of these subsets is N, and no two subsets share an agent. We assume once again that the result of this function is independent of the preferences of the agents and their initial endowments. To construct the allocation, Algorithm 3 is applied independently to the subsets of agents returned by this partition function. To simplify notations, we assume that $\mathcal{P}_i(I)$ denotes the subset of $\mathcal{P}(I)$ that contains agent i.

The TTC procedure described in Algorithm 4 requires that an agent is not selected multiple times for application of Algorithm 3. This translates into the following property.

Definition 6. *A partition function $\mathcal{P}$ is online compatible (OC) if for any instance I and agent i, $\mathcal{P}_i(I) = \mathcal{P}_i(I_{<d_i})$.*

It is easy to check that if partition function $\mathcal{P}$ is OC, then for any agent i, $\mathcal{P}_i(I)$ contains only agents that are in the market at time d_i, i.e., for any $j \in \mathcal{P}_i(I)$, $a_j < d_i < d_j$ must hold. Furthermore, no agent participates multiple times to the TTC procedure when the partition function used by Algorithm 4 is OC. The following proposition outlines the properties of OC partition functions with respect to Algorithm 4.

Proposition 9. ⋆ *Algorithm 4 using a OC partition function as input is both online and WIC.*

5.1 The Departing Agent Excluded Partition

We explore now if a stronger notion of incentive compatibility can be achieved with particular partition functions. Let us introduce the departing agent excluded partition γ, whose construction is described in Algorithm 5. Intuitively, every time that an unpartitioned agent leaves the market, two new

Algorithm 4 Online top trading cycle procedure

```
Input: Partition function $\mathcal{P}$.
1: $B \leftarrow \emptyset$ {Agents already matched.}
2: for $i = 1$ to $n$ do
3:     {Iteration occurring at $d_{\delta(i)}$.}
4:     if $\delta(i) \notin B$ then
5:         Applies Algorithm 3 to $\mathcal{P}_i(I_{<d_{\delta(i)}})$ to obtain $M'$.
6:         for all $j \in \mathcal{P}_i(I_{<d_{\delta(i)}})$ do
7:             $M(j) \leftarrow M'(j)$.
8:         $B \leftarrow B \cup \mathcal{P}_i(I_{<d_{\delta(i)}})$
9: return $M$.
```

Algorithm 5 Departing agent excluded partition γ

```
Input: Instance $I = \{(i, e_i, a_i, d_i, \succ_i)\}_{i \in N}$.
1: $B \leftarrow \emptyset$. {Set of agents belonging to the partition.}
2: $\gamma \leftarrow \emptyset$ {The partition to construct.}
3: for $i = 1$ to $n$ do
4:     {Iteration occurring at $d_{\delta(i)}$.}
5:     if $\delta(i) \notin B$ then
6:         $\gamma \leftarrow \gamma \cup \{\{\delta(i)\}\} \cup \{N_{<d_{\delta(i)}} \setminus (B \cup \{\delta(i)\})\}$
7:         $B \leftarrow B \cup N_{<d_{\delta(i)}}$
8: return $\gamma$.
```

subsets are added to the partition. The first is a singleton containing the agent leaving the market, and the second is the remaining unpartitioned agents that are present in the market.

Proposition 10. ⋆ *The departing agent excluded partition γ is OC.*

Example 4. *Consider the instance I and the three allocations depicted in Figure 2. It is easy to verify that Algorithm 1 using either the ascending departure permutation δ or the ascending arrival permutation α returns allocation M'', which is not IR since agent 5 receives item e_1.*

Let us first run Algorithm 5 to compute the departing agent excluded partition $\gamma(I)$. At d_1, $N_{<d_1} = \{1, 2, 3\}$. Therefore, $\{1\}$ and $\{2, 3\}$ are added to the partition. At d_2 and d_3, nothing occurs since agents 2 and 3 are already in the partition. At d_4, $N_{<d_4} = N$, but agents 1, 2, and 3 are already in the partition. Therefore, $\{4\}$ and $\{5\}$ are added to the partition, and Algorithm 5 returns $\gamma(I) = \{\{1\}, \{2, 3\}, \{4\}, \{5\}\}$.

Let us now run Algorithm 4 with $\gamma(I)$ as input to compute the allocation for this instance. The TTC procedure is applied only once to a subset of more than one agent, which is $\{2, 3\}$. As depicted in Figure 3a, the TTC-graph for the subset of agents $\{2, 3\}$ contains a single cycle covering all agents, and they swap their goods. The resulting allocation is M.

The idea under this partition function γ is to punish agents, who force the assignment to take place because they leave earlier, by leaving them alone in the partition. The consequence for Algorithm 4 is that they keep their initial endowment without exchange. The following proposition shows more formally that no agent has an incentive to misreport her departure time when Algorithm 4 uses partition function γ.

Proposition 11. *Algorithm 4 using the departing agent excluded partition γ is d-IC.*

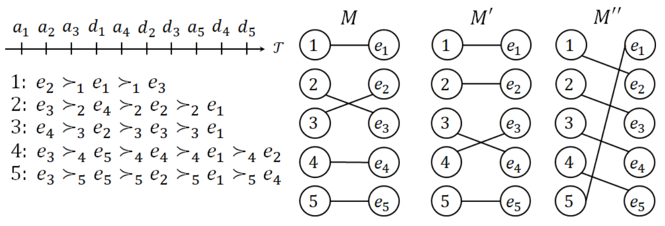
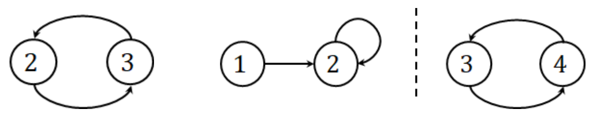

a_1 a_2 a_3 d_1 a_4 d_2 d_3 a_5 d_4 d_5
$\mathcal{T}$
1: $e_2 \succ_1 e_1 \succ_1 e_3$
2: $e_3 \succ_2 e_4 \succ_2 e_2 \succ_2 e_1$
3: $e_4 \succ_3 e_2 \succ_3 e_3 \succ_3 e_1$
4: $e_3 \succ_4 e_5 \succ_4 e_4 \succ_4 e_1 \succ_4 e_2$
5: $e_3 \succ_5 e_5 \succ_5 e_2 \succ_5 e_1 \succ_5 e_4$
M
M'
M''

Figure 2: Example instance (left) and allocations (right).



(a) For $\{2, 3\}$. (b) For $\{1, 2\}$ (left) and $\{3, 4\}$ (right).

Figure 3: TTC-graphs for different instances and subsets of agents.

Proof. We know by Propositions 9 and 10 that Algorithm 4 with partition γ is WIC. It remains to show that no agent has an incentive to misreport her departure time. Let S denote the subset of agents for which the "if" clause of Algorithm 5 is true, i.e., agents that are not partitioned until their departure.

Consider first the case of an agent $i \in S$. Note first that agent i belongs to a singleton in the departing agent excluded partition γ. Let us show that even if agent i declares an earlier departure time d'_i such that $a_i < d'_i < d_i$, then there is no other agent j of S such that $d'_i < d_j < d_i$. Indeed, since $d_j < d_i$, we know that agent j is considered before agent i during the "for" loop of Algorithm 5. In the last instruction of this loop when agent j is considered, $N_{<d_j}$ is added to the set of partitioned agents B. Since agent i is not in B at iteration d_i of the loop, it means that she is not in $N_{<d_j}$, or in other words $a_i > d_j$. This implies with $a_i < d'_i$ that $d_j < d'_i$.

We now know that any agent of S leaving earlier than agent i will still leave earlier than i even after i declares a new departure time d'_i. This implies that agent i will still pass the "if" test in the loop of Algorithm 5 even if she changes her departure time because no agent of S leaving earlier will include i in a partition. Therefore, agent i will be alone in the departing agent excluded partition γ after misreporting her departure time and will keep her item during Algorithm 4. So, she has no incentive to misreport her departure time.

Consider now the case of an agent i not in S. Note first that $i \notin S$ means that there is an agent $j \in S$ such that $d_j < d_i$ and $i \in N_{<d_j}$. We assume that j is the agent of S with the lowest rank in the ascending departure permutation δ for which these inequalities are true. In other words, the subset of agents containing agent i is selected by Algorithm 5 during the iteration of the "for" loop corresponding to d_j. By applying the same reasoning as above, it is easy to check that there is no agent k of S such that $d_k < d_j$ and $d'_i < d_k$. Therefore, the only misreport d'_i of agent i that could change the outcome of Algorithm 5 is such that $d_j > d'_i$. In that case, agent i will belong to the set of agents for which the "if" condition of Algorithm 5 is true, and therefore agent i will be alone in the departing agent excluded partition γ, and will receive her item in Algorithm 4. Since Algorithm 4 is IR, agent i has no incentive to misreport her departure time. □

Algorithm 6 Scheduled departure partition θ.

```
Input: Instance $I$, and schedule $\xi = \{\xi_0, \xi_1, \dots \xi_k\}$.
 1: $A \leftarrow \{0\}$. {Set of intervals of $\xi$ already considered.}
 2: $B \leftarrow \emptyset$. {Set of agents belonging to the partition.}
 3: for $i = 1$ to $n$ do
 4:     {Iteration occurring at $d_{\delta(i)}$.}
 5:     Let $\xi_j$ denote the interval of $\xi$ containing $d_{\delta(i)}$.
 6:     if $j \notin A$ then
 7:         Let $S$ be the subset of agents of $I_{<d_{\delta(i)}}$ whose departure times belong to $\xi_j$.
 8:         $\theta \leftarrow \theta \cup \{S\}$.
 9:         $A \leftarrow A \cup \{j\}$.
10:         $B \leftarrow B \cup S$.
11:     else if $\delta(i) \notin B$ then
12:         $\theta \leftarrow \theta \cup \{\{\delta(i)\}\}$.
13:         $B \leftarrow B \cup \{\delta(i)\}$.
14: return $\theta$.
```

Unfortunately, it is easy to check that Algorithm 4 using the departing agent excluded partition γ is not a-IC.

5.2 Partition Functions Based on Schedules

Let us now examine if a-IC can be achieved by Algorithm 4 using another partition function. To do so, we introduce a subset of non-overlapping time intervals $\xi = \{\xi_0, \xi_1, \dots, \xi_k\}$, that we call schedule. More formally, we assume that for any $j \in \{0, 1, \dots, k\}$, $\xi_j \subseteq \mathcal{T}$, and for any $j' \neq j$, $\xi_j \cap \xi_{j'} = \emptyset$. We assume that each interval is contiguous, potentially with the exception of ξ_0, which covers all time intervals not contained in the other intervals. This schedule will be used to initialize the partition by grouping agents whose departure times belong to the same time interval of ξ.

The construction of the scheduled departure partition θ is described in Algorithm 6, and depends on both the arrival and departure times of the agents, and the input schedule ξ. Intuitively, all agents whose departure times belong to the same time interval and whose arrival times are no later than the earliest departure time of one of these agents are grouped together, whereas the other agents are left alone in the partition. Interval ξ_0 plays a special role since all the agents whose departure times belong to this interval are left alone in the partition. Note that this first interval may be left empty. To simplify notation, we denote by θ the scheduled departure partition resulting from Algorithm 6, without specifying the schedule used. The following proposition shows this the partition function is online compatible.

Proposition 12. $\star$ *Scheduled departure partition θ is OC.*

Example 5. *Consider once again instance I described in Figure 2, and schedule ξ represented in Figure 4. Note that ξ_1 contains the departure time of agents 1 and 2, and ξ_2 contains the departure time of agents 3, 4, and 5. Let us run Algorithm 6 to compute the scheduled departure partition $\theta(I)$. At d_1, both agents 1 and 2 are in $N_{<d_1}$. Therefore, $\{1, 2\}$ is added to the partition. At d_2, nothing happens since agent 2 already belongs to the partition. At d_3, both agents 3 and 4 are*

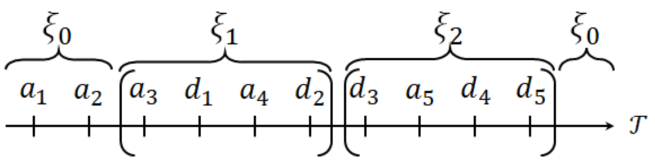


Figure 4: Graphical representation of schedule ξ.

in $N_{<d_3}$. Therefore, $\{3,4\}$ is added to the partition. At d_4, nothing happens since agent 4 is already part of the partition. At d_5, the interval ξ_2 that contains d_5 has already been considered. Therefore, agent 5 remains alone, and $\{5\}$ is added to the partition. Finally, the algorithm halts and returns the scheduled departure partition $\theta(I) = \{\{1,2\},\{3,4\},\{5\}\}$.

Let us now run Algorithm 4 using $\theta(I)$ as input to compute the allocation. The TTC procedure is applied independently to $\{1,2\}$ and $\{3,4\}$. The TTC-graphs during the first iteration of the TTC procedure are represented in Figure 3b. The TTC-graph for the subset of agents $\{1,2\}$ (left part of Figure 3b) contains a single cycle, which is the self-loop involving agent 2. Therefore, agent 2 keeps her good, and agent 1, who remains alone during the second iteration, also keeps her good. The TTC-graph for $\{3,4\}$ (right part of Figure 3b) contains a single cycle that includes both agents, who swap their goods. The resulting matching is M' of Figure 2.

Note that if agents i and j have their departure times belonging to the same time interval and one of them, say i, leaves before the arrival of the other, then agent j is left alone in the scheduled departure partition θ. That was the case for agent 5 in Example 5. The underlying idea is that the set containing agent i in the partition θ must be decided before her departure, and agent j, who arrives later, cannot be included in the same set. Additionally, agent j cannot be partitioned with other agents to avoid incentivizing her to strategically delay her arrival time. The following proposition shows that no agent has an incentive to misreport her arrival time with the partition function returned by Algorithm 6.

Proposition 13. $\star$ *Algorithm 4 using the scheduled departure partition θ is a-IC.*

Unfortunately, Algorithm 4 using the scheduled departure partition θ is not d-IC, with the exception of the schedule containing only ξ_0. In that case, $\xi_0 = T$ holds and the resulting partition is the one where each agent is alone. Algorithm 4 will therefore assign to each agent her own item. This algorithm is obviously SIC since its outcome does not change with the preferences and the arrival and departure time of the agents. However, this algorithm is not really interesting since no exchanges among agents are performed, and the allocation is not improved. By combining the properties of Algorithms 5 and 6, we can create a less trivial partition function that renders Algorithm 4 SIC. To do so, we can use Algorithm 6 as a baseline, and modify line 8 by the assignment $\theta \leftarrow \theta \cup \{\{\delta(i)\}\} \cup \{S \setminus \{\delta(i)\}\}$, which is consistent with line 5 of Algorithm 5. Furthermore, we restrict the input schedule to $\{\xi_0, \xi_1\}$ such that $\xi_0 = \emptyset$ and $\xi_1 = T$. The resulting partition function is called earliest departure partition ζ. In other words, this partition ζ groups together all the agents of $I_{<\delta(1)}$, with the exception of agent 1, and all the other agents stay alone. This partition ζ is easily shown to be OC due to its similarity in construction to Algorithms 5 and 6.

Example 6. *Consider one last time instance I described in Figure 2. Let us compute the earliest departure partition $\zeta(I)$ by running the modified version of Algorithm 6 described in the previous paragraph. At d_1, during the earliest departure time of an agent, $N_{<d_1}$ contains agents 1, 2, and 3. Therefore, $\{1\}$ and $\{2,3\}$ are added to the partition. All the other agents will remain alone in the partition. Thus, the algorithm halts and returns the earliest departure partition $\zeta(I) = \{\{1\},\{2,3\},\{4\},\{5\}\}$, which is identical to the departing agent excluded partition $\gamma(I)$. Therefore, the matching returned by Algorithm 4 is M depicted in Figure 2.*

Proposition 14. $\star$ *Algorithm 4 using the earliest departure partition ζ is SIC.*

The main drawback of the earliest departure partition ζ is that only one coalition contains more than one agent, and the exchanges performed during the TTC procedure may involve few agents. Their number depends on the agents present in the market when the first agent leaves and ranges from 1 (if $a_2 > d_1$) to $n-1$ (if $a_n < d_{\delta(1)}$).

6 Conclusion and Future Works

We extended the serial dictatorship and TTC procedures to an online setting, aiming to develop mechanisms that are Pareto-efficient, individually rational, and incentivize agents to truthfully reveal their preferences, as well as their actual arrival and departure times. Several variants of these mechanisms were proposed and are summarized in Table 1 along with their respective properties. The paper also presents additional results not summarized in Table 1. For example, we proved that Algorithm 1 using the ascending departure permutation δ is the only mechanism always returning a $\mathcal{M}(I)$-PO allocation.

	$\mathcal{M}$-PO	IR	WIC	a-IC	d-IC
Alg. 1 with δ	Prop. 3		Prop. 1		
Alg. 1 with α			Prop. 1		Prop. 7
Alg. 4 with γ		$\surd$	Prop. 9		Prop. 11
Alg. 4 with θ		$\surd$	Prop. 9	Prop. 13	
Alg. 4 with ζ		$\surd$	Prop. 9	Prop. 14	Prop. 14

Table 1: Summary of the different mechanisms and their properties.

An intriguing extension of this work would be to explore dynamic versions of these mechanisms, where the allocation of an agent remaining in the market could evolve even after an item has been assigned to her. Another promising direction is the design of a strongly incentive-compatible mechanism that enables more exchanges than Algorithm 4 with the earliest departure partition ζ. Lastly, we could investigate relaxed forms of efficiency and strategy-proofness, potentially incorporating probabilistic assumptions about arrival patterns. Such probabilistic assumptions are closely related to the online stochastic matching problem [Feldman *et al.*, 2009; Huang and Shu, 2021]. A similar approach has also been explored for the fair assignment of public goods [Banerjee *et al.*, 2022; Banerjee *et al.*, 2023].

References

[Albers, 2003] Susanne Albers. Online algorithms: a survey. *Mathematical Programming*, 97:3–26, 2003.

[Aleksandrov and Walsh, 2019] Martin Aleksandrov and Toby Walsh. Strategy-proofness, Envy-freeness and Pareto Efficiency in Online Fair Division with Additive Utilities. In *Proceedings of the Sixteenth Pacific Rim International Conference on Artificial Intelligence, PRICAI'19*, pages 527–541, 2019.

[Aleksandrov and Walsh, 2020] Martin Aleksandrov and Toby Walsh. Online Fair Division: A Survey. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI'20*, pages 13557–13562, 2020.

[Aleksandrov *et al.*, 2015] Martin Aleksandrov, Haris Aziz, Serge Gaspers, and Toby Walsh. Online Fair Division: Analysing a Food Bank Problem. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI'15*, pages 2540–2546, 2015.

[Anderson *et al.*, 2017] Ross Anderson, Itai Ashlagi, David Gamarnik, and Yash Kanoria. Efficient Dynamic Barter Exchange. *Operations Research*, 65(6):1446–1459, 2017.

[Ashlagi and Roth, 2021] Itai Ashlagi and Alvin E. Roth. Kidney Exchange: An Operations Perspective. *Management Science*, 67(9):5455–5478, 2021.

[Ashlagi *et al.*, 2015] Itai Ashlagi, Felix Fischer, Ian A. Kash, and Ariel D. Procaccia. Mix and match: A strategyproof mechanism for multi-hospital kidney exchange. *Games and Economic Behavior*, 91(C):284–296, 2015.

[Awasthi and Sandholm, 2009] Pranjal Awasthi and Tuomas Sandholm. Online Stochastic Optimization in the Large: Application to Kidney Exchange. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence, IJCAI'09*, pages 405–411, 2009.

[Baccara and Yariv, 2021] Mariagiovanna Baccara and Leeat Yariv. Dynamic matching. In *Online and Matching-Based Market Design*, pages 1221–1278. 2021.

[Baccara *et al.*, 2020] Mariagiovanna Baccara, SangMok Lee, and Leeat Yariv. Optimal dynamic matching. *Theoretical Economics*, 15(3):1221–1278, 2020.

[Banerjee *et al.*, 2022] Siddhartha Banerjee, Vasilis Gkatzelis, Artur Gorokh, and Billy Jin. Online Nash Social Welfare Maximization with Predictions. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'22*, pages 1–19, 2022.

[Banerjee *et al.*, 2023] Siddhartha Banerjee, Vasilis Gkatzelis, Safwan Hossain, Billy Jin, Evi Micha, and Nisarg Shah. Proportionally Fair Online Allocation of Public Goods with Predictions. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI'23*, pages 20–28, 2023.

[Biró, 2017] Péter Biró. Applications of Matching Models under Preferences. In U. Endriss, editor, *Trends in Computational Social Choice*, pages 345–373. AI Access, 2017.

[Bloch and Cantala, 2017] Francis Bloch and David Cantala. Dynamic Assignment of Objects to Queuing Agents. *American Economic Journal: Microeconomics*, 9(1):88–122, 2017.

[Bouveret and Lang, 2014] Sylvain Bouveret and Jérôme Lang. Manipulating picking sequences. In *Proceedings of the Twenty-First European Conference on Artificial Intelligence, ECAI'14*, pages 141–146, 2014.

[Chen and Sönmez, 2002] Yan Chen and Tayfun Sönmez. Improving Efficiency of On-Campus Housing: An Experimental Study. *American Economic Review*, 92(5):1669–1686, 2002.

[Dickerson *et al.*, 2012] John P Dickerson, Ariel D Procaccia, and Tuomas Sandholm. Dynamic Matching via Weighted Myopia with Application to Kidney Exchange. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, AAAI'12*, pages 1340–1346, 2012.

[Feldman *et al.*, 2009] Jon Feldman, Nitish Korula, Vahab S. Mirrokni, S. Muthukrishnan, and Martin Pál. Online Ad Assignment with Free Disposal. In *Proceedings of the Fifth International Workshop on Internet and Network Economics, WINE'09*, pages 374–385, 2009.

[Ferrari *et al.*, 2015] Paolo Ferrari, Willem Weimar, Rachel J Johnson, Wai H Lim, and Kathryn J Tinckam. Kidney paired donation: principles, protocols and programs. *Nephrology Dialysis Transplantation*, 30(8):1276–1285, 2015.

[Gerding *et al.*, 2019] Enrico H. Gerding, Alvaro Perez-Diaz, Haris Aziz, Serge Gaspers, Antonia Marcu, Nicholas Mattei, and Toby Walsh. Fair Online Allocation of Perishable Goods and its Application to Electric Vehicle Charging. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI'19*, pages 5569–5575, 2019.

[Hajaj *et al.*, 2015] Chen Hajaj, John P. Dickerson, Avinatan Hassidim, Tuomas Sandholm, and David Sarne. Strategy-Proof and Efficient Kidney Exchange Using a Credit Mechanism. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI'15*, pages 921–928, 2015.

[Hosseini *et al.*, 2021] Hadi Hosseini, Vijay Menon, Nisarg Shah, and Sujoy Sikdar. Necessarily Optimal One-Sided Matchings. In *Proceedings of the AAAI Conference on Artificial Intelligence, AAAI'21*, pages 5481–5488, 2021.

[Hosseini *et al.*, 2024] Hadi Hosseini, Zhiyi Huang, Ayumi Igarashi, and Nisarg Shah. Class Fairness in Online Matching. *Artificial Intelligence*, 335:104177, 2024.

[Huang and Shu, 2021] Zhiyi Huang and Xinkai Shu. Online Stochastic Matching, Poisson Arrivals, and the Natural Linear Program. In *Proceedings of the Fifty-Third Annual ACM SIGACT Symposium on Theory of Computing, STOC'21*, page 682–693, 2021.

[Hurwicz and Reiter, 2006] Leonid Hurwicz and Stanley Reiter. *Designing Economic Mechanisms*. Cambridge University Press, 2006.

[Klaus *et al.*, 2016] Bettina Klaus, David F. Manlove, and Francesca Rossi. Matching under Preferences. In Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia, editors, *Handbook of Computational Social Choice*, page 333–355. Cambridge University Press, 2016.

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

[Manlove, 2013] David F. Manlove. *Algorithmics of Matching Under Preferences*. World Scientific, 2013.

[Mattei *et al.*, 2017] Nicholas Mattei, Abdallah Saffidine, and Toby Walsh. Mechanisms for Online Organ Matching. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI'17*, pages 345–351, 2017.

[Roth and Postlewaite, 1977] Alvin Roth and Andrew Postlewaite. Weak versus strong domination in a market with indivisible goods. *Journal of Mathematical Economics*, 4(2):131–137, 1977.

[Sankar *et al.*, 2021] Govind S. Sankar, Anand Louis, Meghana Nasre, and Prajakta Nimbhorkar. Matchings with Group Fairness Constraints: Online and Offline Algorithms. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI'21*, pages 377–383, 2021.

[Shapley and Scarf, 1974] Lloyd Shapley and Herbert Scarf. On cores and indivisibility. *Journal of Mathematical Economics*, 1(1):23–37, 1974.

[Swapz, 2025] Swapz. Swapz: The UK's Number One Swapping Marketplace. https://www.swapz.co.uk/, 2025.

[Thomson, 2011] William Thomson. Fair Allocation Rules. In Kenneth J. Arrow, Amartya Sen, and Kotaro Suzumura, editors, *Handbook of Social Choice and Welfare*, volume 2, pages 393–506. Elsevier, 2011.

[Ünver, 2010] M. Utku Ünver. Dynamic Kidney Exchange. *The Review of Economic Studies*, 77(1):372–414, 2010.