Social ranking under incomplete knowledge: elicitation of the lex-cel necessary winners

Appendix

Proofs

Proposition 1

Proposition. *Let X be a finite set. Consider a power relation $\succsim \in \boldsymbol{\mathcal{R}}(\mathcal{P}(X))$ with the associated quotient order $\Sigma_1 \succ \Sigma_2 \succ \ldots \succ \Sigma_l$. Each winner in $R_{le}^{\succsim}$ is necessarily contained in Σ_1.*

Proof. Let x be a winner in $R_{le}^{\succsim}$. By the definition of lex-cel, $\theta^{\succsim}(x) = (x_1, \ldots, x_l) \geq_L (y_1, \ldots, y_l) = \theta^{\succsim}(y)$ for all $y \in X$. So, $x_1 \geq y_1$ for any $y \in X$, which implies that $x_1 \geq 1$ since Σ_1 must contain at least one non-empty subset of X. □

Proposition 2

Proposition. *Algorithm 1 returns a set W of lex-cel necessary winners for the power relation $\succsim$.*

Proof. If $|X| > 1$, after the initialisation the procedure enters in the while loop, otherwise the set of lex-cel necessary winners is directly $W = X$.

At each iteration, if all coalitions in $\mathcal{M}^{\succsim}$ are equivalent, the algorithm updates the current set of lex-cel possible winners by keeping only the elements from W that have at least as many occurrences in $\mathcal{M}^{\succsim}$ as any other element from W. Then, the procedure removes $\mathcal{M}^{\succsim}$ from the collection $\mathcal{C}$ and restricts $\succsim$ to the same collection $\mathcal{C}$.

The subroutine $SELECT(\succsim^{\mathcal{C}}_{\mathcal{P}(X)})$ selects two coalitions C_1 and C_2 from the collection $\mathcal{C}$ such that neither (C_1, C_2) nor (C_1, C_2) are in $\succsim^{\mathcal{C}}_{\mathcal{P}(X)}$. The subroutine $QUERY(C_1, C_2)$ then reveals the relation between the selected coalitions, and returns the associated set of tuples. The gathered information is inserted into the update of $\succsim$, as well as any additional relation determined by transitive closure.

The while loop continues until $|W| = 1$ or there are no more coalitions in $\mathcal{C}$, then it stops and returns the current $\succsim$ and W. If $|W| = 1$, since a lex-cel necessary winner always exists and it must be a lex-cel possible winner, the unique element contained in W is the unique lex-cel necessary winner on $\succsim$. Otherwise, if $|W| > 1$ but $\mathcal{C} = \emptyset$, it means that all the relations between coalitions C_1 and C_2 are revealed and, according to the manner W is generated maximizing the occurrences x_U, we have that each element $x \in W$ is such that $\theta^{\succsim}(x) = \theta^{\succsim}(y)$ for each $y \in W$ and $\theta^{\succsim}(x) >_L \theta^{\succsim}(z)$ for each $z \in X \setminus W$. So W is returned as the set of lex-cel necessary winners on $\succsim$. □

Proposition 3

Proposition. *Given a total preorder $\succsim^*_{\mathcal{P}(X)}$ over $\mathcal{P}(X)$, let $x, y \in X$ be present in the best coalition $C^* \in \succsim^*_{\mathcal{P}(X)}$. It holds that $x \; R_{le}^{\succsim^*_{\mathcal{P}(X)}} y$ if $C_{x,-y} \succ^*_{\mathcal{P}(X)} C_{y,-x}$.*

The proof for this proposition is similar to that of Proposition 1.

Proposition 4

Proposition. *Given a total preorder $\succsim^*_{\mathcal{P}(X)}$ over $\mathcal{P}(X)$. Let $\succsim_{\mathcal{P}(X)}$ be a preorder such that $\succsim_{\mathcal{P}(X)} \subseteq \succsim^*_{\mathcal{P}(X)}$. Let $x, y \in X$ be the two best elements in X according to the global lex-cel. Without loss of generality, it holds that $xR^{\succsim^*_{\mathcal{P}(X)}}_{le} y$ iff there exists a coalition in $\mathcal{M}^{\succsim_{\mathcal{P}(X)}}_{x,-y}$ preferred to every other coalition in $\mathcal{M}^{\succsim_{\mathcal{P}(X)}}_{y,-x}$, or if $\mathcal{M}^{\succsim_{\mathcal{P}(X)}}_{y,-x} = \emptyset$ and $\mathcal{M}^{\succsim_{\mathcal{P}(X)}}_{x,-y} \neq \emptyset$.*

Proof. Let $\succsim^*_{\mathcal{P}(X)}$ be a total preorder over $\mathcal{P}(X)$, and let $\succsim_{\mathcal{P}(X)} \subseteq \succsim^*_{\mathcal{P}(X)}$. By construction, we know that $C_{x,-y} \in \mathcal{M}^{\succsim_{\mathcal{P}(X)}}_{x,-y}$ and $C_{y,-x} \in \mathcal{M}^{\succsim_{\mathcal{P}(X)}}_{y,-x}$. If, without loss of generality, $\mathcal{M}^{\succsim_{\mathcal{P}(X)}}_{y,-x} = \emptyset$ and $\mathcal{M}^{\succsim_{\mathcal{P}(X)}}_{x,-y} \neq \emptyset$, then we know that no coalition containing only y and not x will be situated before a coalition containing x and not y. Therefore, no query is required to uncover that $xR^{\succsim^*_{\mathcal{P}(X)}}_{le} y$.

If both sets are nonempty, however, and let C_1 be the coalition in $\mathcal{M}^{\succsim_{\mathcal{P}(X)}}_{x,-y}$ that is preferred to every coalition in $\mathcal{M}^{\succsim_{\mathcal{P}(X)}}_{y,-x}$. As $C_{y,-x} \in \mathcal{M}^{\succsim_{\mathcal{P}(X)}}_{y,-x}$, then, by transitivity of $\succsim$, it must hold that $C_{x,-y} \succsim C_1 \succ C_{y,-x}$, therefore, from Proposition 3, it is verified that $xR^{\succsim^*_{\mathcal{P}(X)}}_{le} y$. Similarly, if $xR^{\succsim^*_{\mathcal{P}(X)}}_{le} y$, then, from Proposition 3, it must hold that $C_{x,-y} \succ C_{y,-x}$. By construction, this means there exists a coalition in $\mathcal{M}^{\succsim_{\mathcal{P}(X)}}_{x,-y}$ that is preferred to every coalition in $\mathcal{M}^{\succsim_{\mathcal{P}(X)}}_{y,-x}$. □

Pseudo-code

Elicitation by reconstitution: variants of RECO

Algorithm 1: SELECT: C-RAND

$C_1 \leftarrow$*rand*$(\mathcal{P}(X))$;
Cands $\leftarrow \{C \in \mathcal{P}(X) : (C_1, C) \notin \succsim^{\mathcal{C}}_{\mathcal{P}(X)}$ and $(C, C_1) \notin \succsim^{\mathcal{C}}_{\mathcal{P}(X)}\}$;
$C_2 \leftarrow$ *rand*(Cands);
return C_1, C_2.

Algorithm 2: SELECT: R-RAND

$C_1 \leftarrow$*rand*$(\mathcal{M}^{\succsim^{\mathcal{C}}_{\mathcal{P}(X)}})$;
$C_2 \leftarrow$*rand*$(\{C \in \mathcal{M}^{\succsim^{\mathcal{C}}_{\mathcal{P}(X)}} : (C, C_1) \notin \succsim^{\mathcal{C}}_{\mathcal{P}(X)}$ and $(C_1, C) \notin \succsim^{\mathcal{C}}_{\mathcal{P}(X)}\})$;
return C_1, C_2.

Algorithm 3: SELECT: R-LEX

$C_1 \leftarrow m_1$;
$i \leftarrow 1$;
while $C_1 = m_i$ **do**
  $i \leftarrow i + 1$;
$C_2 \leftarrow m_i$;
return C_1, C_2.

Algorithm 4: SELECT: R-MIN_INT

$C_1 \leftarrow \emptyset$; $C_2 \leftarrow \emptyset$; min $\leftarrow |X|$;
for $i \in \{1, \ldots, m\}$ **do**
  for $j \in \{i + 1, \ldots, m\}$ **do**
    intersec $\leftarrow |(\mathcal{M}^{\succsim_{\mathcal{P}(X)}})_i \cap (\mathcal{M}^{\succsim_{\mathcal{P}(X)}})_j|$;
    if *intersec* $\neq 0$ *and intersec* $<$ *min* **then**
      $C_1 \leftarrow (\mathcal{M}^{\succsim_{\mathcal{P}(X)}})_i$;
      $C_2 \leftarrow (\mathcal{M}^{\succsim_{\mathcal{P}(X)}})_j$;
      min $\leftarrow$ intersec;
return C_1, C_2.

Elicitation through prospection: PROSP

In the following, we denote by $(\hat{M}^{\succsim})_i$ the i-th component of a vector $\hat{M}^{\succsim}$.

Algorithm 5: Elicitation through prospection (PROSP)

Input: A preorder $\succsim_{\mathcal{P}(X)}$ over $\mathcal{P}(X)$;
Output: A preorder $\succsim \supseteq \succsim_{\mathcal{P}(X)}$ and a set of lex-cel necessary winners on $\succsim$;

$W \leftarrow \{x \in C \mid C \in \mathcal{M}^{\succsim}_{\mathcal{P}(X)}\}$;
$L \leftarrow \emptyset$;
eq $\leftarrow \emptyset$;
while $|W \setminus L| > 1$ **do**
  $x, y \leftarrow$ top-global-lexcel$(W \setminus L, \succsim)$;
  while $\hat{M}^{\succsim}_{x,-y} \neq \emptyset$ *and* $\hat{M}^{\succsim}_{y,-x} \neq \emptyset$ **do**
    $C_1 \leftarrow (\hat{M}^{\succsim}_{x,-y})_1$;
    $C_2 \leftarrow (\hat{M}^{\succsim}_{y,-x})_1$;
    Newcomp $\leftarrow QUERY(C_1, C_2)$;
    $\succsim \leftarrow$ Transitive_Closure($\succsim \cup$ Newcomp);
    if $C_1 \sim C_2$ **then**
      $\hat{M}^{\succsim}_{x,-y} \leftarrow (\hat{M}^{\succsim}_{x,-y} \setminus \{C_1\}) \cup$ next$(x, y, C_1, \succsim)$;
      $\hat{M}^{\succsim}_{y,-x} \leftarrow (\hat{M}^{\succsim}_{y,-x} \setminus \{C_2\}) \cup$ next$(y, x, C_2, \succsim)$;
  if $\hat{M}^{\succsim}_{x,-y} \cup \hat{M}^{\succsim}_{y,-x} = \emptyset$ **then**
    eq $\leftarrow$ eq $\cup \{(x, y)\}$;
    $W \leftarrow W \setminus \{y\}$;
  else if $\hat{M}^{\succsim}_{x,-y} = \emptyset$ **then**
    $L \leftarrow L \cup \{x\}$;
  else
    $L \leftarrow L \cup \{y\}$;
$NW \leftarrow$ equiv$(W \setminus L,$ eq$)$;
return $\succsim$;
return NW;

We present, in Algorithm 5, the pseudo-code of the method of elicitation through prospection.

Given a set $\mathcal{M}^{\succsim}$ of maximal elements, in the following we denote by $\hat{M}^{\succsim}$ the vector of coalitions in $\mathcal{M}^{\succsim}$ ordered in ascending lowest position in a maximal chain of $\succsim$, and by $(\hat{M}^{\succsim})_i$ its i-th component. We denote by L the set of elements over which preference for another lex-cel possible winner has been determined in a previous step.

In the event of an equivalence between a coalition $C_1 \in \hat{M}^{\succsim}_{x,-y}$ and a coalition $C_2 \in \hat{M}^{\succsim}_{y,-x}$, we know that C_1's impact in favour of x in its occurrence vector will be neutralized by that of C_2 in favour of y. Therefore, since their respective weight in favour of each element in countered by the other, we can simply discard them from the set of studied coalitions, and replace them with the next best coalition C in each of their respective maximal chain(s) verifying that $|C \cap \{x, y\}| = 1$, as each such coalition C has the potential to be the determining coalition to distinguish x from y. This is done using the function next(a,b,C_i,$\succsim$), which returns a set of coalitions placed after in C_i in a maximal chain, that are the first after C_i to contain either x or y, but not both.

NW is the set of necessary winners returned by the procedure, using the function equiv to gather all the elements that may have been found to be equivalent to the last remaining element in $W \setminus L$ in an earlier step.

Notice that the algorithm uses as the set W of lex-cel possible winners a set containing all elements appearing in at least one of the coalitions in $\mathcal{M}^{\succsim}_{\mathcal{P}(X)}$. The set W therefore constitutes a superset of the

true set P of lex-cel possible winners, as an element $x \in X$ may be present in a coalition $C \in \mathcal{M}^{\succsim}_{\mathcal{P}(X)}$ but still such that there exists no extension of $\succsim$ in which x is a winner. However, any such element $x \in W \setminus P$ will verify that $\hat{M}^{\succsim}_{x,-y} = \emptyset$, no matter the element y in P to which it is compared, and placing x in W will therefore yield no unnecessary query.

Experimental results

Best-case scenario: an example of unique lex-cel possible winner

Example 1. *Let $X = \{1, 2, 3, 4, 5\}$. Let the total preorder $\succsim^*_{\mathcal{P}(X)}$ over $\mathcal{P}(X)$ be such that*

$$1245 \succ 1345 \succ 234 \succ 12345 \succ 34 \sim 145 \succ 3 \succ 4 \sim 24 \succ 245 \succ 45 \succ 15 \ldots,$$

and let the available preorder $\succsim_{\mathcal{P}(X)} \subseteq \succsim^*_{\mathcal{P}(X)}$ contain the following maximal chains:

$$1245 \succ 1345 \succ 34 \succ 3 \succ 24 \ldots$$
$$4 \succ 15 \succ 124 \succ 35 \sim 2345 \succ 1234 \ldots$$
$$234 \succ 12345 \succ 145 \succ 25 \succ 2 \ldots$$

Using the global lex-cel, we obtain the following occurrence vectors:
$\vartheta^{\succsim_{\mathcal{P}(X)}}(1) = (1, 3, 2, 0, 1, \ldots)$, $\vartheta^{\succsim_{\mathcal{P}(X)}}(2) = (2, 1, 1, 2, 3, \ldots)$, $\vartheta^{\succsim_{\mathcal{P}(X)}}(3) = (1, 2, 1, 3, 1, \ldots)$,
$\vartheta^{\succsim_{\mathcal{P}(X)}}(4) = (3, 2, 3, 1, 2, \ldots)$, $\vartheta^{\succsim_{\mathcal{P}(X)}}(5) = (1, 3, 1, 3, 0, \ldots)$,
from which we can determine that

$$4 \; R^{\succsim_{\mathcal{P}(X)}}_{gle} \; 2 \; R^{\succsim_{\mathcal{P}(X)}}_{gle} \; 1 \; R^{\succsim_{\mathcal{P}(X)}}_{gle} \; 5 \; R^{\succsim_{\mathcal{P}(X)}}_{gle} \; 3.$$

The first step of PROSP will therefore consist in determining $\mathcal{M}^{\succsim_{\mathcal{P}(X)}}_{4,2} = \{1345, 4, 145\}$. We notice that $\mathcal{M}^{\succsim_{\mathcal{P}(X)}}_{2,-4} = \emptyset$, meaning that we already know that $4 \; R^{\succsim_{\mathcal{P}(X)}}_{le} \; 2$ without submitting a query.

In the next step, the procedure determines $\mathcal{M}^{\succsim_{\mathcal{P}(X)}}_{4,1} = \{34, 4, 234\}$, and we notice that $\mathcal{M}^{\succsim_{\mathcal{P}(X)}}_{1,-4} = \emptyset$, from which we obtain that $4 \; R^{\succsim_{\mathcal{P}(X)}}_{le} \; 1$.

Then we study the set $\mathcal{M}^{\succsim_{\mathcal{P}(X)}}_{4,5} = \{34, 4, 234\}$, and determine $\mathcal{M}^{\succsim_{\mathcal{P}(X)}}_{5,-4} = \emptyset$. It therefore holds that $4 \; R^{\succsim_{\mathcal{P}(X)}}_{le} \; 5$.

Finally, the procedure determines $\mathcal{M}^{\succsim_{\mathcal{P}(X)}}_{4,3} = \{1245, 4, 145\}$, and we notice once again that $\mathcal{M}^{\succsim_{\mathcal{P}(X)}}_{3,-4} = \emptyset$, meaning that $4 \; R^{\succsim_{\mathcal{P}(X)}}_{le} \; 3$.

Since the element 4 is preferred to every other element in X, it is the lex-cel necessary winner for any extension of $\succsim_{\mathcal{P}(X)}$, which includes $\succsim^*_{\mathcal{P}(X)}$. This has been determined without submitting a single query.

Under restriction of monotonicity

A power relation $\succsim_{\mathcal{P}(X)}$ is said to be **monotonic** if, for any $S, T \in \mathcal{P}(X)$ with $S \subseteq T$, it holds that $T \succsim S$. While in terms of number of submitted queries, we find that the performance of RECO and PROSP over monotonic preorders are similar than for any randomly-generated preorder, a significant difference lies in the resulting preorder over elements uncovered throughout the elicitation. Indeed, we find that in most cases, the requirement of monotonicity leads RECO to recover most of the total preorder over X, when PROSP rarely discovers the preference relation over more than 3 elements in X. Therefore, in terms of completeness of the recovered preorder over X, it could be argued that RECO outperforms PROSP over monotonic preorders.

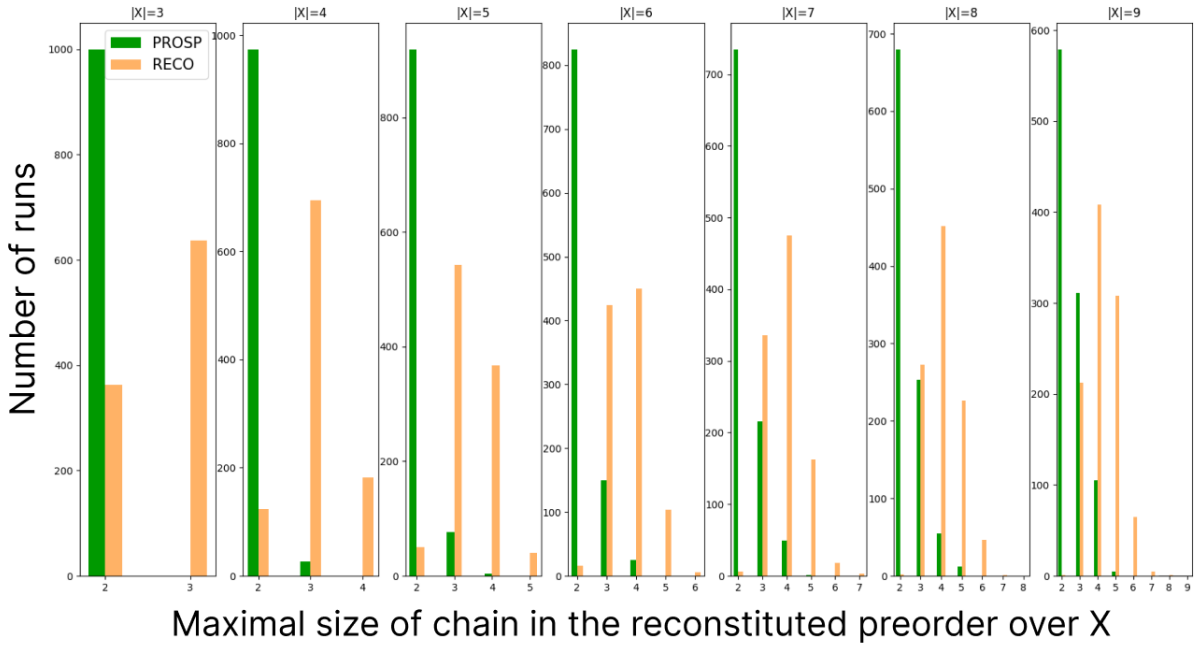

Figure 1: Maximal size of the chains in the preorder over X retrieved by PROSP and RECO over monotonic total preorders

Investigating a restriction on the number of maximal chains

As specified at the beginning of the section on Experimental results, the experiments introduced in this paper are all based on preorders where the number m of maximal chains is determined uniformly at random in the set $\{2, \ldots, \frac{2^n}{2}\}$, with n the size of the studied population X.

We investigate the impact that setting a number of maximal chains may have on the performance of each method, and observe that, when $m \in \{2, \ldots, 6\}$, PROSP systematically outperforms any other variant of RECO. This is for instance made evident in Figure 2.

As this only holds for small values of m, these results indicate a better performance of PROSP in scenarios where there is only a small amount of missing information. Notably, we observe that it is then more likely for PROSP to determine the necessary winner without submitting a single query to the user.

However, when m is set at a higher value than 7, we find that we return to previously observed results, where R-RAND becomes increasingly better as the size of the population increases. This can be seen in Figure 3, with m set as low as 7.

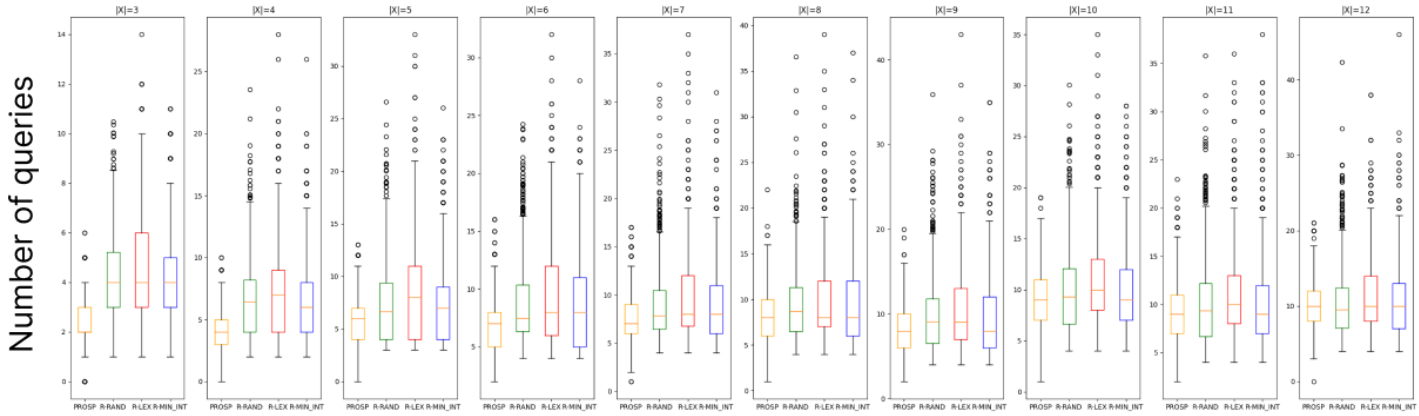

Figure 2: Number of queries submitted by each method when $m = 5$ based on the size of the population

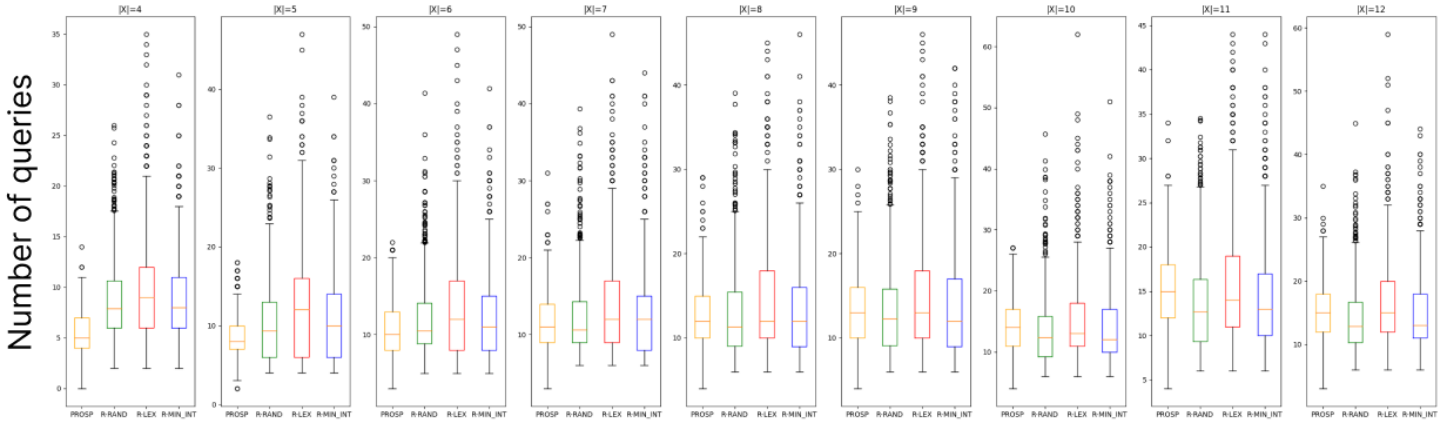

Figure 3: Number of queries submitted by each method when $m = 7$ based on the size of the population

A study of computation times

We study the average computation time per run for each introduced method under different parameters: Table 1 indicates the average computation time for each method when the underlying total preorder is determined uniformly at random; Table 2 when m is set to 5; and Table 3 when the underlying total preorder is monotonic.We observe that, depending on the setup, either the R-RAND or the R-LEX variant of RECO is the fastest to solve our problem.

n	PROSP	R-RAND	R-LEX	R-MIN_INT
3	5.3 $\times e^{-5}$	3.9 $\times e^{-5}$	3.8 $\times e^{-5}$	4.5 $\times e^{-5}$
4	1.6 $\times e^{-4}$	1.9 $\times e^{-4}$	9.0 $\times e^{-5}$	1.9 $\times e^{-4}$
5	0.0008	0.0003	0.0003	0.001
6	0.005	0.0009	0.0009	0.01
7	0.05	0.003	0.004	0.2
8	0.5	0.01	0.02	3.2
9	4.5	0.07	0.1	54

Table 1: Average computation time (in seconds) of each method based on the population size

n	PROSP	R-RAND	R-LEX	R-MIN_INT
3	8.0 $\times e^{-5}$	6.0 $\times e^{-5}$	5.9 $\times e^{-5}$	8.8 $\times e^{-5}$
4	0.0002	0.0001	0.0001	0.0002
5	0.0004	0.0002	0.0001	0.0003
6	0.0007	0.0002	0.0002	0.0005
7	0.002	0.0004	0.0003	0.009
8	0.004	0.0007	0.0005	0.002
9	0.008	0.001	0.0009	0.003
10	0.02	0.002	0.002	0.006
11	0.05	0.005	0.004	0.01
12	0.13	0.01	0.009	0.03

Table 2: Average computation time (in seconds) of each method when $m = 5$ based on the population size

n	PROSP	R-RAND	R-LEX	R-MIN_INT
3	6.2 $\times e^{-5}$	5.4 $\times e^{-5}$	5.4 $\times e^{-5}$	6.1 $\times e^{-5}$
4	0.0002	0.0002	0.0001	0.0003
5	0.001	0.0005	0.0004	0.002
6	0.014	0.002	0.002	0.03
7	0.15	0.005	0.007	0.38
8	1.75	0.02	0.04	5.99
9	10	0.1	0.2	104

Table 3: Average computation time (in seconds) of each method over monotonic total preorders based on the population size