An Optimal Procedure to Check Pareto-Optimality for House Market with Single-Peaked Preferences

Aurélie Beynier[a], Nicolas Maudet[a], Simon Rey[b], Parham Shams[a]

[a]*LIP6, Sorbonne Université, Paris, France*
[b]*ILLC, University of Amsterdam, Amsterdam, the Netherlands*

Abstract

Recently, the problem of allocating one resource per agent with initial endowments (*house markets*) has seen a renewed interest: indeed, while in the domain of strict preferences the Top Trading Cycle algorithm (Shapley and Scarf, 1974) is known to be the only procedure guaranteeing Pareto-optimality, individual rationality, and strategy proofness (Ma, 1994). However, the situation differs in the single-peaked domain. Indeed, Bade (2019) presented the *Crawler*, an alternative procedure enjoying the same properties, with the additional advantage of being implementable in obviously dominant strategies. In this paper we further investigate the Crawler and propose the Diver, a variant which checks optimally whether an allocation is Pareto-optimal for single-peaked preferences, thus improving over known techniques used for checking Pareto-optimality in more general domains. We also prove that the Diver is asymptotically optimal in terms of communication complexity.

1. Introduction

Allocating indivisible resources among a set of agents is a research agenda that has been extensively studied in the recent years. It is a particularly dynamic field in both artificial intelligence (e.g. Brandt et al. (2016)) and economics (e.g. Moulin (2018)). It investigates the issue of fairly and/or efficiently allocating a set of objects to a set of agents while taking into account their preferences.

In the present paper we focus on the model defined by Shapley and Scarf (1974), called *house market* or *assignment problem*, in which there are exactly as many indivisible resources as agents and where each agent should receive exactly one resource. In house markets, Shapley and Scarf (1974) defined the *Top Trading Cycle* (TTC) procedure which has been extensively studied (Roth, 1982). When all preferences expressed as strict linear orders are allowed, the TTC procedure is known to satisfy Pareto-optimality (it is not possible to improve some agents' satisfaction without hurting some others'), strategy-proofness (no one can benefit from reporting non truthful preferences) and individual rationality (no agent receives a house that she likes less than her initial endowment). It is also provably the only procedure enjoying such properties (Ma, 1994) when preferences are strict.

However, preferences frequently exhibit some structures. The domain of *single-peaked* preferences, initially introduced by Black (1948) and Arrow (1951) for voting scenarios, is one of the most studied preference domains (Moulin, 1991; Elkind et al., 2017). It states that there is a common linear order such that all the preferences are decreasing when moving away from the most preferred resource following the order. This domain is also relevant in resource allocation settings (Sprumont, 1991; Bruner and Lackner, 2015). For instance, agents may be looking for houses in a street which has a metro station at one of its ends, and a bike rental platform at the other end; the agents' preferences are then likely to be single-peaked depending on the distance to their favourite means of transportation. However, until recently, only a few papers had studied this domain restriction in house markets. Damamme et al. (2015) investigated (distributed) swap dynamics in such settings. Motivated by the allocation of time-slots, Hougaard et al. (2014) and Aziz et al. (2017) studied deterministic and probabilistic solutions for the problem of assigning objects to a line, the domain being more restrictive than single-peaked preferences in that case.

Recently, Bade (2019) presented the *Crawler* procedure for the assignment problem with single-peaked preferences. This procedure differs from TTC and also satisfies Pareto-optimality, strategy-proofness and individual rationality. As a matter of fact, it is also implementable using *obviously dominant strategies* (Li, 2017) when TTC can not, thus demonstrating the benefit of considering the Crawler in the single-peaked domain restriction.

In this note we define a Crawler-based procedure for checking Pareto-optimality of a given allocation in the single-peaked domain, more efficiently. We first provide a brief description of the Crawler (Section 3), and we analyze its complexity. We then introduce a variant of this procedure, which we call the *Diver* that can be used to check Pareto-optimality of a given allocation in linear time (Section 4). This improves over known results which resort on cycle detection techniques and thus run in $O(n^2)$ (Abraham et al., 2005). The procedure also turns out to be optimal in terms of time and communication complexity.

2. Preliminaries

We consider a set of agents $\mathcal{N} = \{a_1, \ldots, a_n\}$ and a set of resources $\mathcal{R} = \{r_1, \ldots, r_n\}$ of the same size. An allocation $\pi = \langle \pi_{a_1}, \ldots, \pi_{a_n} \rangle$ is a vector of $\mathcal{R}^n$ whose components $\pi_{a_i} \in \mathcal{R}$ represent the single resource allocated to agent $a_i \in \mathcal{N}$.

Agents are assumed to express their preferences over the resources through complete linear orders. Agent a_i's preferences are denoted by $\succ_i$, where $r_1 \succ_i r_2$ means that r_1 is strictly preferred over r_2. A preference profile $L = \langle \succ_i \rangle_{a_i \in \mathcal{N}}$ is then a tuple of all the agents' preferences.

For a given linear order $\succ$, we use $top(\succ)$ to denote the top-ranked resource. Similarly, $snd(\succ)$ refers to the second most preferred resource in $\succ$. With a slight abuse of notation we will write $top(a_i)$ and $snd(a_i)$ to refer to $top(\succ_i)$ and $snd(\succ_i)$. When it is not clear from the context we will subscript these notations to specify the resource set considered. For instance $top_R(a_i)$ is the most preferred resource for agent a_i among the resources in $R \subseteq \mathcal{R}$.

An instance of a resource allocation problem is then a tuple $I = \langle \mathcal{N}, \mathcal{R}, L, \pi^0 \rangle$ composed of a set of agents $\mathcal{N}$, a set of resources $\mathcal{R}$, a preference profile L and an initial allocation π^0.

In some settings, natural properties of the agents' preferences can be identified, thus restricting the set of possible preference orderings. The notion of preference domain formalizes these restrictions. For a set of resources $\mathcal{R}$, we denote by $\mathcal{L}_{\mathcal{R}}$ the set of all linear orders over $\mathcal{R}$. Any subset $D \subseteq \mathcal{L}_{\mathcal{R}}$ is then called a preference domain.

We say that an instance $I = \langle \mathcal{N}, \mathcal{R}, L, \pi^0 \rangle$ is defined over a preference domain D if the preferences of the agents belong to D.

In this note, we consider the single-peaked domain. In this setting, the agents are assumed to share a common axis $\lhd$ over the resources and with respect to which their preferences are defined.

Definition 1. *Let $\mathcal{R}$ be a set of resources and $\lhd$ a linear order (i.e. the axis) over $\mathcal{R}$. We say that a linear order $\succ$ is single-peaked with respect to $\lhd$ if we have:*

$$\forall (r_1, r_2) \in \mathcal{R}^2 \ s.t. \ \left. \begin{array}{l} r_2 \lhd r_1 \lhd top(\succ), \\ or, \ top(\succ) \lhd r_1 \lhd r_2 \end{array} \right\} \Rightarrow r_1 \succ r_2.$$

In other words, $\succ$ is single-peaked over $\lhd$ if $\succ$ is decreasing on both left and right sides of $top(\succ)$, where left and right are defined by $\lhd$.

For a given linear order $\lhd$, we call $\mathcal{SP}_{\lhd}$ the set of all the linear orders single-peaked with respect to $\lhd$:

$$\mathcal{SP}_{\lhd} = \{\succ \ \in \mathcal{L}_{\mathcal{R}} \mid \ \succ \text{ is single-peaked w.r.t. } \lhd\}.$$

A preference domain D is called single-peaked if and only if $D \subseteq \mathcal{SP}_{\lhd}$ for a given $\lhd$. An instance I is said to be single-peaked if it is defined over a single-peaked preference domain.

Pareto optimality of the outcome guarantees that no agent can improve her allocation without incurring a loss on at least another agent, while individual rationality guarantees agents have incentive to participate.

Definition 2 (Pareto-optimality). *Let $I = \langle \mathcal{N}, \mathcal{R}, L, \pi^0 \rangle$ be an instance. An allocation π is said to be Pareto-optimal if there is no other allocation π' such that for every agent $a_i \in \mathcal{N}$ either $\pi'_{a_i} \succ \pi_{a_i}$ or $\pi'_{a_i} = \pi_{a_i}$ and there exists at least one agent $a_j \in \mathcal{N}$ such that $\pi'_{a_j} \succ \pi_{a_j}$.*

If such allocation π' exists, we say that π' Pareto-dominates the allocation π.

Definition 3 (Individual rationality). *For a given instance $I = \langle \mathcal{N}, \mathcal{R}, L, \pi^0 \rangle$, an allocation π is individually rational if for every agent $a_i \in \mathcal{N}$ we have either $\pi_{a_i} \succ \pi^0_{a_i}$ or $\pi_{a_i} = \pi^0_{a_i}$.*

We illustrate these two concepts on a simple example.

Example 1. *Let us consider the following instance with 5 agents and 5 resources. The preferences, presented below, are single-peaked with respect to $r_1 \lhd r_2 \lhd r_3 \lhd r_4 \lhd r_5$. The initial allocation $\pi^0 = \langle r_5, r_1, r_3, r_4, r_2 \rangle$ is defined by the underlined resources.*

$$a_1 : \boxed{r_1} \succ_1 r_2 \succ_1 r_3 \succ_1 r_4 \succ_1 \underline{r_5}$$
$$a_2 : \boxed{r_5} \succ_2 r_4 \succ_2 r_3 \succ_2 r_2 \succ_2 \underline{r_1}$$
$$a_3 : \boxed{r_3} \succ_3 r_2 \succ_3 r_1 \succ_3 r_4 \succ_3 r_5$$
$$a_4 : \boxed{r_4} \succ_4 r_3 \succ_4 r_2 \succ_4 r_1 \succ_4 r_5$$
$$a_5 : r_4 \succ_5 r_5 \succ_5 r_3 \succ_5 \boxed{\underline{r_2}} \succ_5 r_1$$

The allocation π^0 is not Pareto-optimal as it is Pareto-dominated by the squared allocation $\boxed{\pi} = \langle r_1, r_5, r_3, r_4, r_2 \rangle$. Note that the allocation $\pi' = \langle r_1, r_5, r_3, r_2, r_4 \rangle$ would make every agent having at least their third top resource, but would violate individual rationality for agent a_4.

3. The Crawler

In house markets under single-peaked preferences, Bade (2019) recently introduced the Crawler procedure. The agents are initially ordered along the single-peaked axis according to the resource they initially hold. The first agent is the one holding the resource on the left side of the axis and the last agent is the one holding the resource on the right side of the axis. We denote by R the list of available resources ordered according to the single-peaked axis and N the list of available agents such as the ith agent of the list is the one who holds the ith resource in R.

The algorithm then screens the agents from left to right[1] and check, for each agent a_i, where the peak $top_R(a_i)$ of a_i is:

1. If $top_R(a_i)$ is on her right, the algorithm moves to the next agent on the right. The agent is said to "pass".
2. If a_i holds her peak $top_R(a_i)$, then $top_R(a_i)$ is allocated to a_i, a_i and $top_R(a_i)$ are then removed from N and R. The algorithm restarts screening the agents from the left extremity of the axis.

---

[1]Note that the algorithm can equivalently be executed from right to left.

Algorithm 1: The Crawler procedure

Input: An instance $I = \langle \mathcal{N}, \mathcal{R}, L, \pi^0 \rangle$
      single-peaked with respect to $\lhd$
Output: An allocation π

1   $\pi \leftarrow$ empty allocation
2   $R \leftarrow \mathcal{R}$: list of resources sorted accordingly to $\lhd$
3   $N \leftarrow \mathcal{N}$: list of agents such that the ith agent is the
    one who initially holds the ith resource in R
4   **while** $N \neq \emptyset$ **do**
5      $t^* \leftarrow |N|$
6   **for** $t = 0$ *to* $|N| - 1$ **do**
7   **if** $r_t \succ_t r_{t+1}$ **then** /* no crawl   */
8             $t^* \leftarrow t$
9   Break
10   **end**
11   **end**
12      $r \leftarrow top_R(a_{t^*})$
13      $pick(a_{t^*}, r, N, R, \pi)$
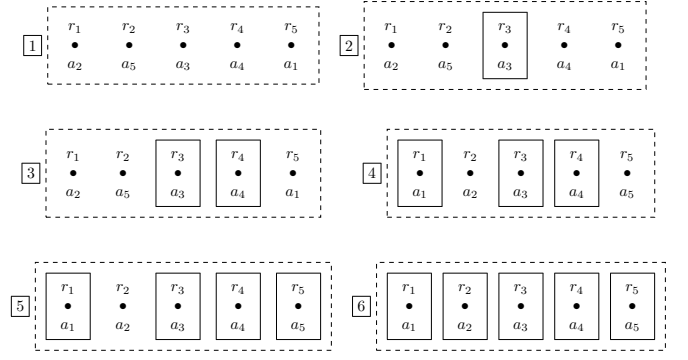14   **end**
15   **return** π

Figure 1: The Crawler procedure run on Example 1. Each dashed box corresponds to a step and a pair resource agent is boxed if the resource has been allocated to the agent.

3. If $top_R(a_i)$ is on the left of a_i, the agent is allocated her peak $top_R(a_i)$. Let t^* be the index of $top_R(a_i)$ and t the index of a_i (we have $t^* < t$). Then, all the agents between t^* and $t - 1$ receive the resource held by the agent on their right (the resources "crawl" towards left). a_i and $top_R(a_i)$ are then removed from N and R. The algorithm restarts screening the agents from the left extremity of the axis.

The algorithm terminates once N, and thus R, are empty.

A formal description of the procedure is given in Algorithm 1. Note that we make use of the sub-procedure $pick(a_{t^*}, r, N, R, \pi)$ which simply assigns the resource r to the given agent a_{t^*} in the allocation π, and then removes the agent and the resource from the lists of available agents and resources, N and R respectively. Since the list of resources is ordered following the single-peaked axis and the ith agent in N corresponds to the owner of the ith resource in R, the removal of r and a_{t^*} is in fact equivalent to assigning r to agent a_{t^*} and crawling the resources from right to left.

Let us illustrate the execution of the Crawler on the instance of Example 1.

Example 2. *Let us return to Example 1. The execution of the Crawler is depicted in Figure 1. The initial allocation is presented in the box 1. Initially, agent a_3 is the first agent whose top is not on her right, she thus receives her top r_3 (box 2). The second step matches agent a_4 to r_4 (box 3). On the third step, agents a_2 and a_5 both have their top on the right but the last agent a_1 has her top on her left, she is then matched to her top r_1 (box 4). Agent a_5 is matched to r_5 on the fourth step (box 5). Finally a_2 is assigned resource r_2 (box 6).*

As observed by Bade (2019), the Crawler always terminates. It is easy to see that it runs in quadratic time.

Proposition 1. *The Crawler procedure terminates and its*

complexity is in $\mathcal{O}(n^2)$ where n is the number of agents and objects.

Proof. Termination is proved by observing that $|N|$ is strictly decreasing at each step of the main while loop. This loop is applied at most n times and each step of the loop requires at most $\mathcal{O}(n)$ elementary operations. The time complexity is then in $\mathcal{O}(n^2)$. $\qquad\square$

We conclude this section by studying the communication requirement of the procedure (Kushilevitz and Nisan, 1996). We are here interested in the amount of information communicated from the agents to the center.

Proposition 2. *The crawler requires at most $n(n + 1)/2 + n \log n$ bits of communication.*

Proof. The crawler runs in n rounds. At each round i, every remaining agent is being asked whether she wishes to pass (answered using 1 bit) or to designate a resource on the left she wants to get (answered in at most $\log n$ bits). Overall, for the i-th round, at most $n - i$ agents will pass, and clearly only one agent designates a resource. Thus the protocol requires in the worst case $\sum_{i=1}^n [(n - i) + \log n] = n(n + 1)/2 + n \log n$, which is in $\mathcal{O}(n^2)$. $\qquad\square$

Since communicating the full preference lists requires $\mathcal{O}(n^2 \log n)$, this protocol does save some communication, even asymptotically, compared to the naive protocol.

4. Optimally checking Pareto-optimaility: the Diver

We now turn to the question of whether we can gain advantage from the single-peaked domain in order to check Pareto-optimality more efficiently. Recall that in the domain allowing any linear order, this can be done in quadratic time (Abraham et al., 2005).

More formally, given an instance $I = \langle \mathcal{N}, \mathcal{R}, L, \pi^0 \rangle$ single-peaked with respect to $\lhd$, we consider the problem CHECKPO whose answer is yes if and only if π^0 is Pareto-optimal. We assume that $\lhd$ is known by the agents. Without loss of generality, we consider that $\forall a_i \in \mathcal{N}, \pi^0_{a_i} = r_i$.

First, observe that the Crawler indeed returns the initial allocation when it is Pareto-optimal.

Observation 1. *Let π be an allocation, π is Pareto-optimal if and only if the Crawler returns π when applied to π as the initial allocation.*

Proof. Any allocation π returned by the Crawler is Pareto-optimal (Bade, 2019). Thus, if π is not Pareto-optimal a different allocation is returned. On the contrary, if π is Pareto-optimal, and since the Crawler is individually rational, no trading cycle will be performed during the execution. The Crawler thus returns π. □

However, this procedure does not enjoy better complexity guarantees than the ones not specific to single-peaked domains, as its worst-case time complexity is in $\mathcal{O}(n^2)$. A worst-case instance can be described as follows: suppose that all the agents (ordered from left to right), have the next resource on their right as their top, except for the last one who likes her own resource. In that case, at each step, the Crawler would go through all the agents before realizing that the last one wants to keep her resource.

We propose a variant of the Crawler, called the *Diver*, which allows to check Pareto-optimality of the initial allocation more efficiently. The key difference with the Crawler is that the *Diver* procedure does not start a new screening once an agent picks a resource: it only checks whether the last agent who was happy to crawl for this resource now agrees to *dive* to the next one. The *Diver* thus proceeds in a *single screening* of the agents.

At each step, the central entity asks the agent whether she wishes to:

(1) pick her current resource;
(2) pass (expressing that she is happy to dive to the next resource); or
(3) pick a smaller resource.

Note that, each time an agent picks a resource, the central entity communicates this information to the other remaining agents so that they can update their list of available resources.

In case (1), the agent (and her resource) are removed and we enter a sub-protocol called *backtrack-call* in which the previous agents are asked one by one whether they still agree to dive to the next resource. This sub-protocol stops as soon as one agent says yes, or when there are no more agents left to consider. All the agents who said 'no' pick their current resources and are themselves removed together with their resource.

Whenever an agent is happy to dive to the next resource, the diver simply moves on to the next agent. This is case (2).

As soon as an agent says she wants a smaller resource, the protocol stops and returns 'not PO'. This corresponds to case (3). Note that in this case, we have the guarantee that there is indeed a better resource available, otherwise the agent would have picked her own resource.

If the screening goes through all the agents, then all the agents left the protocol with their own resource, and the protocol returns 'PO'.

The protocol is formally described in Algorithm 2. The sub-procedure $pick(a_i, r)$ simply assigns resource r to agent a_i, while $pick(a_i, r, D)$ does the same, and removes agent a_i from the list D of agents who crawl or dive.

Example 3. *Coming back to Example 1, by applying the Diver to the initial allocation π^0, the agents are first sorted as follows:*

Algorithm 2: The Diver procedure

Input: An instance $I = \langle \mathcal{N}, \mathcal{R}, L, \pi^0 \rangle$
single-peaked with respect to $\lhd$

Output: PO if π^0 is Pareto-optimal and not PO otherwise

1   $\pi \leftarrow$ list of pairs (a_i, r_i) such that agent a_i holds resource r_i in π^0, sorted according to $\lhd$ for the resources

2   $D \leftarrow \emptyset$: list of agents who crawl or dive

3   **for** (a_i, r_i) *in* π **do**

4   **if** $top_R(a_i) = r_i$ **then** /* pick your top   */

5       $pick(a_i, r_i)$

6   **for** a_j *in* $reverse(D)$ **do**

7   **if** $r_j \succ_{a_j} r_{i+1}$ **then** /* if you don't dive, pick your resource   */

8           $pick(a_j, r_j, D)$

9   **else**

10   Break

11   **end**

12   **end**

13   **else if** $r_i \succ_{a_i} r_{i+1}$ **then** /* your top is on your left: not PO   */

14   **return** not PO

15   **else** /* crawl   */

16       $D \leftarrow D.append(a_i)$

17   **end**
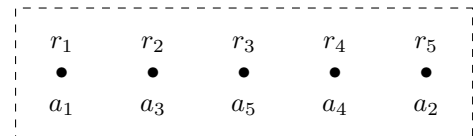
18   **end**

19   **return** PO
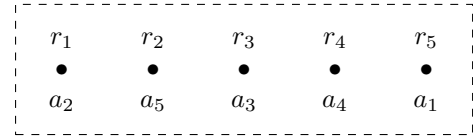
The Diver screens the agent from left to right and asks each agent her wish:

1. a_2 *passes;*
2. a_5 *passes;*
3. a_3 *picks her current resource, a_5 still agrees to pass;*
4. a_4 *picks her current resource, a_5 still agrees to pass;*
5. a_1 *wants to pick a smaller resource (r_1) $\to$ the Diver returns 'not PO'.*

Indeed, this allocation is dominated by $\langle r_1, r_2, r_3, r_4, r_5 \rangle$.

Now let us consider the allocation $\pi = \langle r_1, r_5, r_2, r_4, r_3 \rangle$ leading to the following order:

Again, the Diver screens the agent from left to right and asks each agent her wish:

1. a_1 *picks her current resource;*
2. a_3 *passes;*
3. a_5 *passes;*
4. a_4 *picks her current resource, a_5 still agrees to pass;*

5. a_2 *picks her current resource,* a_5 *picks her current resource,* a_3 *picks her current resource. All the agents have left with their resource and the Diver returns 'PO'.*

Next, we prove the correctness of the Diver and show that it runs in $\mathcal{O}(n)$.

Theorem 1. *The Diver always terminates, runs in $\mathcal{O}(n)$ and returns whether the initial assignment is Pareto-optimal or not.*

Proof. Termination is obvious since the procedure proceeds in a single main screening of the finite set of resources. We first show that the procedure is sound. First observe that when the Diver returns 'PO', all the agents must have picked their initial resource. Indeed, consider the last agent in the order: this agent picked her resource (otherwise the procedure would have returned 'not PO'). But now the agent on her left must also have picked her resource (as there is no more possibility to dive), and so on until there are no agents remaining. Now, following the argument used in Bade (2019), consider all the agents who picked their resource during this process, in the order they picked it: they clearly have all picked their best available resource. The obtained matching is thus indeed Pareto-optimal. On the other hand, when the Diver returns 'not PO', there is indeed an improving cycle, consisting of the agent (say, a_j who chose a resource on her left, and all the agents, from the owner of this resource to a_j, who are not matched yet.

In terms of complexity, sorting the agents according to the single-peaked order can be done in $\mathcal{O}(n)$ using counting sort (Cormen et al., 2009, Section 8.2). Now for the main loop of the procedure: in the reverse loop, note that if $k+1$ agents are screened backwards, then k agents are removed for good. Thus through the entire procedure the reverse loop involves $\mathcal{O}(n)$ steps, and thus the main loop takes $\mathcal{O}(n)$ as well, which gives us the linear time complexity. □

The same line of analysis allows us to derive a result regarding the amount of communication induced by the procedure, as can be done for other social choice mechanisms, see *e.g.* (Brandt et al., 2016, Chapter 10). The Diver only requires a linear (in the number of agents) number of bits to be communicated from the agents.

Proposition 3. *The Diver requires 4n bits of communication.*

Proof. The key is to observe that sub-protocol backtrack-call requires overall $n + n$ bits, as there may only be n agents saying 'no' and n agents saying 'yes' throughout the whole run of the Diver. In the main loop of the protocol, the query requires 2 bits to be answered (as there are three possible answers). This makes overall $2n + 2n = 4n$ bits, thus, only $\mathcal{O}(n)$ bits. □

We now show that the Diver is asymptotically optimal, both in terms of time and communication complexity. In fact, a simple adversarial argument suffices to show that any algorithm solving this problem must query at least $n - 1$ agents and thus read an input of this size at least.

Proposition 4. *In the single-peaked domain, the time complexity of* CHECKPO *is $\Omega(n)$.*

Proof. Simply take as adversarial input an instance where every agent receives initially her preferred house. No procedure can answer CHECKPO before querying less than $n - 1$ agents. Indeed, as long as two agents remain to be queried, it could be that they form a trading cycle. □

As each query requires at least one bit to be answered, this immediately implies a similar bound on the communication complexity. This can alternatively be shown by exhibiting a straightforward *fooling set* (Kushilevitz and Nisan, 1996), which we provide for completeness. We consider strict preferences for the agents written $\succ_i^L$ in a profile L, and by a slight abuse of notation we write $\succ_i^{L|L'}$ to say that $\succ_i$ is the preference of agent a_i in either L or L'. In our context, the fooling set will be a collection of profiles $\mathcal{F} = \langle L_1, \ldots L_K \rangle$ such that:

1. for any $i \in \{1, \ldots K\}$ CHECKPO's answer on $\langle \mathcal{N}, \mathcal{R}, L_i, \pi^0 \rangle$ is yes.

2. for any $i \neq j$, there exists $L' = \langle \succ_1^{L_i|L_j}, \cdots \succ_n^{L_i|L_j} \rangle$, such that CHECKPO's answer on $\langle \mathcal{N}, \mathcal{R}, L', \pi^0 \rangle$ is no.

By a standard result in communication complexity, it is known that $\log |\mathcal{F}|$ is a lower bound on the communication complexity of the problem (Kushilevitz and Nisan, 1996).

Proposition 5. *In the single-peaked domain, the communication complexity of* CHECKPO *is $\Omega(n)$.*

Proof. Let us call a *consensual profile* the profile where $\succ_i = \succ_j$ for any (i,j), i.e. all the agents have the same linear orders over the resources. The consensual linear order will be denoted by $\succ$. We claim that the set $\mathcal{F}$ of the 2^{n-1} (single-peaked) consensual profiles constitutes a fooling set.

To show this, first observe that in any such profile, the original assignment π^0 is Pareto-optimal. Indeed, in a consensual profile, no trading cycle is possible. Hence the aforementioned condition 1. of a fooling set is satisfied.

Now to show that we can fool the function, consider any pair of profiles (L_i, L_j). As these profiles are different, there must exist at least one pair of resources (r_p, r_q) such that $r_p \succ r_q$ in L_i, while $r_q \succ r_p$ in L_j (it is true for all agents since the profiles are consensual). Now consider the agent a_p (resp. a_q) holding r_p (resp. r_q) in π^0 and a mixed profile L' such that:

$$\forall k \in \mathcal{N}, \succ_k^{L'} = \begin{cases} \succ_k^{L_i} & \text{if } k \neq p, \\ \succ_k^{L_j} & \text{if } k = p. \end{cases}$$

Hence, a_p and a_q have opposite preferences for r_p and r_q and would prefer to swap, i.e. CHECKPO's answer on $\langle \mathcal{N}, \mathcal{R}, L', \pi^0 \rangle$ is no. This concludes the proof. □

The Diver is thus asymptotically optimal in terms of time and communication complexity.

5. Future work

A natural extension of the domain studied is to allow for indifferences in preferences. In the universal domain where indifferences can be expressed, Aziz and

De Keijzer (2012) defined a set of Pareto-optimal procedures generalizing TTC in that setting, which however include procedures that are not strategy-proof. Plaxton (2013) and Saban and Sethuraman (2013) independently proposed general frameworks for efficient and strategy-proof generalization of the TTC procedure with indifferences. In her paper, Bade (2019) defines the "circle crawling" procedure, which enjoys the same properties as the crawler. It would be interesting to study whether a variant in the spirit of the Diver could be adapted in that setting as well.

6. References

Abraham, D.J., Cechlárová, K., Manlove, D.F., Mehlhorn, K., 2005. Pareto optimality in house allocation problems, in: Deng, X., Du, D.Z. (Eds.), Algorithms and Computation, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 1163–1175.

Arrow, K.J., 1951. Social choice and individual values .

Aziz, H., De Keijzer, B., 2012. Housing markets with indifferences: A tale of two mechanisms, in: Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI).

Aziz, H., Hougaard, J.L., Moreno-Ternero, J.D., Østerdal, L.P., 2017. Computational aspects of assigning agents to a line. Mathematical Social Sciences 90, 93–99.

Bade, S., 2019. Matching with single-peaked preferences. Journal of Economic Theory 180, 81–99.

Black, D., 1948. On the rationale of group decision-making. Journal of political economy 56, 23–34.

Brandt, F., Conitzer, V., Endriss, U., Lang, J., Procaccia, A.D., 2016. Handbook of computational social choice. Cambridge University Press.

Bruner, M.L., Lackner, M., 2015. On the likelihood of single-peaked preferences. CoRR abs/1505.05852.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C., 2009. Introduction to algorithms. MIT press.

Damamme, A., Beynier, A., Chevaleyre, Y., Maudet, N., 2015. The power of swap deals in distributed resource allocation, in: Proceedings of the 14th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), pp. 625–633.

Elkind, E., Lackner, M., Peters, D., 2017. Structured preferences, in: E., U. (Ed.), Trends in Computational Social Choice, pp. 187–207.

Hougaard, J.L., Moreno-Ternero, J.D., Østerdal, L.P., 2014. Assigning agents to a line. Games and Economic Behavior 87, 539–553.

Kushilevitz, E., Nisan, N., 1996. Communication Complexity. Cambridge University Press.

Li, S., 2017. Obviously strategy-proof mechanisms. American Economic Review 107, 3257–87.

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

Moulin, H., 1991. Axioms of cooperative decision making. 15, Cambridge university press.

Moulin, H., 2018. Fair division in the age of internet .

Plaxton, C.G., 2013. A simple family of top trading cycles mechanisms for housing markets with indifferences.

Roth, A.E., 1982. Incentive compatibility in a market with indivisible goods. Economics letters 9, 127–132.

Saban, D., Sethuraman, J., 2013. House allocation with indifferences: a generalization and a unified view, in: Proceedings of the 14th ACM Conference on Electronic Commerce (ACM-EC), ACM. pp. 803–820.

Shapley, L., Scarf, H., 1974. On cores and indivisibility. Journal of mathematical economics 1, 23–37.

Sprumont, Y., 1991. The division problem with single-peaked preferences: a characterization of the uniform allocation rule. Econometrica , 509–519.