Incomplete Preferences in Single-Peaked Electorates

Martin Lackner
Vienna University of Technology, Austria
lackner@dbai.tuwien.ac.at

Abstract

Incomplete preferences are likely to arise in real-world preference aggregation and voting systems. This paper deals with determining whether an incomplete preference profile is single-peaked. This is essential information since many intractable voting problems become tractable for single-peaked profiles. We prove that for incomplete profiles the problem of determining single-peakedness is NP-complete. Despite this computational hardness result, we find four polynomial-time algorithms for reasonably restricted settings.

1 Introduction

Both human and automated decision making often have to rely on incomplete information. The same issue arises in joint decision making – voting – in multi-agent systems. For example, the majority of preference data collected on `preflib.org` (Mattei and Walsh 2013) is incomplete. Konczak and Lang (2005) distinguish two main sources of incompleteness: The first one is *intrinsic* incompleteness where the voter is unable or unwilling to give complete information, i.e., a total order on all candidates. The second one is *epistemic* incompleteness where the voters do have preferences specified by total orders but at the time of decision making these total orders are not fully available. Also a combination of these two scenarios is possible.

Whereas complete preferences are usually modeled as total orders, incomplete preferences can be modeled as partial orders and are therefore a more general concept. In particular, the determination of winners becomes harder since voting protocols usually require total orders. It is therefore necessary to consider *completions* of incomplete votes. Completions of incomplete votes are total orders that are compatible with the original partial orders. The determination of possible and necessary winners in incomplete elections is often NP-hard and thus a fast winner determination is not feasible (Konczak and Lang 2005; Walsh 2007; Betzler and Dorn 2010; Pini et al. 2011; Xia and Conitzer 2011; Baumeister and Rothe 2012).

A popular approach to deal with hardness of voting problems is to consider domain restrictions. The most common restriction is *single-peakedness* (Black 1948) (see preliminaries for a definition). For example, computing the winner of a Dodgson or Kemeny election, though Θ_2^P-complete in general (Hemaspaandra, Hemaspaandra, and Rothe 1997; Hemaspaandra, Spakowski, and Vogel 2005), can be done in polynomial time for single-peaked elections (Brandt et al. 2010). Also the complexity of manipulation and control problems often decreases (Faliszewski et al. 2011). These results let us hope that efficient, polynomial time algorithms for computing possible and necessary winners of single-peaked, incomplete elections could be found. Walsh (2007) started investigating this issue and also pointed out a central question in that regard: What happens if the axis for which the incomplete preference profile is single-peaked is not given as part of the input but has to be determined?

Our paper deals with this question, namely how to determine single-peakedness for incomplete elections. In the following, let n denote the number of votes and let m denote the number of candidates. The main results are as follows:

• We prove that determining whether an incomplete preference profile is single-peaked is NP-complete. This is in contrast to the case of complete preferences for which single-peakedness can be determined in linear time (Escoffier, Lang, and Öztürk 2008). Furthermore, we strengthen this result by showing that NP-completeness still holds if one voter completely specifies his preferences.

Apart from these hardness results, this paper contains four polynomial time algorithms:

• The first algorithm requires that the preference profile contains at least one complete vote, i.e., a total order. The algorithm is applicable to weak orders (see Figure 1 for an example and the preliminaries for a definition). We obtain a runtime of $\mathcal{O}(m \cdot n)$. This algorithm is an improvement over the algorithm by Escoffier, Lang, and Öztürk (2008) since it is applicable to a broader class of preference profiles (weak orders instead of total orders) while maintaining its runtime.

• Our second algorithm is 2-SAT based. It also requires a total order but is applicable to local weak orders, which are a generalization of weak orders. This more general algorithm does not run in linear time but requires $\mathcal{O}(m^3 \cdot n)$ time.

• In contrast to the previous two algorithms, the third algorithm does not require the profile to contain a total order. However, it is restricted to top orders. Top orders rank an arbitrary number of top candidates; all remaining candidates are ranked last and incomparable to one another (see Figure 1 for an example). This algorithm has a runtime of $\mathcal{O}(m^2 \cdot n)$.

• Finally, we have considered the problem of determining single-peakedness for an already given axis. We prove this problem to be polynomial-time solvable even for incomplete profiles consisting of partial orders.

2 Preliminaries

In this paper, preferences are represented by different types of orders (see Figure 1 for examples). The most general type are partial orders. A *partial order* P on a set X is a reflexive, antisymmetric and transitive binary relation on X. We say that y *is ranked above* x if xPy holds. If for two elements $x, y \in X$ neither xPy nor yPx holds, these two elements are *incomparable*. A partial order where the incomparability relation is transitive is called a *weak order*. A weak order can thus be considered a total order with ties. Weak orders are also referred to as bucket orders (elements that tie are in the same "bucket"), cf. (Fagin et al. 2006). A weak order where every incomparable element is minimal is called *top order*. The *ranked* candidates of a top order T are those that are not incomparable to another candidate. We would like to remark that top orders appear as *top lists* in (Dwork et al. 2001; Fagin, Kumar, and Sivakumar 2003) and as top-truncated votes in (Baumeister et al. 2012). A partial order with no incomparable elements is called *total order*. Any partial order P can be *extended* to some total order T such that aPb implies aTb; T is then a (not necessarily unique) *extension* of P. Finally, we define a *local weak order* P on a set X to be a partial order on X with the following property: there exist sets X_1, X_2 with $X_1 \cup X_2 = X$ such that the elements in X_1 are incomparable to every other element in X and the profile P restricted to X_2 is a weak order. Intuitively, a local weak order is a weak order together with some isolated elements for which absolutely no information is available. Note that we do not distinguish between tied and incomparable elements in this paper; both are treated in the same way.

Total orders are denoted by $\langle c_1 > c_2 > \ldots > c_k \rangle$. We use $\langle c_1 > c_2 > \ldots > c_k > \bullet \rangle$ to denote a top order where $c_1, \ldots, c_k$ are ranked as stated and all other elements (usually the remaining candidates in C) are ranked last, i.e., are minimal elements. We use $\langle\rangle$ to denote the empty order relation. We sometimes use set operators ($\cup, \cap, \setminus$) on top orders with the intended meaning that we apply these operators to the corresponding sets of ranked candidates.

We would now like to address the usefulness of these orders for expressing preferences. Total orders allow to fully specify a ranking of options. Given a large set of options, this might be unfeasible. Partial orders, on the other hand, allow to specify the relative order of any pair of options. Thus they can be seen as a very general formalism for representing incomplete preferences. They are compatible with total orders in the sense that partial orders can always be extended to total orders. Weak orders are less general than partial orders but arise in many natural scenarios. For example every real-valued utility function implies a weak order (candidates with the same utility tie, i.e., are incomparable). Local weak orders correspond to partial real-valued utility functions and thus arise in scenarios where voters do not have knowledge about all candidates. If the elicitation of preferences is costly, one might ask only for the most important (top ranked) options of each voter. In such a case, top orders arise. Top orders also are the natural type of order for specifying preferences in some scoring protocols. We will further comment on scoring protocols and top orders at the end of the paper.

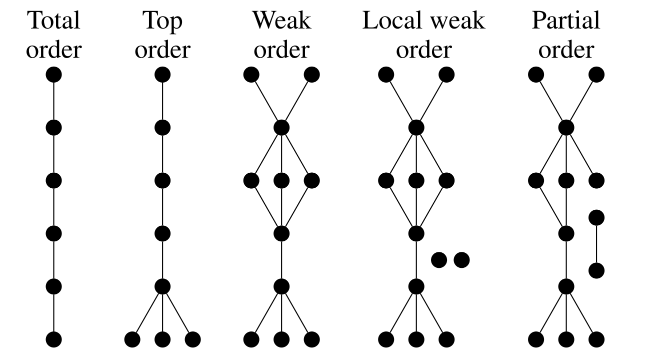

Figure 1: The order zoo: examples of different types of orders that are used to specify preferences.

Throughout this paper we use C to denote the set of candidates or options. Votes are considered to be either partial, local weak, weak, top or total orders. For a vote V_i, we use $x \succ_i y$ to denote that $(yV_ix) \wedge (x \neq y)$, i.e., x is ranked strictly higher than y. If there is only one vote under consideration, usually denoted by V, we omit the index and write $x \succ y$. A tuple $(V_1, \ldots, V_n)$ of votes is called a *(preference) profile of* $\{$*partial orders, weak orders, top orders, total orders*$\}$, depending on the type of orders. Given a vote V and a set of candidates $C' \subseteq C$, we define $V[C']$ to be the vote V restricted to candidates in C'. Analogously, given a preference profile $\mathcal{P} = (V_1, \ldots, V_n)$, we define $\mathcal{P}[C']$ to be $(V_1[C'], \ldots, V_n[C'])$. We denote the number of candidates with m and the number of votes with n.

3 Single-peaked Profiles

We start by giving a definition for single-peaked profiles of total orders and then extend this definition to partial orders. A central concept is that of an axis, which is a total order on C. Let A be an axis. Throughout this paper we write $x < y$ instead of xAy. (Note that we use $\succ$ for votes and $<$ for axes.) Our definition of single-peakedness for profiles of total orders as well as for profiles of partial orders is based on so-called *valleys*.

Definition 1 (v-valleys)**.** Let V be a partial order on C. A vote V *contains a v-valley with respect to an axis* A if there exist $c_1, c_2, c_3 \in C$ such that $c_1 < c_2 < c_3$, $c_2 \prec c_1$ and $c_2 \prec c_3$.

The definition of v-valleys suffices to define single-peakedness for profiles of total orders: A profile $\mathcal{P}$ of total orders is single-peaked with respect to A if no vote $V \in \mathcal{P}$ contains a v-valley with respect to A (and thus every vote has only a single "peak"). A profile of total orders is single-peaked consistent (or simply, single-peaked) if there exists

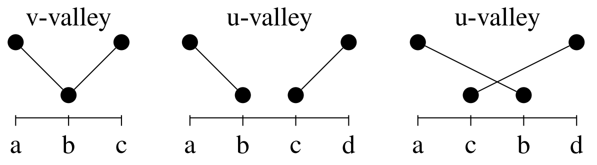

Figure 2: What v-valleys and u-valleys may look like.

some axis A such that $\mathcal{P}$ is single-peaked with respect to A.

We now want to extend this definition to profiles of partial orders. The natural way is to consider extensions of partial orders to total orders:

Definition 2. Let $\mathcal{P} = (V_1, \dots, V_n)$ be a profile of partial orders. The profile $\mathcal{P}$ is single-peaked with respect to an axis A if for every $k \in \{1, \dots, n\}$, V_k can be extended to a total order V'_k such that the profile of total orders $\mathcal{P}' = (V'_1, \dots, V'_n)$ is single-peaked with respect to A.

While it is also conceivable to require that every extension is single-peaked, this would yield an extremely restrictive definition. In this sense, our definition seems to be preferable.Next, we want to find an equivalent definition based on valleys, for which we require also u-valleys:

Definition 3 (u-valleys)**.** Let V be a partial order on C. The vote V *contains a u-valley with respect to* A if there exist distinct $a, b, c, d \in C$ with $a < b < d$ and $a \succ b$ as well as $a < c < d$ and $d \succ c$.

In Figure 2 a graphical representation of v- and u-valleys is shown. These two types of valleys allow a characterization of single-peakedness for profiles of partial orders.

Lemma 4. *Let $\mathcal{P} = (V_1, \dots, V_n)$ be a profile of partial orders. The following two statements are equivalent.*

(i) The profile $\mathcal{P}$ is single-peaked with respect to A.

(ii) No vote $V \in \mathcal{P}$ contains either a u-valley or v-valley with respect to A.

This lemma immediately yields a polynomial time algorithm for checking whether an incomplete profile is single-peaked with respect to a given axis:

Theorem 5. *Checking whether a profile of partial orders is single-peaked with respect to a given axis can be done in $\mathcal{O}(n \cdot m^4)$ time.*

Proof. For every quadruple of candidates and every vote, one has to check whether a u- or v-valley arises. □

Let $\mathcal{T} \in$ {partial order, local weak order, weak order, top order, total order} be a type of order. In this paper we are going to study the following problem:

$\mathcal{T}$ SINGLE-PEAKED CONSISTENCY
Instance: A profile $\mathcal{P}$ of type $\mathcal{T}$, a set of candidates C.
Question: Is $\mathcal{P}$ single-peaked consistent?

Note that in contrast to Theorem 5, the input of this problem does not include an axis. The TOTAL ORDER SINGLE-PEAKED CONSISTENCY problem is known to be solvable in polynomial time (Bartholdi and Trick 1986; Doignon and Falmagne 1994; Escoffier, Lang, and Öztürk 2008). In the next section, we show that this is likely not to be the case for partial orders and even local weak orders.

4 Hardness Results

Theorem 6. *The* LOCAL WEAK ORDER SINGLE-PEAKED CONSISTENCY *problem is* NP*-complete.*

Proof. We reduce from the NP-complete BETWEENNESS problem (Opatrny 1979). A BETWEENNESS instance consists of a finite set S and a set T containing (ordered) triples of distinct elements of S. The decision problem asks whether there is a total order L such that for every triple $(a, b, c) \in T$ we have either $aLbLc$ or $cLbLa$. Intuitively, a triple $(a, b, c) \in T$ corresponds to the constraint that b has to lie "in between" a and c on the total order L.

We construct an incomplete election $(C, \mathcal{P})$ with $C = S$, i.e., we identify elements in S with candidates. The preference profile $\mathcal{P}$ consists of two votes for each triple (a, b, c): the partial orders $\{a \succ c, b \succ c\}$ and $\{b \succ a, c \succ a\}$. These two votes form a valley on any axis with c between a and b and on any axis with a between b and c. Thus b has to be between a and c on any single-peaked axis. □

Corollary 7. *The* PARTIAL ORDER SINGLE-PEAKED CONSISTENCY *problem is* NP*-complete.*

The proof of Theorem 6 uses elections where the votes contain very little information: only two pairs of candidates are comparable in each vote. We know that determining single-peaked consistency is possible in polynomial time if every vote is a total order, i.e., all votes contain complete information. Now the question arises: what happens if only a single voter provides complete information? Having a single completely specified vote has been found to be helpful in a related context: it allows to efficiently elicit single-peaked preferences using only few comparison queries (Conitzer 2009) and thus reduces the communication complexity of preference elicitation. However, in our case such a voter does not provide enough additional information for a decrease in (computational) complexity.

Theorem 8. *The* PARTIAL ORDER SINGLE-PEAKED CONSISTENCY *problem is* NP*-complete even if the preference profile contains a total order.*

Proof. We reduce from SET SPLITTING: Let X be a finite set. Given a collection Z of subsets of X, is there a partition of X into two subsets X_1 and X_2 such that no subset of Z is contained entirely in either X_1 or X_2? This problem is NP-complete even if the sets in Z have cardinality 3.

Let $X = \{c_1, \dots, c_m\}$. For the construction, we identify the elements of X with candidates and add an additional candidate x. For each set $\{c_i, c_j, c_k\} \in Z$ with $i < j < k$ we introduce one vote: $\{c_i \succ c_j, x \succ c_k\}$. In addition, we add the vote $x \succ c_m \succ \dots \succ c_1$. One can show that the resulting preference profile $\mathcal{P}$ is single-peaked if and only if (X, Z) is a SET SPLITTING yes-instance. The key observation is that, on A, x separates the sets X_1 and X_2. □

It is important to note that – in contrast to the NP-hardness result in Theorem 6 – in this proof we make use of u-valleys instead of v-valleys. This means in particular that this hardness result does not hold for weak orders, which cannot contain u-valleys. This is not incidental: in the next section, we present a polynomial-time algorithm for weak orders.

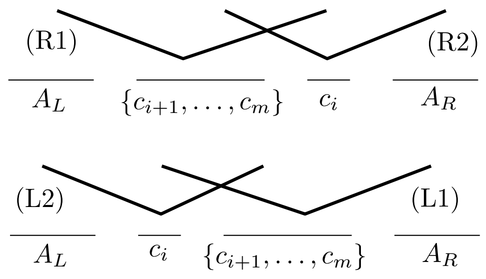

Figure 3: Graphical representation of the conditions testing whether c_i can be placed on the right-hand side ((R1), (R2)) or on the left-hand side ((L1), (L2))

5 The Guided Algorithm

In this section, we present a polynomial time algorithm for profiles of weak orders. This algorithm requires that the profile contains at least one total order to guide the placement of candidates on the axis. We call this vote the *guiding vote*.

Theorem 9. *If the profile contains a total order, the* WEAK ORDER SINGLE-PEAKED CONSISTENCY *problem can be solved in $\mathcal{O}(m \cdot n)$ time.*

We will refer to Algorithm 1, which Theorem 9 is based on, as the *Guided Algorithm*. Without loss of generality, we assume that the guiding vote is $\langle c_m \succ c_{m-1} \succ \dots \succ c_1 \rangle$, i.e., we number the candidates based on the guiding vote.

The algorithm has a simple structure: The lowest ranked candidate in the guiding vote, c_1, is placed on the rightmost position of the axis (The leftmost position would work as well.) Starting with the second lowest ranked candidate, c_2, in the guiding vote, the candidates are successively placed on the axis – either at the leftmost or rightmost still available position. The lists A_L and A_R correspond to the left-hand and right-hand side of the axis. For each candidate, we test whether it can be placed on the right-hand side or left-hand side without creating a valley. If only one of these options is viable, the candidate is placed accordingly. If both left and right are possible, we place the candidate arbitrarily right. If neither is possible, the preference profile is not single-peaked.

Testing whether a vote V_k imposes restrictions on the placement of a candidate is achieved by four conditions. These conditions distinguish four categories of candidates: candidates in A_R, candidates in A_L, candidates that have not yet been placed ($C_{>i} = \{c_{i+1}, \dots, c_m\}$) and the candidate that is currently under consideration (c_i). We are only checking for valleys that include c_i. This gives rise to these four conditions: (R1) and (R2) test whether placing c_i on the right-hand side of the already placed candidates leads to valleys, (L1) and (L2) do the same for the left-hand side. Refer to Figure 3 for a graphical representation. Since we only consider weak orders, we do not have to consider every candidate triple possibly fulfilling these conditions but have to check only maximal or minimal candidates. More specifically, checking whether there is a candidate $c \in A_L$ and $c' \in C_{>i}$ with $c \succ c'$ is equivalent to whether any maximal element in A_L is preferred to some minimal element in $C_{>i}$. Let $\mathit{min}_k(X)$ denote a function that picks one element in X that is minimal with respect to $\succ_k$. The function $\mathit{max}_k(X)$ is defined analogously.

$$c_i \succ_k \mathit{min}_k(C_{>i}) \text{ and } \mathit{max}_k(A_L) \succ_k \mathit{min}_k(C_{>i}) \quad \text{(R1)}$$
$$\mathit{max}_k(C_{>i}) \succ_k c_i \text{ and } \mathit{max}_k(A_R) \succ_k c_i \quad \text{(R2)}$$
$$c_i \succ_k \mathit{min}_k(C_{>i}) \text{ and } \mathit{max}_k(A_R) \succ_k \mathit{min}_k(C_{>i}) \quad \text{(L1)}$$
$$\mathit{max}_k(C_{>i}) \succ_k c_i \text{ and } \mathit{max}_k(A_L) \succ_k c_i \quad \text{(L2)}$$

Using these four definitions, we can give a succinct description of the algorithm (Algorithm 1).

Algorithm 1: The Guided Algorithm

```
$A_L \leftarrow \langle\rangle$; $A_R \leftarrow \langle c_1 \rangle$
for $i \leftarrow 2 \dots m$ do
    right $\leftarrow$ true; left $\leftarrow$ true
    for $k \leftarrow 2 \dots n$ do
        if Condition (R1) or (R2) holds then
            right $\leftarrow$ false
        if Condition (L1) or (L2) holds then
            left $\leftarrow$ false
    if right $=$ true then
        $A_R \leftarrow \langle c_i < A_R \rangle$
    else
        if left $=$ true then
            $A_L \leftarrow \langle A_L < c_i \rangle$
        else
            return not_single_peaked
return $A_L < A_R$
```

Theorem 9 claims that the Guided Algorithm requires $\mathcal{O}(m \cdot n)$ time. This is only possible if the conditions can be checked in constant time. Thus, the minima and maxima have to be computable in constant time. For $\mathit{max}_k(A_L)$ and $\mathit{max}_k(A_R)$ this is easily possible by storing and updating these two values. If c_i is placed left, we update $\mathit{max}_k(A_L)$ in case c_i is the new maximum (with respect to $\succ_k$); if c_i is placed right, we proceed analogously $\mathit{max}_k(A_R)$. For computing a minimal value of $C_{>i}$, observe that the set $C_{>i}$ becomes smaller with increasing i. Thus, a minimal value of $C_{>i}$ might disappear at some point and a new (larger) value has to be chosen. The new minimum is the smallest element (with respect to $\succ_k$) in $C_{>i}$ that is at least as large as the old minimum. If we maintain pointers to the minimum elements, the amortized cost of this update procedure is $\mathcal{O}(1)$. A maximal value of $C_{>i}$ can be found analogously.

We conclude this section with a lemma showing that we can weaken the total order requirement: it suffices that the guiding vote is given implicitly in the profile.

Lemma 10. *Let $C = \{c_1, c_2, \dots, c_m\}$ and $T = \langle c_1 < c_2 < \dots < c_m \rangle$ be a total order on C with the following property: for each $i \in \{1, \dots, m\}$, it holds that there is a vote $V \in \mathcal{P}$ such that c_i is the unique last ranked candidate in $V[\{c_i, c_{i+1}, \dots, c_m\}]$. If $\mathcal{P}$ is a single-peaked profile, then $\mathcal{P}$ is also single-peaked if the total order T is added to it as a vote.*

It is computationally easy to find such an implicitely given guiding order: Look for a vote with a unique last ranked candidate. This candidate is ranked last in the guiding vote. Remove this candidate from the profile and repeat this step to obtain the second-to-last element in the guiding order, etc.

6 A 2-SAT Based Algorithm

Theorem 8 and Theorem 9 leave open the case of profiles of local weak orders which contain at least one total order. Here, we show that this case is polynomial time solvable as well.

Theorem 11. *If the profile contains a total order, the* LOCAL WEAK ORDER SINGLE-PEAKED CONSISTENCY *problem can be solved in $\mathcal{O}(n \cdot m^3)$ time.*

We encode the LOCAL WEAK ORDER SINGLE-PEAKED CONSISTENCY instance in a 2-SAT instance. The 2-SAT problem asks whether a Boolean formula of the form $(a \vee b) \wedge (\neg a \wedge c) \wedge \ldots$ (each clause has size two) is satisfiable. Solving 2-SAT requires only linear time (Aspvall, Plass, and Tarjan 1979). The boolean variables in our instance corresponds to pairs of candidates, i.e., for each $a, b \in C$ we have a variable ab. The intended meaning of these variables is that $ab = \texttt{true}$ if and only if a is left of b on the axis. Now, for each vote V and triple $a, b, c \in C$, if $a \succ b$ and $c \succ b$ (a, b, c form a v-valley), then we add the clauses $(ba \vee cb)$ and $(ab \vee bc)$ to the 2-SAT instance. These clauses correspond the requirement that b must be placed between a and c. Finally, we add for each pair of variables a, b the clauses $(ab \vee ba)$ and $(\neg ab \vee \neg ba)$ (corresponding to the exclusive or operator). Solving the 2-SAT instance either yields the information that the instance is not satisfiable or a true/false assignment to the variables. In the first case, the profile is not single-peaked. In the second case, we obtain a relation $A = \{(a, b) : ab = \texttt{true}\} \cup \{(a, a) : a \in C\}$ which is our wanted axis. Indeed, one can show that A is a total order and the profile is single-peaked with respect to A. We remark that the transitivity of A is due to the guiding vote. Since the instance contains at most $\mathcal{O}(n \cdot m^3)$ clauses, we obtain the stated runtime.

Both the 2-SAT based algorithm and the Guided Algorithm rely on the guiding vote. In the next section, we will consider profiles that do not have a guiding vote.

7 The Unguided Algorithm

Here, we present a polynomial-time algorithm (Algorithm 2) that, in contrast to the Guided Algorithm, is not dependent on a guiding vote. We therefore refer to it as the Unguided Algorithm. The Unguided Algorithm is applicable to top orders. We assume the input preference profile to be *connected*: Let us consider the ranked candidates in a top order to be a hyperedge of a hypergraph with candidates as vertices. A profile of top orders is called connected if this graph has only one connected component. This assumption does not limit the applicability: if two or more connected components exist in this graph, we can use the algorithm for each component (i.e., its respective candidates and votes) and concatenate the resulting axes in arbitrary order.

The algorithm works as follows: First, we choose a candidate c_{start} which is going to be the leftmost candidate on the axis A. Since we have no guiding vote, each candidate

Algorithm 2: The Unguided Algorithm

```
1  foreach $c_{start} \in C$ do
2      $A \leftarrow \langle c_{\text{start}} \rangle$
3      for $i \leftarrow 1 \ldots m$ do
4          foreach $V \in \texttt{VotesWithPeak}(a_i)$ do
5              if $A \oplus V = incompatible$ then
6                  Continue with next $c_{\text{start}} \in C$ in
                   Line 1.
7              else $A \leftarrow A \oplus V$
8          if $|A| = i$ and $i < m$ then
9              $V \leftarrow \texttt{IntersectingVote}(A)$
10             if $a_i \notin V$ then
11                 Continue with next $c_{\text{start}} \in C$ in
                   Line 1.
12             Let $x$ be new candidate not in $C$.
13             $C' \leftarrow \{c \in V \mid c \succ a_i\} \cup \{a_i, x\}$
14             $S \leftarrow \emptyset$
15             for $k \leftarrow 1 \ldots n$ do
16                 $V'_k \leftarrow \texttt{RepTop}(V_k, C \setminus (A \cup C'), x)$
17                 $S \leftarrow S \cup \{V'_k[C']\}$
18             $A' \leftarrow \texttt{GuidedSP}(S, V[C'], a_i, x)$
19             if $A' = \texttt{not\_single\_peaked}$ then
20                 Continue with next $c_{\text{start}} \in C$ in
                   Line 1.
21             else $A \leftarrow A < A'[C' \setminus \{x\}]$
22     return $A$
23 return $\texttt{not\_single\_peaked}$
```

might be placed at the leftmost position. Hence we loop over all candidates (Line 1). The corresponding axis under construction is $A = \langle c_{\text{start}} \rangle$. We now aim to complete this axis by adding candidates to the right in such a way that all votes are single-peaked with respect to this axis. To this end we employ the loop in Line 3. In this loop (variable i) we infer from the already placed candidate a_i (the i-th candidate on A from left) the candidate a_{i+1} (or even more candidates further to the right).

The Lines 4 to 7 are based on the following observation: Let us assume that at a certain point $A = \langle c_1 < c_2 < c_3 \rangle$ and $V = \langle c_3 \succ c_2 \succ c_4 \succ c_5 \succ \bullet \rangle \in \mathcal{P}$. Since c_3, the peak of V, is already contained in A, there is only one compatible extension of A: $\langle c_1 < c_2 < c_3 < c_4 < c_5 \rangle$. We formalize this extension operation with the $\oplus$ operator:

Definition 12. Let A be an incomplete axis, V a top order and let $V[C \setminus A] = \langle c'_1 \succ c'_2 \succ \ldots \succ c'_j \rangle$. We define $A \oplus V = \langle A < c'_1 < c'_2 < \ldots < c'_j \rangle$ if V is single-peaked with respect to this axis and $A \oplus V = $ *incompatible* otherwise.

The loop in Line 4 enumerates all votes with peak a_i ($\texttt{VotesWithPeak}(a_i)$). Let $V \in \texttt{VotesWithPeak}(a_i)$. If $A \oplus V =$ *incompatible* then A cannot be extended to a complete (single-peaked) axis and we consider the next $c_{\text{start}} \in C$ in Line 1. Otherwise, we obtain a new incomplete axis $A \leftarrow A \oplus V$.

It might be the case that the candidate a_{i+1} has not yet been determined after these steps. The Lines 8 to 21 deal with this case. Since the election is connected there has to be at least one vote that contains both a candidate on A and a candidate that has not been placed yet. The procedure `IntersectingVote` in Line 9 returns such a vote V with $A \cap V \neq \emptyset$ and $V \setminus A \neq \emptyset$. For such a vote V it holds that peak$(V) \notin A$. If peak(V) were contained in A, then V would have been already considered in the first part of the algorithm (Lines 4 to 7). If V does not contain a_i (and thus a_i is ranked last in V), A cannot be extended to a single-peaked axis.

Now that we have such an intersecting vote V with $a_i \in V$, we employ the Guided Algorithm to find a further extension of A. The main idea is to use V as a guiding vote and find an axis for the candidates in $\{c \in V \mid c \succ a_i\}$. In principle, this axis can be found independently of the existing axis A. However, the leftmost and rightmost candidates have to be chosen with regard to "external" considerations: The leftmost candidate has to be a_i, otherwise A and the newly obtain partial axis A' could not be merged. For the rightmost candidate, we have to consider votes with candidates that are not being placed on the axis in this step. The following example illustrates the issue.

Example. Let $A = \langle c_1 \rangle$, $V_1 = \langle c_2 \succ c_3 \succ c_1 \succ \bullet \rangle$ and $V_2 = \langle c_3 \succ c_4 \succ \bullet \rangle$. The vote V_1 intersects A and hence $C' = \{c_1, c_2, c_3\}$. We employ the Guided Algorithm and might obtain $A' = \langle c_1 < c_3 < c_2 \rangle$.[1] Now observe that $A \oplus A' = A'$ can no longer be extended in a way that it is single-peaked for V_2. This would have been possible if c_3 had been chosen as the rightmost candidate in A'.

As we see from this example, we sometimes have to "force" the rightmost candidate in A'. We do this by adding an additional candidate x to every vote (Line 15 to 17). It is placed at the position of the top ranked candidate in each vote that is not contained in $A \cup C'$. This is done by the `RepTop` function: `RepTop`(V, D, x) replaces the one candidate in vote V that is the top ranked of the candidates in D with candidate x. By forcing this element x to be the rightmost candidate, we ensure that A' is chosen under consideration of all votes with ranked candidates not in C'.

Example (continued)**.** We apply `RepTop` to the votes V_1 and V_2 with $C' = \{c_1, c_2, c_3, x\}$. The vote $V_1'[C'] = \langle c_2 \succ c_3 \succ c_1 \succ x \rangle$ and $V_2'[C'] = \langle c_3 \succ x \succ \bullet \rangle$. Now, we can only obtain the axis $\langle c_1 < c_2 < c_3 < x \rangle$.

The set S, as computed in Lines 14 to 17, is the election $\mathcal{P}$ restricted to C' together with the additional candidate x. We now employ `GuidedSP` $(S, V[C'], A', a_i, x)$ which means that we employ the Guided Algorithm for the profile S and guiding vote $V[C']$. Furthermore, we require that the leftmost candidate on the axis is a_i and the rightmost is x. The function `GuidedSP` either returns `not_single_peaked` or an axis A'. If it returns `not_single_peaked`, the next $c_{\text{start}} \in C$ is considered (Line 1). Otherwise, we continue with the extended axis $A \leftarrow A \oplus A'[C' \setminus \{x\}]$.

[1]Whether we obtain this axis or $\langle c_1 < c_2 < c_3 \rangle$ depends on whether the algorithm prefers placing candidates to the left or to the right if both choices are possible.

X	general	guiding vote
Partial	NP-c (Cor 7)	NP-c (Thm 8)
Local Weak	NP-c (Thm 6)	poly (Thm 11)
Weak	open	poly (Thm 9)
Top	poly (Thm 13)	poly (Thm 9)
Total	poly[†]	poly[†]

Table 1: Overview of the complexity results for X Order Single-Peaked Consistency

Theorem 13. *The* Top Order Single-peaked Consistency *problem can be solved in* $\mathcal{O}(m^2 \cdot n)$ *time.*

8 Conclusions

In this paper we have analyzed the Partial Order Single-peaked Consistency problem (see Table 1) for different types of order and with/without the assumption of having a guiding vote. Such a vote is likely to exist for large preference profiles. In the case that top orders are elicited however, a guiding vote might not exist. Here the Unguided Algorithm is applicable. We therefore believe to have succeeded in covering a large spectrum of possible application scenarios with our algorithms.

We would like to mention one particular application of the Unguided Algorithm concerning single-peaked scoring protocols. Scoring protocols are specified by a scoring vector $(\alpha_1, \dots, \alpha_m)$. A vote $V = \langle c_1, \dots, c_m \rangle$ gives α_1 points to c_1, α_2 points to c_2, etc. The winner candidate is determined by summing over all votes. Often scoring vectors of the type $(\alpha_1, \dots, \alpha_k, 0, \dots, 0)$ with $\alpha_1 > \dots > \alpha_k > 0$ are considered. For such scoring rules, top orders (with k ranked candidates) constitute *full* information. It is therefore debatable whether the input may be considered to be given as a profile of total orders. This is of relevance for single-peaked profiles. For example, Brandt et al. (2010) study the constructive coalition weighted manipulation problem for scoring protocols in single-peaked elections. The authors consider the axis to be part of the input (for good reasons as explained in their paper). The computation of such an axis with existing algorithms is possible only if preferences are specified by total orders and thus contain problem-irrelevant information. If only relevant information is given, i.e., the input consists of top orders, an algorithm such as the Unguided Algorithm is required.

The most important future research direction is to investigate the computational advantages arising from incomplete single-peaked preferences: does such structure decrease the computational complexity of voting problems? In addition, it is desirable to extend our algorithms to nearly single-peaked domains (Faliszewski, Hemaspaandra, and Hemaspaandra 2011; Cornaz, Galand, and Spanjaard 2012; Erdélyi, Lackner, and Pfandler 2013; Elkind, Faliszewski, and Slinko 2012; Bredereck, Chen, and Woeginger 2013; Sui, Nienaber, and Boutilier 2013), since notions of nearly single-peakedness are more robust and applicable in real-life settings.

[†]This is a result by Bartholdi and Trick (1986).

Acknowledgements

This work was supported by the Austrian Science Fund (FWF): P25518-N23 and by the Vienna Science and Technology Fund (WWTF) project ICT12-15.

References

Aspvall, B.; Plass, M. F.; and Tarjan, R. E. 1979. A linear-time algorithm for testing the truth of certain quantified Boolean formulas. *Information Processing Letters* 8(3):121–123.

Bartholdi, J. J., and Trick, M. 1986. Stable matching with preferences derived from a psychological model. *Operations Research Letters* 5(4):165 – 169.

Baumeister, D., and Rothe, J. 2012. Taking the final step to a full dichotomy of the possible winner problem in pure scoring rules. *Information Processing Letters* 112(5):186 – 190.

Baumeister, D.; Faliszewski, P.; Lang, J.; and Rothe, J. 2012. Campaigns for lazy voters: truncated ballots. In *Proc. of AAMAS-12, Volume 2*, 577–584.

Betzler, N., and Dorn, B. 2010. Towards a dichotomy for the possible winner problem in elections based on scoring rules. *Journal of Computer and System Sciences* 76(8):812–836.

Black, D. 1948. On the rationale of group decision making. *Journal of Political Economy* 56(1):23–34.

Brandt, F.; Brill, M.; Hemaspaandra, E.; and Hemaspaandra, L. A. 2010. Bypassing combinatorial protections: Polynomial-time algorithms for single-peaked electorates. In *Proc. of AAAI-10*, 715–722.

Bredereck, R.; Chen, J.; and Woeginger, G. J. 2013. Are there any nicely structured preference profiles nearby? In *Proc. of IJCAI-13*.

Conitzer, V. 2009. Eliciting single-peaked preferences using comparison queries. *Journal of Artificial Intelligence Research* 35:161–191.

Cornaz, D.; Galand, L.; and Spanjaard, O. 2012. Bounded single-peaked width and proportional representation. In *Proc. of ECAI-12*, volume 242 of *FAIA*, 270–275. IOS Press.

Doignon, J.-P., and Falmagne, J.-C. 1994. A polynomial time algorithm for unidimensional unfolding representations. *Journal of Algorithms* 16(2):218–233.

Dwork, C.; Kumar, R.; Naor, M.; and Sivakumar, D. 2001. Rank aggregation methods for the web. In *Proc. of WWW-01*, 613–622. ACM.

Elkind, E.; Faliszewski, P.; and Slinko, A. M. 2012. Clone structures in voters' preferences. In *Proc. of EC-12*, 496–513. ACM.

Erdélyi, G.; Lackner, M.; and Pfandler, A. 2013. Computational aspects of nearly single-peaked electorates. In *Proc. of AAAI-13*. AAAI Press.

Escoffier, B.; Lang, J.; and Öztürk, M. 2008. Single-peaked consistency and its complexity. In *Proc. of ECAI-08*, volume 178 of *FAIA*, 366–370. IOS Press.

Fagin, R.; Kumar, R.; Mahdian, M.; Sivakumar, D.; and Vee, E. 2006. Comparing partial rankings. *SIAM Journal on Discrete Mathematics* 20(3):628–648.

Fagin, R.; Kumar, R.; and Sivakumar, D. 2003. Comparing top k lists. *SIAM Journal on Discrete Mathematics* 17(1):134–160.

Faliszewski, P.; Hemaspaandra, E.; Hemaspaandra, L. A.; and Rothe, J. 2011. The shield that never was: Societies with single-peaked preferences are more open to manipulation and control. *Information and Computation* 209(2):89–107.

Faliszewski, P.; Hemaspaandra, E.; and Hemaspaandra, L. A. 2011. The complexity of manipulative attacks in nearly single-peaked electorates. In *Proc. of TARK-11*, 228–237.

Hemaspaandra, E.; Hemaspaandra, L. A.; and Rothe, J. 1997. Exact analysis of Dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP. *Journal of the ACM* 44(6):806–825.

Hemaspaandra, E.; Spakowski, H.; and Vogel, J. 2005. The complexity of Kemeny elections. *Theoretical Computer Science* 349(3):382–391.

Konczak, K., and Lang, J. 2005. Voting procedures with incomplete preferences. In *Proc. of MPref-05*.

Mattei, N., and Walsh, T. 2013. Preflib: A library of preference data. In *ADT-13)*, LNCS. Springer.

Opatrny, J. 1979. Total ordering problem. *SIAM Journal on Computing* 8(1):111–114.

Pini, M. S.; Rossi, F.; Venable, K. B.; and Walsh, T. 2011. Incompleteness and incomparability in preference aggregation: Complexity results. *Artificial Intelligence* 175(7-8):1272 – 1289.

Sui, X.; Nienaber, A.; and Boutilier, C. 2013. Multi-dimensional single-peaked consistency and its approximations. In *Proc. of IJCAI-13*.

Walsh, T. 2007. Uncertainty in preference elicitation and aggregation. In *Proc. of AAAI-07*, 3–8. AAAI Press.

Xia, L., and Conitzer, V. 2011. Determining possible and necessary winners under common voting rules given partial orders. *Journal of Artificial Intelligence Research* 41(2):25–67.