Testing a Mixture Model of Single-Peaked Preferences

Smeulders, B.[*]

Abstract

Single-Peaked preferences play an important role in the social choice literature. In this paper, we look at necessary and sufficient conditions for aggregated choices to be consistent with a mixture model of single-peaked preferences for a given ordering of the alternatives. These conditions can be tested in time polynomial in the number of choice alternatives. In addition, algorithms are provided which identify the underlying ordering of choice alternatives if the ordering is unknown. These algorithms also run in polynomial time, providing an efficient test for the mixture model of single-peaked preferences.

1 Introduction

Preferences play an important role in many areas of research. When faced with different alternatives, be it different cars, candidates in an election, budgets, etc., it is commonly assumed that people have a preference ordering over all of these alternatives, ranking them from best to worst. Often the nature of the alternatives restricts the possible preferences. An important restriction is given by *single-peakedness*, introduced by Black [3]. Suppose a linear ordering exists, which ranks all alternatives along a line. An agent's preferences are then single-peaked if she has a most preferred alternative, the *peak*, and when comparing two alternatives that are both on the same side of the peak, the alternative closest to the peak is preferred. This restriction is very natural when considering a situation where a single attribute of the alternatives drives the choice, for example, an election where candidates range from left to right wing or choices over budgets of various sizes. Given these examples, it is no wonder that this restriction has gained central importance in the areas of political science and social choice. Apart from being an appealing model in these areas, the assumption of single-peaked preferences has led to interesting theoretical results. For example, aggregation of single-peaked preferences avoids the Condorcet paradox [3].

Given the importance of single-peaked preferences, it is of interest to test if and in which situations agents hold such preferences. A number of algorithmic papers have appeared on the subject. Given the complete preference profile of agents Bartholdi and Trick [2] provide a polynomial time algorithm to test whether these are single-peaked with respect to some ordering of the alternatives and to identify this ordering. Doignon and Falmagne [10] provides a different algorithm for this problem with a better worst-case bound. Escoffier et al. [15] rediscovered this algorithm and give a detailed description. Ballester and Haeringer [1] give two forbidden substructures, whose absence is a necessary and sufficient condition for the given preference profile to be consistent with single-peakedness. Puppe [26] gives a global characterization of the single-peaked domain. Furthermore, Trick [33] provides an algorithm for recognizing single-peakedness on trees, which again runs in polynomial time. Doignon and Falmagne [10], Knoblauch [19] and Elkind and Falizewski [13] investigate a closely related preference restriction, one-dimensional Euclidean preference profiles, and all provide polynomial time algorithms. Preference profiles

[*]QuantOM, HEC Management School, Université de Liège. The author is a post-doctoral fellow of the F.R.S-FNRS.

that are nearly-single peaked are investigated by Erdélyi et al.[14] and Bredereck et al.[6]. Testing for single-peakedness given real-world preference orderings is done by List et al. [20] and Sui et al.[32].

One common factor in these papers is that they test for single-peakedness given full preference profiles of the agents. However, there are several drawbacks to using this kind of data. First, it assumes that choices are made based on preference orderings. Agents who use a different decision rule, for example heuristics, can not accurately report their decision rule. Second, even if the assumption of preference orders is correct, there is a high risk that the ordering is misreported. This can be due to a simple error, as ranking all alternatives is a complex task, or because the agent has little incentive to determine her ranking over average alternatives.

The dominant setting for experiments in choice behaviour research is two-alternative forced choice (see Luce [21], Block and Marschak [4]). In this setting, agents are faced with choices between two alternatives, and must choose one. A large number of such choices are given, including repetitions of the same choice situations. This simple choice situation reduces the risk of agents misreporting. In the simplest situation, agents will consistently make the same choice when faced with repetitions of choice situations. However, it is very common that agents exhibit choice reversals. The data resulting from such experiments is thus, for every pair of alternatives, a rate with which one alternative is chosen over another. In this paper, it is our goal to provide a way of testing for single-peakedness in this setting, continuing a line of research started by Dridi [12].

To test for single-peaked preferences, while being able to account for choice reversals, we will make use of a 'random preference model' or 'mixture model', which can be traced back to Block and Marschak [4]. Such a model considers all possible ways a 'core theory', here single-peaked preferences, can be satisfied. It states that at any given point in time, the agent has a single-peaked preference, but that these may be different single-peaked preferences at different points in time. The data is consistent with this model if there exist single-peaked preferences, and a probability distribution over these preferences, such that the probability that the agent holds a preference in which she prefers one alternative over another is equal to the rate at which she chooses that alternative over the other. Notice that this model may also be used on the level of a population. The rate at which a population chooses one alternative over another can be determined by polling. The probability function then represents the probability that a member of the population holds a particular preference. The conditions such mixture models impose on data have been studied for a number of different classes of decision rules. It should be noted that aggregation of preferences in this way may lead to artifacts complicating analysis of the data (Davis-Stober et al.[9]). There is an equivalence between testing mixture models and the membership problem of a polytope associated with the class of preference being studied. The preference orderings can be encoded as points, and all convex combinations of these points are consistent with a mixture model (see, amongst others, Megiddo [23], Suck [31]). In the case of general strict preference orders, the mixture model corresponds to random utility models, which were studied by, amongst many others, Block and Marschak [4] and Dridi [11]. The associated polytope is the complex and well-studied linear ordering polytope [22]. Weak orders were studied by Fiorini and Fishburn [18], bi-orders by Christophe et al. [7] and a class of lexiographic semiorders by Davis-Stober [8]. Most importantly, Dridi [12] worked on single-peaked preferences, providing necessary and sufficient conditions for testing mixture models of single-peaked preferences given an ordering of the alternatives.

The main contributions of this paper are as follows.

- Given an ordering of the alternatives and the rates of choice, necessary and sufficient conditions for testing a mixture model of single-peaked preferences are described. These conditions can be tested in $O(n^2)$ time, where n is the number of choice alternatives. This result corresponds to the work of Dridi [12]. The characterization and proof in this paper follow the same lines as Dridi's paper, but may be of use to non-French speakers.
- We provide a polynomial time algorithm which given the rates of choice, provides an ordering of the alternatives for which a mixture model of single-peaked preferences is satisfied (if such an ordering exists). This algorithm also runs in $O(n^2)$.

During the review process of this paper, Spanjaard and Weng [30] published a paper that is closely related to this work. Given a net preference matrix (i.e., a matrix M where element M_{ij} represents the net difference between the number of agents preferring alternative i over alternative j and the number of agents preferring j over i), they provide necessary and sufficient conditions for the net preferences to be consistent with a group of agents with single-peaked preference with respect to some ordering of the alternatives. This condition on the net preferences is the same as the condition on the rates of choice for consistency with a mixture model. Spanjaard and Weng also provide an algorithm for identifying an ordering of alternatives for which the net preference matrix satisfies the conditions, by making use of the consecutive ones algorithm by Booth and Lueker [5]. This algorithm can also be applied to the problem considered in this paper. However, the algorithm given by Spanjaard and Weng runs in $O(n^3)$ as opposed to the $O(n^2)$ time algorithm described in this paper.

The rest of this paper is organized as follows. In section 2, we give formal definitions of single-peaked preferences and the mixture model. Sections 3 and 4 contain the main results. First, necessary and sufficient conditions for a mixture model of single-peaked preferences are described in section 3. Next, an algorithm to identify the underlying ordering of alternatives is described in section 4. In section 5, we briefly investigate the power of the tests described. We show that it is highly restrictive and that there is little risk of false positives. In the final section, we conclude and discuss possible extensions to nearly-single-peaked preferences.

2 Notation and Definitions

Consider a set A, consisting of n alternatives, and a dataset $P = \{p_{ij} \geq 0, \forall (i,j) \in A^2\}$. The values p_{ij} represent the rate at which i is chosen over j. As we work with forced binary choice situations, $p_{ij} + p_{ji} = 1$ if $i \neq j$. We let $p_{ii} = 0$ for all $i \in A$. We consider linear preference orderings $\succ$, which are complete, asymmetric and transitive, over all alternatives. We will use the index m to denote a particular preference ordering. If for a given preference ordering m, an alternative i is preferred over another alternative j, we denote this by $i \succ_m j$. We also consider a (given) ordering of the alternatives in A. This ordering is complete, asymmetric and transitive and is denoted by $\rhd$.

Definition 1. *A preference ordering m is single-peaked with respect to a given ordering of the alternatives $\rhd$ if and only if for every triple $(i,j,k) \in A^3$ we have:*

$$\text{if } (i \rhd j \rhd k \text{ and } i \succ_m j) \text{ then } i \succ_m k. \tag{1}$$

$$\text{if } (i \rhd j \rhd k \text{ and } k \succ_m j) \text{ then } k \succ_m i. \tag{2}$$

The set of all preference orderings that are single-peaked with respect to an ordering $\rhd$ is denoted by $O^{\rhd}$. We further consider the subsets $O_{ij}^{\rhd}$, defined as follows: $m \in O_{ij}^{\rhd}$ if both $m \in O^{\rhd}$ and $i \succ_m j$. A mixture model of preference assumes that, a decision maker has a number of different preference orderings, each with an associated probability. When faced with a choice between alternatives i and j, the probability that the decision maker chooses i is equal to the sum of the probabilities of all preference orderings in which i is preferred over j.

Definition 2. *A dataset P can be rationalized by a mixture model of single-peaked linear ordering preferences with respect to a given ordering of alternatives $\rhd$ if and only if there exist numbers $x_m \geq 0, \forall m \in O^{\rhd}$ for which:*

$$\sum_{m \in O_{ij}^{\rhd}} x_m = p_{ij}, \qquad \forall (i,j) \in A^2. \tag{3}$$

3 Consistency conditions

The existence of a solution to the system of equalities (3) can be checked easily by verifying a condition on the p_{ij} values, as shown by Dridi [12]. We provide a reformulation of this condition and give a proof showing both the condition's sufficiency and necessity. We finish this section by showing that the condition may be tested in $O(n^2)$ time.

Theorem 1. *A dataset P can be rationalized by a mixture model of single-peaked preferences with respect to a given ordering $\rhd$ if and only if for every triple $(i,j,k) \in A^3$ we have:*

$$\text{if } i \rhd j \rhd k \text{ then } p_{ij} \leq p_{ik} \text{ and } p_{kj} \leq p_{ki}. \tag{4}$$

Before embarking on a proof of this theorem, let us first note that the condition (4) is similar, but subtly different from the conditions for Robinsonian dissimilarities [29, 24]. In the case of Robinsonian dissimilarities, there is a set of objects and a dissimilarity score (d_{ij}) for each pair of objects. Objects must be ranked in such a way that for any given triple of objects, the dissimilarity between the middle object and an outer object is not greater than the dissimilarity between the two outer objects. The main differences are that, for Robinsionian dissimilarities the values d_{ij} are symmetric, i.e, $d_{ij} = d_{ji}$ (and there is no constraint on $d_{ij} + d_{ji}$). Furthermore, a dissimilarity is Robinsonian with respect to an order if and only if for every triple $(i,j,k) \in A^3$, with $i \rhd j \rhd k$, it must be the case that $d_{ij} \leq d_{ik}$ and $d_{jk} \leq d_{ik}$.

We also note that by encoding a preference order as follows, set $e_{ij} = 1$ if $i \succ j$ and $e_{ij} = 0$ otherwise, the encoding of any preference order satisfying conditions (1)-(2) satisfies condition (4). Thus, the convex combination of these single-peaked preference orders must also satisfy condition (4). While we will formally argue the necessity later on, it is thus clear that if condition (4) is violated, at least part of the population has to hold preferences that violate either condition (1) or (2). However, the sufficiency of this condition is not as straightforward. Indeed, if we look at mixture models with general preferences, we find that the number of necessary and sufficient conditions is exponential in the number of alternatives (see [17, 22]). This is the case, even though general preference orderings are constrained only by transitivity, which can also be defined by a condition over all triples.

Proof. First, we will show sufficiency of the condition. We will show that if there exist numbers x_m satisfying the equalities (3) for all $(i,j) \in A^2$ with $i \rhd j$, then a solution exists for the complete set of equalities. If $\sum_{m \in O^{\rhd}} x_m = 1$, then this is already the case. Otherwise, for the order l

obtained by reversing $\rhd$, we increase x_l by $1 - \sum_{m \in O^{\rhd}} x_m$. Therefore, we must only show a solution exists for the pairs of alternatives $(i, j) \in A^2$ with $i \rhd j$.

Now suppose condition (4) is satisfied for every triple of alternatives. We will constructively show numbers x_m exist satisfying (3) for all $(i, j) \in A^2$ with $i \rhd j$. We begin by sorting all numbers p_{ij}, with $i \rhd j$ from smallest to largest (ties are broken arbitrarily). We denote the smallest such number by $p(1)$, the second smallest $p(2)$ and so on. For each number $p(l)$ we construct preference relations $\succ_l$ as follows. For each pair of alternatives $(i, j) \in A^2$ with $i \rhd j$, if $p_{ij} \geq p(l)$, then $i \succ_l j$, otherwise $j \succ_l i$. By construction this relation is complete and asymetric. We now show that they are single-peaked linear orders. Consider Condition (1), which requires that if $i \rhd j \rhd k$ and $i \succ_l j$, it must be the case that $i \succ_l k$. Assume $p_{ij} \geq p(l)$, and thus $i \succ_l j$. Since we assume Condition (4) is satisfied, we have $p_{ik} \geq p_{ij} \geq p(l)$, and thus $i \succ_l k$, satisfying condition (1). The same argument holds true for condition (2). Since the relation $\succ_l$ is complete and asymmetric by construction, and single-peakedness ensures transitivity of $\succ_l$, it is a single-peaked linear order. Next, we set the numbers x_m as follows, $x_l = p(l) - p(l-1)$, with $p(0) = 0$.

It can now easily be checked that the orders and numbers form a solution satisfying equalities (3) for all $(i, j) \in A^2$ with $i \rhd j$. Consider any such pair, without loss of generality, let $p_{ij} = p(l)$. Then, for every l' with $p(l') \leq p(l)$, it is the case that $i \succ_{l'} j$, i.e. $l' \in O^{\rhd}_{ij}$. It follows that $\sum_{m \in O^{\rhd}_{ij}} x_m = \sum_{m=1}^{l} x_m = \sum_{m=1}^{l} p(m) - p(m-1) = p(l) = p_{ij}$. Since the existence of numbers satisfying the equalities (3) for all $(i, j) \in A^2$ with $i \rhd j$ implies the existence of a solution for all $(i, j) \in A^2$, this proves sufficiency.

Next, we turn to the necessity of condition (4). This can easily be verified by considering any three alternatives $(i, j, k) \in A^3$, with $i \rhd j \rhd k$. By definition of single-peaked linear orders, each order for which $i \succ j$ also has $i \succ k$. This means $O^{\rhd}_{ij} \subset O^{\rhd}_{ik}$ and $\sum_{m \in O^{\rhd}_{ij}} x_m \leq \sum_{m \in O^{\rhd}_{ik}} x_m$. A solution to (3) requires $p_{ij} = \sum_{m \in O^{\rhd}_{ij}} x_m$ and $p_{ik} = \sum_{m \in O^{\rhd}_{ik}} x_m$, and this proves $p_{ij} \leq p_{ik}$. The same argument can be used for $p_{kj} \leq p_{ki}$. This shows necessity of the condition. $\square$

Theorem 2. *For a given dataset P and ordering $\rhd$, Condition (4) can be checked in time $O(n^2)$.*

Proof. It can be easily seen that this condition may be checked in polynomial time. As written, two inequalities must be checked for each triple of alternatives, giving an obvious $O(n^3)$ time test. This can be improved upon by noting that when using a matrix of p_{ij} values, with rows and columns ranked according to the ordering $\rhd$, values above the diagonal must be non-decreasing in the rows and the columns. Conversely, as $p_{ij} + p_{ji} = 1$, values below the diagonal are non-increasing in both rows and columns. As such, each p_{ij} value must be compared with only two other values, providing an $O(n^2)$ test. $\square$

4 Recognizing single-peaked orderings

In the previous section, we have described necessary and sufficient conditions for the data to be consistent with a mixture model of single-peaked preferences with respect to an order $\rhd$. In this section, we investigate the case where this order is not given a priori. The question is now whether there exist orderings $\rhd$ for which the dataset P satisfies the mixture model. This problem is similar to the problem of recognizing Robinsonian dissimilarities. Prea and Fortin [25] describe an algorithm to recognize Robinsonian dissimilarities in $O(n^2)$ time.

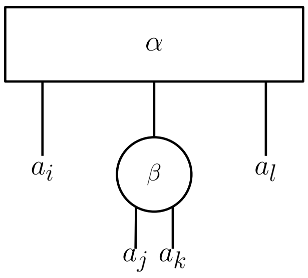
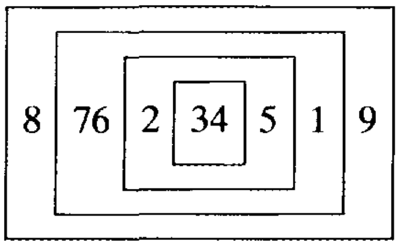


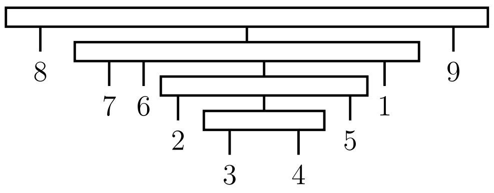
Figure 1: A PQ-tree

We define the set of orders L_P, with $\rhd \in L_P$ if and only if the dataset P satisfies condition (4) with respect to $\rhd$. In this section, we will describe an algorithm which identifies this set L_P. We will represent the set L_P by a PQ-tree. This PQ-tree $\mathcal{T}$ on the set A represents a set of permutations on A, corresponding to the orderings $\rhd \in L_P$. The leaves of $\mathcal{T}$ are the elements of A and the nodes are of two types. A P-node (represented by circles) allows any permutation of its children. On Q-nodes (represented by rectangles), the children are ordered and only reversals are allowed. For example, the PQ-tree in Figure 1 represents the set of permutations $\{(i, j, k, l), (i, k, j, l), (l, j, k, i)(l, k, j, i)\}$. For every node α of the PQ-tree, the set $S(\alpha)$ is the set of its children and this set can contain both alternatives as well as other nodes. The ordering of children of a P node is represented by $[i \rhd \beta \rhd l]$, the brackets indicating that this order can be reversed. The use of a PQ-tree allows us to efficiently represent L_P, which may have an exponential number of members.

In their paper on single-peaked orders Doignon and Falmagne [10] use a representation for the reference orders $\rhd$ for which a given preference profile is single-peaked. They represent these using nested 'boxes', where the order of alternatives within the same box can be reversed. Figure 2a is from an example in the Doignon and Falmagne paper, showing the reference orders consistent with a given preference profile using the 'boxes' representation. Figure 2b shows these same orders, represented by a PQ-tree.



(a) 'Boxes' representation



(b) PQ-Tree representation

Figure 2: Representations of reference orders $\rhd$.

We proceed as follows. First, in Section 4.1 we derive a number of properties which are satisfied by each $\rhd \in L_P$. Specifically, these properties will allow us to identify an *extreme alternative*, $\bar{a}$, which is either the first or last alternative in any ordering $\rhd \in L_P$. Next, the main part of this section is the description of an algorithm which uses this information on extreme alternatives to determine the set L_P in Section 4.2.

4.1 Preliminaries

Claim 1. *For any $\rhd \in L_P$ and each triple of alternatives $i, j, k \in A$:*

$$\text{if } i \rhd j \rhd k \text{ then } p_{ij} \leq p_{jk}. \tag{5}$$

Proof. Suppose this is not the case and $p_{jk} < p_{ij}$. Condition (4) requires $p_{ij} \leq p_{ik}$, thus $p_{jk} < p_{ik}$. Equivalently, $1 - p_{kj} < 1 - p_{ki}$ or $p_{kj} > p_{ki}$, which violates condition (4). □

It is a well known result that if preferences are single-peaked, there exists one (or more) *Condorcet winner(s)* [3]. A Condorcet winner is an alternative $c \in A$, such that for every other alternative $i \in A$, $p_{ci} \geq 0.5$. Such a Condorcet winner must obviously exist for L_P to be non-empty. We next show that if there are multiple Condorcet winners, there can not be any Condorcet non-winner in between two Condorcet winners in an ordering $\rhd \in L_P$.

Claim 2. *Consider two Condorcet winners, $c, c' \in A$ and an alternative $i \in A$, with $c \rhd i \rhd c'$. If $\rhd \in L_P$, i is also a Condorcet winner.*

Proof. Since both c and c' are Condorcet winners, $p_{cc'} = 0.5$ and both $p_{ic} \leq 0.5$ and $p_{ic'} \leq 0.5$. By condition (5), $0.5 \geq p_{ic'} \geq p_{ci} \geq 0.5$, thus $p_{ic'} = 0.5$ and by the same argument, $p_{ic} = 0.5$. Furthermore, for any alternative $j \in A$ with $j \rhd c \rhd i$ ($i \rhd c' \rhd j$), it must be the case that $p_{ij} \geq p_{ic} = 0.5$ (resp. $p_{ic'} = 0.5$). Thus, for every alternative $j \in A$, $p_{ij} \geq 0.5$ and i is a Condorcet winner. □

Claim 3. *Consider an alternative $i \in A$, such that for all Condorcet winners $c \in A$, $p_{ic} = 0.5$. Then, if $L_P \neq \varnothing$, the alternative i is also a Condorcet winner.*

Proof. Consider a Condorcet non-winner $j \in A$, and let $c \in A$ be a Condorcet winner. If there exists an ordering $\rhd \in L_P$, one of the following three cases must be true. we show that in each case $p_{ij} \geq 0.5$. Since this is true for each $j \in A$, i must be a Condorcet winner.

- If $j \rhd i \rhd c$, we have $p_{cj} \geq 0.5$, as c is a Condorcet winner. By condition (5), $p_{ij} \geq p_{cj}$, so $p_{ij} \geq 0.5$.
- If $i \rhd c \rhd j$, we have $p_{ic} = 0.5$ and by condition (4) we have $p_{ic} \leq p_{ij}$, so again $p_{ij} \geq 0.5$.
- If $i \rhd j \rhd c$, we have $p_{ic} = 0.5$ and $p_{ic} \leq p_{jc}$ (condition (4)). It must thus also be the case that $p_{jc} = 0.5$. Note that if there are alternatives j for which $i \rhd j \rhd c$, there must exist an alternative j' with $i \rhd j' \rhd c$, for which there does not exist an alternative $k \in A$ such that $j' \rhd k \rhd c$. For this alternative, only the first two situations described apply, and thus $p_{j'j} \geq 0.5$ for all $j \in A$. j' is thus also a Condorcet winner. By induction, if $p_{ic} = 0.5$ for some Condorcet winner c, then for all alternatives j with $i \rhd j \rhd c$, j is a Condorcet winner. Since we start from the premise that $p_{ic} = 0.5$ for all Condorcet winners, $p_{ij} = 0.5$.

□

For every alternative $i \in A$, we define its Minimax score, $minmax(i) = \max_{j \in A} p_{ji}$. This value is the alternative's greatest head-to-head defeat. In the following claim, we show that for any ordering $\rhd \in L_P$, $minmax(i) = p_{ji}$, with j the immediate neighbour of i to the side of the Condorcet winners.

Claim 4. *Consider the alternative $i \in A$, a Condorcet winner $c \in A$ and $j \in A$. For every $\rhd \in L_P$, with $i \rhd j \rhd c$ (or j is a Condorcet winner itself) and for which there does not exist any $k \in A$ for which $i \rhd k \rhd j$, it must be the case that:*

$$minmax(i) = p_{ji}. \tag{6}$$

Proof. Suppose this is not the case, then there exists an alternative $l \in A$, such that $p_{li} > p_{ji}$. For this alternative, it must be the case that either $i \rhd j \rhd l$ or $l \rhd i \rhd j$. The first case immediately contradicts condition (4). In the second case, condition (5) requires $p_{il} \geq p_{ji} \geq p_{cj} \geq 0.5$ (if $j = c$, $p_{ji} \geq 0.5$ is immediate). Thus, $p_{li} \leq 0.5 \leq p_{ji}$, another contradiction. □

For the next claim, we first define a second value for every alternative. We denote the number of head-to-head wins of an alternative by $w(i)$, i.e., $w(i)$ is the number of alternatives $j \in A$ for which $p_{ij} > 0.5$. We are now in a position to identify an *extreme* alternative. This is an alternative which, for every $\rhd \in L_P$ is either the first or last alternative of this order.

Claim 5. *If there exist an alternative that is not a Condorcet winner, then there exists an extreme alternative $i \in A$, such that*

1. *i is not a Condorcet winner.*
2. *$minmax(i) = \max_{j \in A} minmax(j)$.*
3. *For all other $i' \in A$ for which $minmax(i') = \max_{j \in A} minmax(j)$, $w(i) \leq w(i')$.*
4. *For every $\rhd \in L_P$, it must be the case that this alternative i is either the first or the last alternative (i.e. let c be a Condorcet winner, then there is no alternative $j \in A$ such that $c \rhd i \rhd j$).*

Before we give the proof, note that one of the requirements of an extreme alternative is that it is not a Condorcet winner. It can be easily checked that if no Condorcet non-winner exists, $p_{ij} = 0.5$ for all $i, j \in A$ and every possible ordering $\rhd \in L_P$.

Proof. First, it can be easily checked that if there exists a Condorcet non-winner, an alternative satisfying the first three conditions must exist. Indeed, for a Condorcet non-winner i, there must be at least one Condorcet winner $c \in A$ for which $p_{ci} > 0.5$ (Claim 3). Since for any Condorcet winner c and for every alternative $j \in A$, $p_{jc} \leq 0.5$, no Condorcet winner can satisfy the second condition. In the remainder of the proof we argue that if there is an alternative $i \in A$ that does not satisfy the fourth condition, it also does not satisfy either the second or third condition.

To start, if there exists an ordering $\rhd \in L_P$, there exists a Condorcet winner $c \in A$, for which $p_{ci} > 0.5$. Now suppose i is not an extreme alternative, i.e., there exists an ordering $\rhd \in L_P$ such that for some alternative $j \in A$, $c \rhd i \rhd j$. Let $i' \in A$ be the immediate neighbour of i ($c \rhd i' \rhd i$), by condition (5) and claim 4, $minmax(i) = p_{i'i} \leq p_{ij} \leq minmax(j)$. Thus, either i does not satisfy the second condition, or both i and j do. If i does satisfy the second condition, then $w(i) > w(j)$. Indeed, consider $k \in A$. If $k \rhd i \rhd j$, then $p_{ik} \geq p_{jk}$ (condition (4)). Second, if $c \rhd i \rhd k \rhd j$, because $p_{ci} > 0.5$, we have $p_{ik} > 0.5$ and $p_{jk} < 0.5$ (condition (5)). In the third situation, $c \rhd i \rhd j \rhd k$, both $p_{ik} > 0.5$ and $p_{jk} > 0.5$. Thus, whenever j is preferred to some third alternative k, i is also preferred to it, so at least $w(i) \geq w(j)$. Finally, because $p_{ij} > 0.5$, $w(i) > w(j)$. □

4.2 The Algorithm

We are now in a position to describe our algorithm for identifying the set L_P, represented by a PQ-tree $\mathcal{T}$. The main idea of this algorithm is as follows. First, we go through some initialization steps (Algorithm 2). Initially, all alternatives are assigned to a P-node $\mathcal{A}_1$, which serves as a holding node from which the alternatives will later on be assigned to other parts of the tree. Next, we identify an extreme alternative $\bar{a}$ and Condorcet winners, these Condorcet winners are assigned to a P-node $\mathcal{C}$. Starting from the extreme alternative, we will split the Condorcet non-winners into two groups (Algortihm 5). (In terms of the PQ-tree, we assign them to P-nodes $\mathcal{R}_1$ and $\mathcal{R}_2$.) The first group (node $\mathcal{R}_1$) are alternatives which must be on the same side of the Condorcet winners as the initially identified extreme alternative in every $\rhd \in L_P$, while the second group (node $\mathcal{R}_2$) is on the opposite side. We use a combination of two simple subprocedures, Splitting 1 and 2 (Algorithms 3 and 4), to make this split. Given this split, the relative ordering of the alternatives in the two groups can easily be established based on the $w(a)$ values (Algorithm 6). It is possible that not all Condorcet non-winners can be assigned to either of these two groups, and there remains a subset of alternatives as children of $\mathcal{A}_1$. We show that in this case, any ordering $\rhd' \in L_{P'}$ over these children of $\mathcal{A}_1$ can be used to complete the partial ordering we have already found over the other alternatives. The final step is the ordering of the Condorcet winners (Algorithm 7).

Algorithm 1 shows an overview of the different procedures used.

Algorithm 1 Identifying $\mathcal{T}$

```
1: Intialization of $\mathcal{T}$ with nodes $\mathcal{A}_0, \mathcal{A}_1, \mathcal{R}_1, \mathcal{R}_2, \mathcal{C}$ (Algorithm 2).
2: while $S(\mathcal{A}_1) \neq \{\mathcal{C}\}$ do
3:   Splitting Procedure (Algorithm 5).
4:   Ordering Condorcet non-winners (Algorithm 6).
5:   if $S(\mathcal{A}_1) = \{\mathcal{C}\}$ then
6:     Break
7:   else
8:     Change $\mathcal{A}_1$ into a $Q$-Node.
9:     Create $\mathcal{R}_1, \mathcal{R}_2 \in S(\mathcal{A}_1)$ ($[\mathcal{R}_1 \rhd \mathcal{C} \rhd \mathcal{R}_2]$).
10:    Identify an extreme alternative of $S(\mathcal{A}_1)$.
11:    In the following iteration of the loop, treat $\mathcal{A}_1$ as the root node.
12:  end if
13: end while
14: Ordering Condorcet winners (Algorithm 7).
```

Before we begin describing the algorithm in more detail, an important note. In the interest of simplicity, the algorithm has been constructed under the assumption that $L_P \neq \varnothing$. We will show that if this assumption holds, the output, the PQ-tree $\mathcal{T}$, represents the set L_P. If this assumption does not hold, there will still be an output $\mathcal{T}$. However, every ordering $\rhd$ consistent with this tree will obviously violate condition (4) for the dataset P. As a result, we can tell whether $\mathcal{T}$ represents L_P or if $L_P = \varnothing$ by testing condition (4) for a single ordering of the alternatives.

We begin by describing the initialization steps of the algorithm. For each alternative $i \in A$, we compute $w(i)$, the number of alternatives $j \in A$ for which $p_{ij} > 0.5$. Second, its Minimax score, $minmax(i)$. We also initialize the tree $\mathcal{T}$. Initially, we assign all alternatives to the node $\mathcal{A}_1$. This node has a child node $\mathcal{C}$, to which all Condorcet winners are then assigned. Next, we identify an extreme alternative $\bar{a}$ and assign it to the node $\mathcal{R}_1$ and the set V, which will be used later on in the algorithm. This set contains alternatives assigned to $\mathcal{R}_1$ or $\mathcal{R}_2$ which have not yet been used to sort other alternatives into these nodes. Figure 3 shows the PQ-tree after this initialization.

Algorithm 2 Initialization

```
 1: Create $\mathcal{T}$ with root $Q$-node $\mathcal{A}_0$ and $P$-nodes $\mathcal{R}_1, \mathcal{R}_2, \mathcal{A}_1, \mathcal{C}$.
 2: Set $S(\mathcal{A}_0) := \{\mathcal{R}_1, \mathcal{R}_2, \mathcal{A}_1\}$, with $[\mathcal{R}_1 \rhd \mathcal{A}_1 \rhd \mathcal{R}_2]$.
 3: Set $i \in S(\mathcal{A}_1)$ for all $i \in A$ and $\mathcal{C} \in S(\mathcal{A}_1)$.
 4: for all $i \in A$ do
 5:    Set $w(i) := |\{p_{ij} > 0.5, j \in A\}|$.
 6:    Set $minmax(i) := \max_{j \in A}(p_{ji})$.
 7:    if $minmax(i) \leq 0.5$ then
 8:       Set $S(\mathcal{C}) := S(\mathcal{C}) \cup \{i\}, S(\mathcal{A}_1) := S(\mathcal{A}_1) \backslash \{i\}$.
 9:    end if
10: end for
11: Find an extreme alternative $\bar{a}$: $minmax(\bar{a}) = \max_{i \in A} minmax(i)$ and $w(\bar{a}) \leq w(j)$ for all
    $j \in A$ for which $minmax(\bar{a}) = minmax(j)$
12: Set $S(\mathcal{R}_1) := S(\mathcal{R}_1) \cup \{\bar{a}\}, S(\mathcal{A}_1) := S(\mathcal{A}_1) \backslash \{\bar{a}\}$.
13: Set $V := V \cup \{\bar{a}\}$.
```

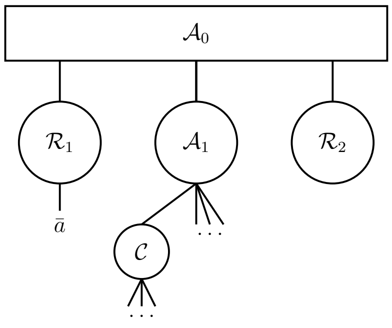


Figure 3: PQ-tree after Initialization

Starting from this single extreme alternative $\bar{a}$ in $S(\mathcal{R}_1)$, we will partition the other Condorcet non-winners over the sets $S(\mathcal{R}_1)$ and $S(\mathcal{R}_2)$, using the procedures Splitting 1 and Splitting 2 (Algorithms 3 and 4). The main idea for the first procedure is simple. Consider, without loss of generality, $i \in S(\mathcal{R}_1)$ and $j \in S(\mathcal{A}_1)$. Next, suppose $p_{ij} < \max_{c \in \mathcal{C}}(p_{ic})$, then $i \rhd c \rhd j$ violates condition (4). Thus, for every $\rhd \in L_P$, the alternative j must be on the same side of the Condorcet winners as i. In this case, we set $j \in S(\mathcal{R}_1)$. Likewise, suppose $p_{ij} > \min_{c \in S(\mathcal{C})}(p_{ic})$. Then the ordering $i \rhd j \rhd c$ violates condition (4) and the options $j \rhd i \rhd c$ and $i \rhd c \rhd j$ remain. In claim 6, we show that if there is an ordering $\rhd \in L_P$ with $j \rhd i \rhd c$, j would be assigned to an $\mathcal{R}$ node before or in the same iteration as i. It follows that if $j \in S(\mathcal{A}_1)$, $i \in S(\mathcal{R}_1)$, and $p_{ij} > \min_{c \in S(\mathcal{C})}(p_{ic})$, the alternative j must be on the other side of the Condorcet winners and we set $j \in S(\mathcal{R}_2)$.

Claim 6. *If there exists an ordering $\rhd \in L_P$, two Condorcet non-winners j, k and one Condorcet winner c, such that $j \rhd k \rhd c$, Splitting 1 will assign j to $S(\mathcal{R}_t)$ $(t = \{1, 2\})$ before, or during the same iteration as, it assigns k to $S(\mathcal{R}_t)$.*

Proof. Suppose an alternative $i \in V$ is chosen in step 2 of Splitting 1, for which there does not exist an ordering $\rhd' \in L_P$ with $j \rhd' i \rhd' c$. First, notice that the existence of $\rhd \in L_P$, with $j \rhd k \rhd c$ and the non-existence of $\rhd' \in L_P$ with $j \rhd' i \rhd' c$ implies either $i \rhd j \rhd k \rhd c$ or $j \rhd k \rhd c \rhd i$. Now, suppose i is (without loss of generality) a member of $S(\mathcal{R}_1)$. The alternative k is then assigned to $S(\mathcal{R}_2)$ if $p_{ik} > \min_{c \in S(\mathcal{C})}(p_{ic})$ or to $S(\mathcal{R}_1)$ if $p_{ik} < \max_{c \in S(\mathcal{C})}(p_{ic})$. In case $p_{ik} > \min_{c \in S(\mathcal{C})}(p_{ic})$, the ordering $i \rhd j \rhd k \rhd c$ violates condition (4), so it must be the case that $j \rhd k \rhd c \rhd i$. Then $p_{ij} \geq p_{ik} > \min_{c \in S(\mathcal{C})}(p_{ic})$ and both j and k are assigned to $S(\mathcal{R}_2)$. If $p_{ik} < \max_{c \in S(\mathcal{C})}(p_{ic})$, the same reasoning holds. An ordering $j \rhd k \rhd c \rhd i$ violates condition (4), so it must be the case that $i \rhd j \rhd k \rhd c$. Condition (4) then requires that $p_{ij} \leq p_{ik} < \max_{c \in S(\mathcal{C})}$ and both alternatives are assigned to $S(\mathcal{R}_1)$.

This line of reasoning hinges on the choice of an alternative $i \in V$ in step 2, for which there does not exist an ordering $\rhd' \in L_P$ with $j \rhd' i \rhd' c$ (for all $j \in S(\mathcal{A}_1)$). In the first iteration, this is guaranteed because i will be an extreme alternative. In later iterations, this property follows from induction. If there was an ordering $\rhd \in L_P$ with $j \rhd i \rhd c$, both j and i would have been assigned to either $S(\mathcal{R}_1)$ or $S(\mathcal{R}_2)$ and $j \notin S(\mathcal{A}_1)$. □

Algorithm 3 Splitting 1

```
 1: while $V \neq \varnothing$ do
 2:   Choose an $i \in V$.
 3:   Set $r$ such that $i \in S(\mathcal{R}_r)$.
 4:   for all $j \in S(\mathcal{A}_1)$ do
 5:     if $p_{ij} > \min_{c \in S(\mathcal{C})}(p_{ic})$ then
 6:       Set $S(\mathcal{R}_{3-r}) := S(\mathcal{R}_{3-r}) \cup \{j\}$.
 7:       Set $S(\mathcal{A}_1) := S(\mathcal{A}_1) \backslash \{j\}$.
 8:     else if $p_{ij} < \max_{c \in S(\mathcal{C})}(p_{ic})$ then
 9:       Set $S(\mathcal{R}_r) := S(\mathcal{R}_r) \cup \{j\}$.
10:       Set $S(\mathcal{A}_1) := S(\mathcal{A}_1) \backslash \{j\}$.
11:     end if
12:   end for
13:   Set $V := V \backslash \{i\}$.
14: end while
```

It is possible that this procedure ends before all Condorcet non-winners have been assigned to sets $S(\mathcal{R}_1)$ or $S(\mathcal{R}_2)$. In the next claim, we show this can only happen in a specific situation.

Claim 7. *If Splitting 1 ends before all Condorcet non-winners have been assigned to sets $S(\mathcal{R}_1)$ or $S(\mathcal{R}_2)$, then for every alternative $i \in S(\mathcal{R}_1) \cup S(\mathcal{R}_2)$ and every pair of alternatives $j, j' \in S(\mathcal{A}_1) \cup S(\mathcal{C})$, it must be the case that $p_{ij} = p_{ij'}$.*

Proof. We argue as follows, suppose that at the end of Splitting 1, there exists an alternative $i \in S(\mathcal{R}_1) \cup S(\mathcal{R}_2)$ for which there is a pair $j, k \in S(\mathcal{C})$ such that $p_{ij} \neq p_{ik}$. In this case, Splitting 1 can assign at least one more alternative in $S(\mathcal{A}_1)$ to either $S(\mathcal{R}_1)$ or $S(\mathcal{R}_2)$. First, assume there exist two alternatives $j, k \in S(\mathcal{C})$, such that $p_{ij} > p_{ik}$. Then for any Condorcet non-winner $l \in S(\mathcal{A}_1)$, either $p_{il} < p_{ij}$ or $p_{il} > p_{ik}$. In the first case, $p_{il} < \max_{c \in S(\mathcal{C})}(p_{ic})$ and in the second $p_{il} > \min_{c \in S(\mathcal{C})}(p_{ic})$. In either case, l would get sorted into either $S(\mathcal{R}_1)$ or $S(\mathcal{R}_2)$. Next, suppose that $p_{ij} = p_{ik}$ for all $j, k \in S(\mathcal{C})$, then there is some $l \in S(\mathcal{A}_1)$ for which $p_{il} \neq p_{ic}$ for all $c \in S(\mathcal{C})$. It follows that either $p_{il} < \max_{c \in S(\mathcal{C})}(p_{ic})$ or $p_{il} > \min_{c \in S(\mathcal{C})}(p_{ic})$ and l would get sorted. □

In this case we turn to a second splitting procedure. Consider two alternatives $i, j \in S(\mathcal{A}_1) \cup S(\mathcal{C})$, with $minmax(i) = p_{ji}$. Through Claim 4, we know that for any $\rhd \in L_P$, $i \rhd j \rhd c$, with c a Condorcet winner (or j is itself a Condorcet winner). Now consider an alternative $k \in S(\mathcal{R}_r)$, with $p_{jk} < p_{ji}$, then $k \rhd i \rhd j \rhd c$ violates Condition (4) and we must have $k \rhd c \rhd j \rhd i$. In other words, we know i must be on the other side of the Condorcet winners than k.

Searching through all possible triples $i', j' \in S(\mathcal{A}_1) \cup S(\mathcal{C})$ and $k' \in S(\mathcal{R}_r)$ is not very efficient. In Splitting 2, we therefore make use of some previous claims to quickly identify potential violations. Specifically, we use these to find alternatives $i, j \in S(\mathcal{A}_1) \cup S(\mathcal{C})$ and $k \in S(\mathcal{R}_r)$ such that $p_{jk} = \min_{j' \in S(\mathcal{A}_1) \cup S(\mathcal{C}), k' \in S(\mathcal{R}_r)} p_{j'k'}$ and $p_{ji} = \max_{i', j' \in S(\mathcal{A}_1) \cup S(\mathcal{C})} p_{j'i'}$. Clearly, if this triple does not have $p_{jk} < p_{ji}$, there doesn't exist any such triple. First, we identify an extreme alternative $\bar{a}$ of the set $S(\mathcal{A}_1)$ using Claim 5. It is a property of extreme alternatives that $minmax(\bar{a}) = \max_{i' \in S(\mathcal{A}_1)} minmax(i')$. Furthermore, we have the following claim.

Claim 8. *Consider the set $S(\mathcal{A}_1)$ of Condorcet non-winners not assigned in Splitting 1. If there exists an ordering $\rhd \in L_P$ then $\max_{i,j \in S(\mathcal{A}_1) \cup S(\mathcal{C})} p_{ji} = \max_{i \in S(\mathcal{A}_1)} minmax(i)$.*

Proof. Suppose this is not the case, then there must be some alternative $k \in S(\mathcal{R}_1) \cup S(\mathcal{R}_2)$, such that $p_{ki} = minmax(i)$. As i is a Condorcet non-winner, $minmax(i) > 0.5$. In claim 7, we have shown that for every $k \in S(\mathcal{R}_1) \cup S(\mathcal{R}_2)$, $p_{kj} = p_{kj'}$ for every distinct pair of alternatives $j, j' \in S(\mathcal{A}_1) \cup S(\mathcal{C})$. This implies that for all Condorcet winners $c \in S(\mathcal{C})$, $p_{ki} = p_{kc} > 0.5$, which is not possible. Thus, for every $i \in S(\mathcal{A}_1)$, $\max_{j \in S(\mathcal{A}_1) \cup S(\mathcal{C})} p_{ji} = minmax(i)$ and $\max_{i,j \in S(\mathcal{A}_1) \cup S(\mathcal{C})} p_{ji} = \max_{i \in S(\mathcal{A}_1)} minmax(i)$. □

Second, we compute $\min_{k' \in S(\mathcal{R}_r)} p_{\bar{a}k'}$ for both $r = 1, 2$ and we denote this value by $p(\mathcal{R}_r)$. Through Claim 7, we know that $p_{\bar{a}k'} = p_{j'k'}$ for all $j' \in S(\mathcal{A}_1) \cup S(\mathcal{C})$, thus $p(R_r) = \min_{k' \in S(\mathcal{R}_r), j' \in S(\mathcal{A}_1) \cup S(\mathcal{C})} p_{j'k'}$. We can now check whether $p(\mathcal{R}_r) < minmax(\bar{a})$. If this is the case, adding the alternative $\bar{a}$ to $S(\mathcal{R}_r)$ leads to violations and we add $\bar{a}$ to the opposite set.

Algorithm 4 Splitting 2

```
1:  Find the extreme alternative $\bar{a}$ of $S(\mathcal{A}_1)$.
2:  Set $p(\mathcal{R}_1) := \min_{i \in S(\mathcal{R}_1)}(p_{\bar{a}i})$.
3:  Set $p(\mathcal{R}_2) := \min_{i \in S(\mathcal{R}_2)}(p_{\bar{a}i})$.
4:  if $minmax(\bar{a}) > p(\mathcal{R}_1)$ then
5:      Set $S(\mathcal{R}_2) := S(\mathcal{R}_2) \cup \{\bar{a}\}$.
6:      Set $S(\mathcal{A}_1) := S(\mathcal{A}_1) \backslash \{\bar{a}\}$.
7:      Set $V := V \cup \{\bar{a}\}$.
8:  else if $minmax(\bar{a}) > p(\mathcal{R}_2)$ then
9:      Set $S(\mathcal{R}_1) := S(\mathcal{R}_1) \cup \{\bar{a}\}$.
10:     Set $S(\mathcal{A}_1) := S(\mathcal{A}_1) \backslash \{\bar{a}\}$.
11:     Set $V := V \cup \{\bar{a}\}$.
12: end if
```

Note that by assigning the extreme alternative of $S(\mathcal{A}_1)$ in Splitting 2, we ensure Claim 6 remains valid. Indeed, this ensures that at no point is there a triple of two Condorcet non-winners i, j and a Condorcet winner c, with $i \rhd j \rhd c$ for some $\rhd \in L_P$ where j is already assigned to either $S(\mathcal{R}_1)$ or $S(\mathcal{R}_2)$ while i is not.

If Splitting 2 can assign an alternative to either $\mathcal{R}_1$ or $\mathcal{R}_2$ (in which case V is no longer empty), we can return to Splitting 1. If at any point, Splitting 1 again terminates without having sorted all Condorcet non-winners, Splitting 2 is again used, and so on. Algorithm 5 summarizes the complete Splitting Procedure.

Algorithm 5 Splitting Procedure

```
1: Initialization
2: while $V \neq \varnothing$ do
3:     Splitting 1
4:     if $S(\mathcal{A}_1) \neq \{\mathcal{C}\}$ then
5:         Splitting 2
6:     end if
7: end while
```

After the splitting procedure finishes, the alternatives assigned to $\mathcal{R}_1$ and $\mathcal{R}_2$ will be ordered. By construction of these sets, any two alternatives $i, j \in S(\mathcal{R}_r)$ must be placed to the same side of any alternative $c \in S(\mathcal{C})$. Their relative ordering, i.e., which alternatives must be placed closest to $\mathcal{C}$, can be determined by the values of $w(i)$ as shown in the following claim.

Claim 9. *If $L_P \neq \varnothing$, then for any two Condorcet non-winners $i, j \in S(\mathcal{R}_r)$, $w(i) \neq w(j)$ and a Condorcet winner $c \in S(\mathcal{C})$, $w(i) < w(j)$ if and only if for any $\rhd \in L_P$ it is the case that $i \rhd j \rhd c$ or $c \lhd j \lhd i$.*

Proof. For all k with $i \rhd j \rhd k$, we have $p_{jk} > p_{ik}$, so j has at least as many wins against alternatives the its right as i has. For all $k \neq i$ with $k \rhd j \rhd c$, we have $p_{jk} > 0.5$ (since $p_{ck} > 0.5$), so j also has at least as many wins against alternatives to its left as i has. Finally, because $p_{ci} > 0.5$ we have $p_{ji} > 0.5$, so j wins head-to-head against i, which gives at j at least one more win than i.

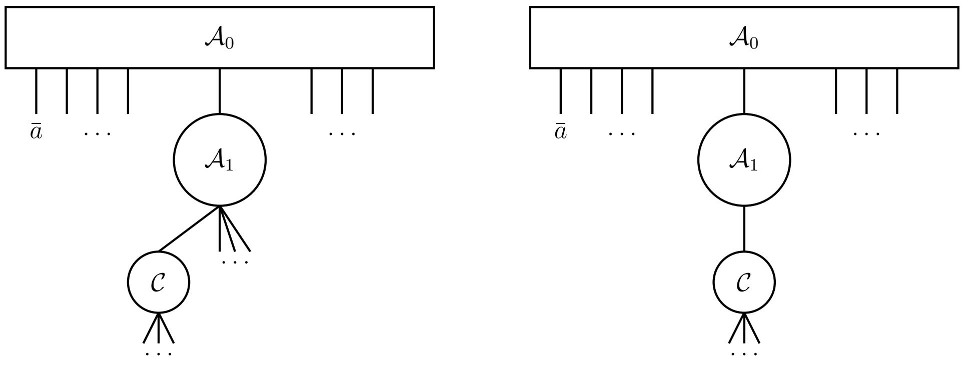
Suppose $w(i) \geq w(j)$, then there should be either at least one alternative k such that both $p_{ik} > 0.5$ and $p_{jk} \leq 0.5$ or it should be the case that $p_{ij} = 0.5$. However, by the same arguments as above neither situation can occur. $\square$

At this point, we are either in the situation depicted in Figure 4a or the one given in Figure 4b.

Let us first consider the case with $S(\mathcal{A}_1) \neq \{\mathcal{C}\}$, depicted in Figure 4a. We have shown in Claim 7 that if Splitting 1 terminates with $S(\mathcal{A}_1) \neq \{\mathcal{C}\}$, then for any given $i \in S(\mathcal{A}_0)$ and every pair $j, j' \in S(\mathcal{A}_1) \cup S(\mathcal{C})$, the values $p_{ij} = p_{ij'}$. Thus both $[i \rhd j \rhd j']$ and $[i \rhd j' \rhd j]$ are allowed. Furthermore, if the splitting procedure ends with $S(\mathcal{A}_1) \neq \{\mathcal{C}\}$, then Splitting 2 has run without finding a pair $j, j' \in S(\mathcal{A}_1) \cup S(\mathcal{C})$ and alternative $i \in S(\mathcal{R}_1) \cup S(\mathcal{R}_2)$, such that $p_{jj'} > p_{ji}$. This again allows both $[i \rhd j \rhd j']$ and $[i \rhd j' \rhd j]$. Combining these two observations, it is clear that no matter how we order the children of $S(\mathcal{A}_1)$, no new violations of condition (4) involving alternatives both in $S(\mathcal{A}_0)$ and $S(\mathcal{A}_1) \cup S(\mathcal{C})$ can be created. $\mathcal{A}_1$ can thus be handled independently of the other alternatives. This is done by changing $\mathcal{A}_1$ into a Q-node, creating $\mathcal{R}_1, \mathcal{R}_2$ and $\mathcal{A}_2$ as children of $\mathcal{A}_1$ and running the complete algorithm as before. Note that this

Algorithm 6 Ordering Condorcet non-winners

```
1: Create sets $R_1 = S(\mathcal{R}_1)$ and $R_2 = S(\mathcal{R}_2)$.
2: for all $i \in R_1 \cup R_2$ do
3:     Set $S(\mathcal{A}_0) = S(\mathcal{A}_0) \cup \{i\}$.
4: end for
5: Order the children of $\mathcal{A}_0$ such that:
6: for $i, j \in R_1$ with $w(i) < w(j)$ do
7:     $[i \rhd j \rhd \mathcal{A}_1]$
8: end for
9: for $k \in R_1$, $i, j \in R_2$, with $w(i) < w(j)$ do
10:    $[k \rhd \mathcal{A}_1 \rhd j \rhd i]$
11: end for
12: Delete $\mathcal{R}_1$ and $\mathcal{R}_2$.
```



(a) $S(\mathcal{A}_1) \neq \{\mathcal{C}\}$ (b) $S(\mathcal{A}_1) = \{\mathcal{C}\}$

Figure 4: Possible states after the Condorcet non-winners have been ordered.

situation can reoccur at most $O(n)$ times. Each time the algorithm is run, at least one alternative (the extreme alternative), becomes a direct child of the Q-node $\mathcal{A}_0, \mathcal{A}_1, \ldots$. Eventually, the second situation will occur.

Next, we consider the case depicted in 4b. In this case, the node $\mathcal{A}_1$ is deleted and $\mathcal{C}$ is made a direct child of $\mathcal{A}_0$. Since all children of $\mathcal{C}$ are Condorcet-Winners, $p_{cc'} = 0.5$ for all $c, c' \in S(\mathcal{C})$. It is therefore clear that no violation of condition (4) can exist involving only alternatives in $S(\mathcal{C})$. The relative ordering of alternatives in $S(\mathcal{C})$ is thus determined by alternatives outside of C. Algorithm 7 provides the full pseudo-code for finding this ordering. The main idea is as follows; we look for alternatives $i \in S(\mathcal{A}_0)$, for which there exist two alternatives $c, c' \in S(\mathcal{C})$ such that $p_{ic'} > p_{ic}$. The node $\mathcal{C}$ is then replaced by three P-nodes, $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$. Alternatives $j \in S(\mathcal{C})$ are divided over these three nodes, based on whether $p_{ic} \geq p_{ij}$ ($\mathcal{C}_1$), $p_{ic'} > p_{ij} > p_{ic}$ ($\mathcal{C}_2$) or $p_{ij} \geq p_{ic'}$ ($\mathcal{C}_3$). These nodes are ordered such that $[i \rhd \mathcal{C}_1 \rhd \mathcal{C}_2 \rhd \mathcal{C}_3]$. It is clear that any other ordering of these nodes would allow violations of Condition (4). If these new nodes have more than one child, they can be split in the same manner. This process of splitting P-nodes continues until either all of these nodes contain only one child, or until no more nodes can be split. We again use a set V in this algorithm, which contains all alternatives in $S(\mathcal{A}_0)$ which can potentially be used to split one of the P-nodes. This set will be important later in the time analysis of the algorithm.

Algorithm 7 Ordering Condorcet winners

```
 1: Set $V := S(\mathcal{A}_0)\backslash\{\mathcal{C}\}$.
 2: while $V \neq \varnothing$ and there exist a $P$-node $\mathcal{C}_j$ with $|S(\mathcal{C}_j)| > 1$ do
 3:    Choose $i \in V$.
 4:    if There does not exist a $P$-node $\mathcal{C}_j$, with $c, c' \in S(\mathcal{C}_j)$ and $p_{ic'} > p_{ic}$ then
 5:       Set $V := V\backslash\{i\}$.
 6:    else
 7:       for all $P$-nodes $\mathcal{C}_j$, with $c, c' \in S(\mathcal{C}_j)$ and $p_{ic'} > p_{ic}$ do
 8:          Create $P$-nodes $\mathcal{C}_{j,1}, \mathcal{C}_{j,2}, \mathcal{C}_{j,3} \in S(\mathcal{A}_0)$ with $[i \rhd \mathcal{C}_{j,1} \rhd \mathcal{C}_{j,2} \rhd \mathcal{C}_{j,3}]$
 9:          for all $k \in S(\mathcal{C}_j)$ do
10:             if $p_{ik} \geq p_{ic'}$ then
11:                Set $S(\mathcal{C}_{j,3}) := S(\mathcal{C}_{j,3}) \cup \{k\}$.
12:             else if $p_{ic'} > p_{ik} > p_{ic}$ then
13:                Set $S(\mathcal{C}_{j,2}) := S(\mathcal{C}_{j,2}) \cup \{k\}$.
14:             else
15:                Set $S(\mathcal{C}_{j,1}) := S(\mathcal{C}_{j,1}) \cup \{k\}$.
16:             end if
17:          end for
18:          Delete node $C_j$.
19:       end for
20:    end if
21: end while
22: for all $P$-nodes $\mathcal{C}_j$ with $|S(\mathcal{C}_j)| = 0$ do
23:    Delete node $\mathcal{C}_j$.
24: end for
25: for all $P$-nodes $\mathcal{C}_j$ with $|S(\mathcal{C}_j)| = 1$ do
26:    Delete node $\mathcal{C}_j$, replace it by its child in the ordering of $\mathcal{A}$.
27: end for
```

As mentioned earlier, we must still test the output of Algorithm 1. We take one ordering $\rhd$ consistent with the PQ-tree $\mathcal{T}$ and test whether condition (4) holds. If this ordering satisfies the condition, obviously $L_P \neq \varnothing$. We will now show that if $L_P \neq \varnothing$, the PQ-tree $\mathcal{T}$ represents L_P. This also implies that if $\rhd$ does not satisfy the condition (4), L_P is an empty set. Indeed, if $L_P \neq \varnothing$, every ordering consistent with $\mathcal{T}$ satisfies the condition (4).

Theorem 3. *If the set $L_P \neq \varnothing$, then Algorithm 1 constructs a PQ-tree $\mathcal{T}$, representing L_P.*

Proof. To prove this theorem, we must show two things to be true:

(i) All restrictions on the ordering consistent with the tree $\mathcal{T}$ are necessary to avoid violations of Condition (4).

(ii) No ordering consistent with the tree $\mathcal{T}$ violates Condition (4).

Throughout the description of the algorithm, we argue (i) is true. Whenever a subroutine of the algorithm fixes the relative ordering of alternatives, we argue that allowing any other relative ordering of these alternatives leads to violations of Condition (4). For brevity, we will only summarize these arguments in this proof. To show (ii) is true, we must show that whenever there is more than one possible relative ordering for a triple of alternatives consistent with $\mathcal{T}$, all of these relative orders satisfy Condition (4).

Let us begin by summarizing the arguments for (i). The subroutines Splitting 1, Splitting 2, Ordering of Condorcet non-winners and Ordering of Condorcet Winners, are used to build the tree $\mathcal{T}$. If these routines only fix the relative ordering of alternatives to eliminate possible violations of Condition (4), then (i) is satisfied. We first look at Splitting 1. This routine begins with an extreme alternative, which must be either the first or the last alternative in any ordering $\rhd \in L_P$ (Claim 5). We then identify potential violations of Condition (4) involving the extreme alternative, a Condorcet winner and a Condorcet non-winner. If such a potential violation is found, the Condorcet non-Winner is fixed to a side of the Condorcet winners, so that the relative ordering leading to the potential violation is no longer consistent with $\mathcal{T}$. Once the Condorcet non-winner is fixed to a side, Splitting 1 can use it to identify other potential violations. The main text shows how these potential violations are found. Claim 6 further supports the correctness. Splitting 2 works in a similar way, again we look for potential violations of Condition (4). In this routine, we identify potential violations involving one alternative already assigned to $S(\mathcal{R}_1)$ (or $S(\mathcal{R}_2)$) and 2 alternatives in $S(\mathcal{A}_1)$ or $S(\mathcal{C})$. Again, we show how to identify these potential violations in the main text and argue how re-assigning one alternative to $S(\mathcal{R}_1)$ or $S(\mathcal{R}_2)$ is necessary to avoid it. The ordering of Condorcet non-Winners is based on their $w(i)$ values. Claim 9 proves the correctness of the procedure. Finally, the Ordering of Condorcet winners is again based on identify potential violations, this time involving two Condorcet winners and one Condorcet non-winner. Again, the main text shows how these potential violations can be identified and how the tree can be ordered to avoid these.

We have now summarized the arguments that show any relative orders imposed by the tree $\mathcal{T}$ are necessary to avoid violations of Condition (4). It remains to be shown that there is no need for any additional restrictions, i.e., all relative orders consistent with $\mathcal{T}$ satisfy Condition (4). The figure 5 is used to illustrate the arguments made to do so. The figure shows the structure of a tree $\mathcal{T}$ at the end of the algorithm. This tree $\mathcal{T}$ consists of a root Q-node $\mathcal{A}_0$, whose direct children are Condorcet non-winners and one Q-node $\mathcal{A}_1$. A node with this group of children is formed if the situation in Figure 4a occurs. The children of $\mathcal{A}_1$ are choice alternatives, both Condorcet winners and non-winners, and P-nodes $\mathcal{C}_1$ and $\mathcal{C}_2$, this group of children is formed if the situation in Figure 4b occurs.

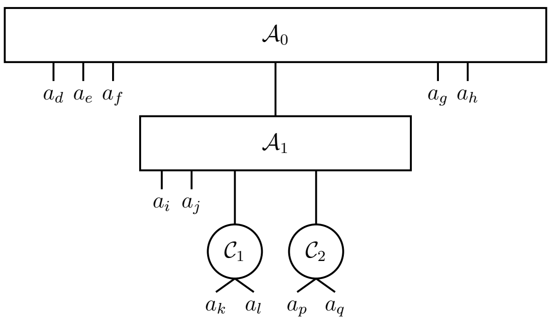


Figure 5: Structure of a tree $\mathcal{T}$ at the end of the algorihm.

There are three cases were, for a triple of alternatives, multiple relative orders are allowed. First, this can happen if there is there is one direct child of $\mathcal{A}_0$ and 2 children (direct or indirect) of $\mathcal{A}_1$. Note that $\mathcal{A}_1$ is only kept in the tree $\mathcal{T}$ if after Splitting 1 and Splitting 2, we have $S(\mathcal{A}_1) \neq \mathcal{C}$, the situation depicted in figure 4a. We have already argued that if this situation occurs, there can be no violation involving a triple with alternatives in both $S(\mathcal{A}_0)$ and $S(\mathcal{A}_1)$. A second case is if there is a Condorcet non-winner $i \in S(\mathcal{A}_1)$ and two Condorcet winners j, k, direct children of the node $\mathcal{C}_l$ node. It can be quickly checked that if $p_{ij} \neq p_{ik}$, the node $\mathcal{C}_l$ would have been split in the Ordering of Condorcet winners routine. Thus, $p_{ij} = p_{ik}$ and both relative orders consist with $\mathcal{T}$ satisfy Condition (4). Finally, there are several cases where a triple of Condorcet winners allows multiple relative orderings of the alternatives. None of the relative orderings of these triples can be violations of Condition (4), since for any two Condorcet winners c, c', it must be the case that $p_{cc'} = 0.5$.

□

Let us now turn to the analysis of the time complexity of this algorithm. We claim and prove the following result.

Theorem 4. *Algorithm 1 runs in $O(n^2)$ time.*

Proof. We begin with the initialization procedure. For a given alternative i, the values of $w(i)$ and $minmax(i)$ can be computed by checking all elements of the i^{th} row and column of the p_{ij}-matrix (O(n) elements). While reading these values, it can also be determined whether i is a Condorcet winner. Computing these values for all alternatives and determining the Condorcet winners thus takes $O(n^2)$-time. Identifying an extreme alternative can be done by checking the $minmax(i)$ and $w(i)$ values for all alternatives once ($O(n)$), and thus the complexity of the complete initialization procedure is bounded by $O(n^2)$.

Next, we turn to the splitting procedures (Splitting 1 and Splitting 2). Notice that for every alternative $i \in A$, the main loop of Splitting 1 (lines 4-11) is run at most once. In this loop, the value p_{ij} for all elements $j \in S(\mathcal{A}_1)$ is checked and a constant number of operations performed based on this value. The $O(n)$ iterations of the loop in Splitting 1, and the $O(n)$ operations in this loop provide an upper bound of $O(n^2)$ operations for the Splitting 1 procedure throughout the algorithm. Likewise, Splitting 2 is used at most $O(n)$ times. The three most costly steps are contained in lines 1-3, and in each case, at most $O(n)$ values must be checked. In particular, for line 1, there are $O(n)$ values $minmax(i)$ to be compared. In line 2 and line 3, for a given alternative $j \in S(\mathcal{A}_1)$, the p_{ji} values must be checked, of which there are again $O(n)$. Since Splitting 2 is run at most $O(n)$ time and takes at most $O(n)$ time for each iteration, the total time spent on Splitting 2 is also at most $O(n^2)$.

The time complexity of ordering the Condorcet non-winners (Algorithm 6) is straightforward. In this procedure, alternatives which are a child of the same node ($\mathcal{R}_1$ or $\mathcal{R}_2$) are ordered based on their $w(i)$ values. For m alternatives, ordering can be done in $O(m \log(m))$ time. Since each alternative is ordered in only one iteration of this procedure, the total time spent ordering Condorcet non-winner alternatives is less than $O(n \log(n))$. Finally, Algorithm 7 orders the Condorcet winners. In line 3, an alternative $i \in V$ is chosen and one of two cases will be true. Either there is a P-node $\mathcal{C}_j$, with $c, c' \in S(\mathcal{C}_j)$ and $p_{ic'} > p_{ic}$, or there is not. Determining which of the two situations is relevant takes $O(n)$ time, since this can be done by determining the minimum and maximum p_{ij} values. If no node with $c, c' \in S(\mathcal{C}_j), p_{ic'} > p_{ic}$ exists, the *if* statement resolves in constant time. If there is such a node, the *else* procedure resolves, which again takes $O(n)$ time, since all p_{ij} values for $j \in S(\mathcal{C}_j)$ must be checked against the minimum and maximum determined earlier. Given this, it is clear that after choosing an $i \in V$, it takes at most $O(n)$ time to resolve the *if-else* procedure. Next, notice that this *if-else* procedure is run at most $O(2n)$ times. Indeed, if no node with $c, c' \in S(\mathcal{C}_j), p_{ic'} > p_{ic}$ exists, i is removed from the set V. If this situation occurs $O(n))$ times, the set V is empty and the algorithm stops. If such a node exists, at least one P-node will be split into at least two parts. Since there are at most n alternatives in $S(\mathcal{C})$ at the start, it must be the case that after n splits, each $\mathcal{C}_j$ node has only 1 child. Since one iteration of the *if-else* procedure takes $O(n)$ time and there can be at most $O(n)$ iterations, ordering the Condorcet winners happens in $O(n^2)$ time.

In conclusion, we have shown that at most $O(n^2)$ time is spent in each of the procedures, as a result, the total running time of the algorithm is also bounded by $O(n^2)$. □

4.3 An Example

In this section, we will illustrate the algorithm described in this section with an example. Table 1 shows the p_{ij} values for all pairs of alternatives.

	1	2	3	4	5	6	7	8	9	10
1	-	0.5	0.5	0.5	0.6	0.8	0.5	0.8	0.5	0.7
2	0.5	-	0.6	0.5	0.6	0.8	0.6	0.8	0.5	0.7
3	0.5	0.4	-	0.5	0.6	0.8	0.6	0.8	0.5	0.7
4	0.5	0.5	0.5	-	0.6	0.8	0.6	0.8	0.5	0.7
5	0.4	0.4	0.4	0.4	-	0.7	0.4	0.8	0.4	0.4
6	0.2	0.2	0.2	0.2	0.3	-	0.2	0.9	0.2	0.2
7	0.5	0.4	0.4	0.4	0.6	0.8	-	0.8	0.4	0.7
8	0.2	0.2	0.2	0.2	0.2	0.1	0.2	-	0.2	0.2
9	0.5	0.5	0.5	0.5	0.6	0.8	0.6	0.8	-	0.7
10	0.3	0.3	0.3	0.3	0.6	0.8	0.3	0.8	0.3	-

Table 1: Matrix of p_{ij}-values.

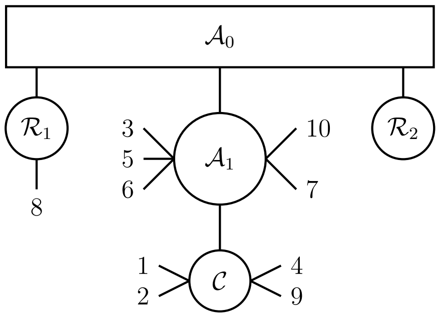
Initialization
We compute the head-to-head wins and minimax scores for each alternative. Results are given in Table 2.

	1	2	3	4	5	6	7	8	9	10
$minmax(i)$	0.5	0.5	0.6	0.5	0.6	0.8	0.6	0.9	0.5	0.7
$w(i)$	4	6	5	5	2	1	4	0	5	3

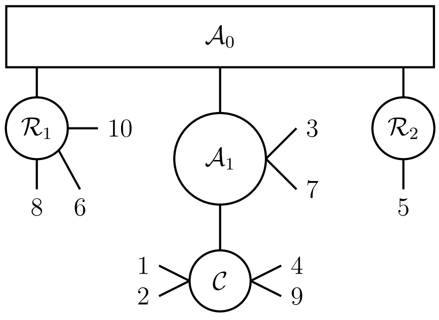
Table 2: Head-to-head wins and Minimax Score.

From these values, we see that $1, 2, 4$ and 9 are Condorcet winners, we set $S(\mathcal{C}) = \{1, 2, 4, 9\}$. The alternative 8 is an extreme alternative, as $minmax(8) = \max_{i \in A} minmax(i)$ and there are no other alternatives for which this is the case. We set $S(\mathcal{R}_1) = \{8\}$ and $V = \{8\}$. All other alternatives are direct children of $\mathcal{A}_1$. Figure 6a depicts the tree after the initialization.

Splitting Procedure - Splitting 1
Given the extreme alternative 8 and the Condorcet winners $1, 2, 4, 9$, we use Splitting 1 to place Condorcet non-winners at the different sides of the Condorcet winners.



(a) Tree after Initialization



(b) Tree after Splitting

Figure 6

- We choose $8 \in V$.
  - For 6, we have $p_{86} < \min_{c \in S(\mathcal{C})} p_{8c}$.
    - We set $S(\mathcal{A}_1) := S(\mathcal{A}_1) \backslash \{6\}, S(\mathcal{R}_1) := S(\mathcal{R}_1) \cup \{6\}$ and $V := V \cup \{6\}$.
  - For $i = 3, 5, 7, 10$, we have $p_{8i} = \min_{c \in S(\mathcal{C})} p_{8c} = \max_{c \in S(\mathcal{C})} p_{8c}$.
  - Set $V := V \backslash \{8\}$.
- We choose $6 \in V$.
  - For 5, we have $p_{65} > \max_{c \in S(\mathcal{C})} p_{6c}$.
    - We set $S(\mathcal{A}_1) := S(\mathcal{A}_1) \backslash \{5\}, S(\mathcal{R}_2) := S(\mathcal{R}_2) \cup \{5\}$ and $V := V \cup \{5\}$.
  - For $i = 3, 7, 10$, we have $p_{6i} = \min_{c \in S(\mathcal{C})} p_{6c} = \max_{c \in S(\mathcal{C})} p_{6c}$.
  - Set $V := V \backslash \{6\}$.
- We choose $5 \in V$.
  - For $i = 3, 7, 10$, we have $p_{5i} = \min_{c \in S(\mathcal{C})} p_{5c} = \max_{c \in S(\mathcal{C})} p_{5c}$.
  - Set $V := V \backslash \{5\}$.

At this point $V = \varnothing$ and we turn to the second splitting procedure.

Splitting Procedure - Splitting 2
The extreme alternative of $S(\mathcal{A}_1)$ is 10, since $minmax(10) = \max_{k \in S(\mathcal{A}_1)} minmax(k) = 0.7$, and for all other alternatives in $S(\mathcal{A}_1)$ this is not the case.

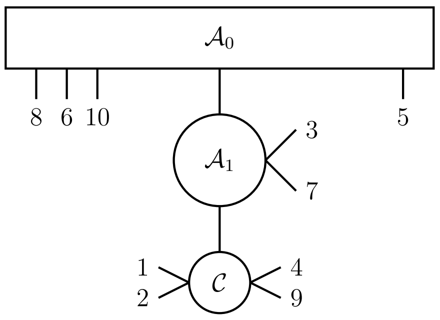
- We compute $p_{\mathcal{R}_1} = 0.8$, $p_{\mathcal{R}_2} = 0.6$.
- Because $minmax(10) = 0.7 > p_{\mathcal{R}_2}$.
  - Set $S(\mathcal{A}_1) := S(\mathcal{A}_1) \backslash \{10\}, S(\mathcal{R}_1) := S(\mathcal{R}_1) \cup \{10\}$ and $V := V \cup \{10\}$.

The algorithm now returns to Splitting 1, but is not able to assign any other alternatives to either $\mathcal{R}_1$ or $\mathcal{R}_2$. Figure 6b shows the tree at the end of the splitting procedures.

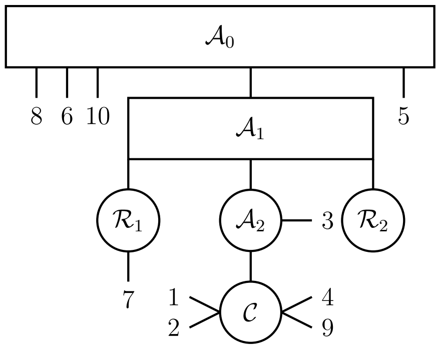
Ordering Condorcet non-winners
$S(\mathcal{R}_1)$ now contains three alternatives, $S(\mathcal{R}_2)$ only one. These nodes are removed and their children are added to $S(\mathcal{A}_0)$. We first order the alternatives that were in $S(\mathcal{R}_1)$, based on their head-to-head wins ($w(i)$ values). The only alternative that was in $S(\mathcal{R}_2)$, 5, is placed on the other side of $\mathcal{A}_1$. This leads to the ordering

$$[8 \rhd 6 \rhd 10 \rhd \mathcal{A}_1 \rhd 5].$$

Figure 7a shows the tree at the end of the Condorcet non-winner ordering procedure.



(a) Tree after ordering Condorcet non-Winners



(b) Tree after re-initialization

Figure 7

Re-Initialization
We are now in the situation depicted in Figure 4a, i.e., the set $S(\mathcal{A}_1) \neq \{C\}$. We re-run the algorithm, now treating $\mathcal{A}_1$ as the root node. We chance $\mathcal{A}_1$ into a Q-node, and we create $\mathcal{A}_2, \mathcal{R}_1$ and $\mathcal{R}_2$, setting $S(\mathcal{A}_1) := \{\mathcal{R}_1, \mathcal{R}_2, \mathcal{A}_2\}$. All alternatives that were direct children of $\mathcal{A}_1$ are made direct children of $\mathcal{A}_2$. The $minmax(i)$ and $w(i)$ values computed earlier are re-used, and the Condorcet winners also do not change. We identify a new extreme alternative for $\mathcal{A}_1$. We see that $minmax(3) = minmax(7) = \max_{i \in S(\mathcal{A}_1)} minmax(i)$, and since $w(7) < w(3)$, alternative 7 is that extreme alternative. We set $S(\mathcal{R}_1) := \{7\}$ and $V := \{7\}$. The tree at this point is depicted in Figure 7b.

Splitting Procedure - Splitting 1

- We choose $7 \in V$.
  - For 3, we have $p_{73} < \min_{c \in S(\mathcal{C})} p_{7c}$.
    - We set $S(\mathcal{A}_2) := S(\mathcal{A}_2) \backslash \{3\}, S(\mathcal{R}_1) := S(\mathcal{R}_1) \cup \{3\}$ and $V := V \cup \{3\}$.
  - Set $V := V \backslash \{7\}$.

At this point, $S(\mathcal{A}_2) = \{\mathcal{C}\}$. Running the loop of Splitting 1 for 3 has no effect. We also do not run Splitting 2.

Ordering Condorcet non-winners
$S(\mathcal{R}_1)$ now contains two alternatives, $S(\mathcal{R}_2)$ is empty. These nodes are removed and their

children are added to $S(\mathcal{A}_1)$. We order the alternatives that were in $S(\mathcal{R}_1)$, based on their head-to-head wins ($w(i)$ values). This leads to the ordering

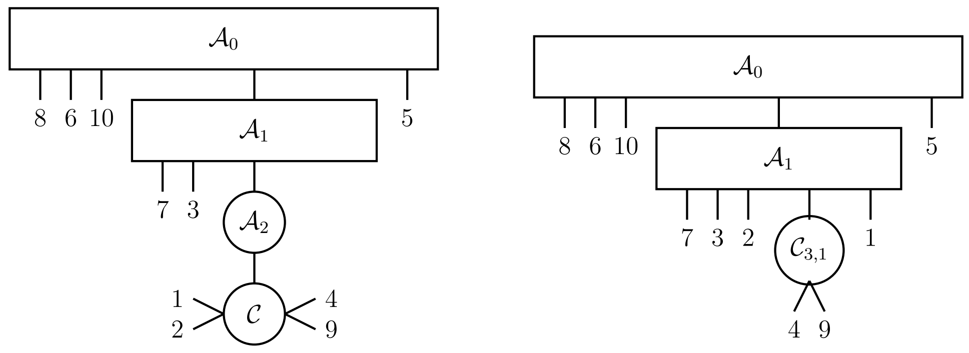
$$[7 \rhd 3 \rhd \mathcal{A}_2].$$

Figure 8a shows the tree at the end of the Condorcet non-winner ordering procedure. At this point, we are in the situation depicted in Figure 4b, we delete the node $\mathcal{A}_2$ and make $\mathcal{C}$ a direct child of $\mathcal{A}_1$, we continue by using Algorithm 7.

Ordering Condorcet winners

- Set $V := \{3, 7\}$.
- We choose $3 \in V$
  - We find for $1, 2 \in \mathcal{C}$ that $p_{3,1} = 0.5 > 0.4 = p_{3,2}$.
  - We create P-node $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3 \in S(\mathcal{A}_1)$, ordered $[3 \rhd \mathcal{C}_1 \rhd \mathcal{C}_2 \rhd \mathcal{C}_3]$.
  - Since $p_{3,1} = p_{3,4} = p_{3,9} = 0.5$, we set $S(\mathcal{C}_3) := \{1, 4, 9\}$.
  - Since $p_{3,2} = 0.4$, we set $S(\mathcal{C}_1) = \{2\}$.
  - We delete $\mathcal{C}$.
- We again choose $3 \in V$.
  - We do not find a node $\mathcal{C}_i$ with two children j, k for which $p_{3j} \neq p_{3k}$.
  - We set $V := V \backslash \{3\}$.
- We choose $7 \in V$.
  - We find for $1, 4 \in \mathcal{C}_3$ that $p_{7,1} = 0.5 > 0.4 = p_{7,4}$.
  - We create P-node $\mathcal{C}_{3,1}, \mathcal{C}_{3,2}, \mathcal{C}_{3,3} \in S(\mathcal{A}_1)$, ordered $[7 \rhd \mathcal{C}_{3,1} \rhd \mathcal{C}_{3,2} \rhd \mathcal{C}_{3,3}]$.
  - Since $p_{7,1} = 0.5$, we set $S(\mathcal{C}_{3,3}) := \{1\}$.
  - Since $p_{7,4} = p_{7,9} = 0.4$, we set $S(\mathcal{C}_{3,1}) = \{4, 9\}$.
  - We delete $\mathcal{C}_3$.
- We again choose $3 \in V$.
  - We do not find a node $\mathcal{C}_i$ with two children j, k for which $p_{7j} \neq p_{7k}$.
  - We set $V := V \backslash \{7\}$.

Figure 8b shows the final tree $\mathcal{T}$, after all nodes with no or only one child have been deleted. Table 3 is a re-ordered version of Table 1, with the rows and columns re-ordered to show one ordering of the alternatives with the tree $\mathcal{T}$. We see that it satisfies Condition 4.



(a) Tree after ordering Condorcet non-winners of $\mathcal{A}_1$

(b) Tree after ordering Condorcet winners

Figure 8

	8	6	10	7	3	2	4	9	1	5
8	-	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
6	0.9	-	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.3
10	0.8	0.8	-	0.3	0.3	0.3	0.3	0.3	0.3	0.6
7	0.8	0.8	0.7	-	0.4	0.4	0.4	0.4	0.5	0.6
3	0.8	0.8	0.7	0.6	-	0.4	0.5	0.5	0.5	0.6
2	0.8	0.8	0.7	0.6	0.6	-	0.5	0.5	0.5	0.6
4	0.8	0.8	0.7	0.6	0.5	0.5	-	0.5	0.5	0.6
9	0.8	0.8	0.7	0.6	0.5	0.5	0.5	-	0.5	0.6
1	0.8	0.8	0.7	0.5	0.5	0.5	0.5	0.5	-	0.6
5	0.8	0.7	0.4	0.4	0.4	0.4	0.4	0.4	0.4	-

Table 3: Ordered Matrix of p_{ij}-values.

5 Model Power

An important consideration for any test of choice behaviour is its power. If decisions made using other choice processes often generate datasets which satisfy the test conditions, the test has little value. Indeed, in this case there is no way of telling whether a positive test is the result of a correct hypothesis or just random chance. In this section, we briefly investigate the power of the tests we have described. Obviously, the test can not perfectly discriminate between single-peaked and non-single-peaked preferences. It is possible that the rates of choice of a non-single-peaked population still satisfy the conditions imposed by the mixture model with single-peaked preferences. Consider for example the following preferences and probabilities.

$$\begin{array}{ll} a \succ_1 c \succ_1 b & x_1 = 0.4 \\ b \succ_2 c \succ_2 a & x_2 = 0.4 \\ b \succ_3 a \succ_3 c & x_3 = 0.2 \end{array}$$

It is clear that these preferences are not single-peaked. Given the probabilities for each preference ordering, we obtain the p_{ij} values in Table 4, which satisfy Condition (4).

We show that the conditions of the model are very restrictive, which means it is unlikely

	a	b	c
a	-	0.4	0.6
b	0.6	-	0.6
c	0.4	0.4	-

Table 4: p_{ij} values satisfying Condition (4).

that a non-single-peaked population will satisfy them. We do so by comparing the volumes of the polytopes described by the conditions of the mixture model with single-peaked preferences against the volume of the linear ordering polytope. We represent these volumes as a percentage of the volume of the complete sample space, an $n^2/2$ dimensional hypercube with edge length 1, each dimension corresponds to a pair of alternatives. ([27, 28]).

Exact computation of the volume of polytopes is a difficult task. Instead, we estimate them by uniform sampling from the complete sample space and testing them against the conditions imposed by single-peaked and linear ordering polytopes. The percentage of sampled datasets satisfying the conditions is an estimate for the volume of the associated polytopes. By limiting ourselves to five alternatives, we only have to check the triangle-inequalities for the linear ordering polytope [4, 22]. For single-peaked polytopes, we implemented the algorithm described in section 4. Since this algorithm may return an ordering for which the mixture model with single-peaked preferences is not satisfied (if such orders do not exist), we then checked the conditions of the mixture model. The result is the volume of the union of the single-peaked preference polytopes for all orderings of the alternatives. We also report an estimate of the volume of the single-peaked preference polytope for a given ordering. All of these results are given in table 5.

	Linear Ordering Polytopes	Union of Single-Peaked Polytopes	Single-Peaked Polytope
n = 3	66.728%	49.999%	16.704%
n = 4	25.152%	3.3509%	0.2814%
n = 5	4.8975%	0.0208%	0.0007%

Table 5: Volumes of the Linear Ordering and Single-Peaked Polytopes for n = 3,4,5

It is clear that the mixture model with single-peaked preferences is highly restrictive. Even if we make the assumption that decisions are based on non-single-peaked preference orderings, there is a very low probability that the resulting choices are consistent with the mixture model for single-peaked preferences. For $n = 5$, this probability is already less than 0.5% if the union of all single-peaked polytopes is considered and it is clear that this probability quickly decreases as n grows larger. Note that the estimated volume for the single-peaked polytope is about $\frac{2}{n!}$ times the volume of the union of single-peaked polytopes. This must be the case as it can easily be seen that the polytopes for two different orderings $\rhd, \rhd'$ are either equal, if $\rhd$ is the reverse of $\rhd'$, or do not overlap. Indeed, if a dataset P satisfies the mixture model for both $\rhd, \rhd', \rhd \neq \lhd'$, then it must satisfy a constraint of both with equality and as such the dataset is on the border of both.

6 Conclusions

In this paper, we presented a mixture model from the choice behaviour literature and applied it to a well-known choice domain from the social choice literature. Necessary and sufficient conditions, first derived by Dridi [12], are described for the mixture model to hold for single-peaked preferences and a given ordering of the alternatives. We also showed that these conditions are easy to check in polynomial time, in contrast to the mixture model for general preferences. Furthermore, a polynomial time algorithm is provided to identify whether or not there exists some ordering of the alternatives for which the mixture model is satisfied. We also show that the conditions imposed on data by this model are very restrictive.

6.1 A Possible Extension

As single-peakedness is a very strong condition, it is generally recognized in the literature that populations are seldom truly single-peaked. Even if a logical ordering exists of alternatives, there will often be some agents whose preferences are based on very different criteria. For this reason, increasing attention is paid to notions of near-single-peakedness [16]. The mixture model as described in this paper could be extended to allow a limited number of states (weighted by probability) or percentage of the population to hold non-single-peaked preferences. For example, suppose we wish to test whether at most a fraction y of the population holds a non-single-peaked preference. Let O be the set of all strict preference orderings.

$$\sum_{m \in O_{ij}} x_m = p_{ij}, \qquad \forall (i,j) \in A^2, i \neq j. \tag{7}$$

$$\sum_{m \in O^{\triangleright}} x_m \geq 1 - y \tag{8}$$

Another possibility is to test a mixture model where all preference orders are close to single-peaked according to some measure. For example, any preference order for which a limited number of switches of alternatives give a single-peaked order. Note that in the limit, these models suggested here become mixture models for general preference orders. Indeed, if $y = 1$, or the number of allowed switches is sufficiently high this is the case. Since the general mixture model is hard to test, it seems likely that testing such nearly-single-peaked mixture models will also turn out to be a hard problem.

7 Acknowledgements

I thank the two anonymous referees for their comments on this paper, and the action editor, Jean-Paul Doignon, for pointing me to the paper by Dridi and his other comments. Furthermore, I would like to thank the members of my doctoral committee for their comments on an earlier version of this paper.

This research has been partially funded by FWO grant G.0447.10, and by the Inter-university Attraction Poles Programme initiated by the Belgian Science Policy Office.

References

[1] M. Ballester and G. Haeringer. A characterization of the single-peaked domain. *Social Choice and Welfare*, 36(2):305–322, 2011.

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

[3] D. Black. On the rationale of group decision-making. *The Journal of Political Economy*, 56(1):23, 1948.

[4] H.D. Block and J. Marschak. Random orderings and stochastic theories of responses. *Contributions to probability and statistics*, 2:97–132, 1960.

[5] K. Booth and G. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, 1976.

[6] R. Bredereck, J. Chen, and G. Woeginger. Are there any nicely structured preference profiles nearby? *Mathematical Social Sciences*, 79:61–73, 2016.

[7] J. Christophe, J.P.L. Doignon, and S.L. Fiorini. The biorder polytope. *Order*, 21(1):61–82, 2004.

[8] C. Davis-Stober. A lexicographic semiorder polytope and probabilistic representations of choice. *Journal of Mathematical Psychology*, 56(2):86–94, 2012.

[9] C. Davis-Stober, S. Park, N. Brown, and M. Regenwetter. Reported violations of rationality may be aggregation artifacts. *Proceedings of the National Academy of Sciences*, 113(33):E4761–E4763, 2016.

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

[11] T. Dridi. Sur les distributions binaires associées à des distributions ordinales. *Mathématiques et Sciences Humaines*, 69:15–31, 1980.

[12] T Dridi. Distributions binaires unimodales. *Discrete Mathematics*, 126(1-3):373–378, 1994.

[13] E. Elkind and P. Faliszewski. Recognizing 1-euclidean preferences: An alternative approach. In *SAGT*, 2014.

[14] G. Erdélyi, M. Lackner, and A. Pfandler. Computational aspects of nearly single-peaked electorates. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI 2013)*, pages 283 – 289. AAAI Press, 2013.

[15] B. Escoffier, J. Lang, and M. Öztürk. Single-peaked consistency and its complexity. In *ECAI*, volume 8, pages 366–370, 2008.

[16] P. Faliszewski, E. Hemaspaandra, and L Hemaspaandra. The complexity of manipulative attacks in nearly single-peaked electorates. *Artificial Intelligence*, 207:69–99, 2014.

[17] S. Fiorini. 0, 1/2-cuts and the linear ordering problem: Surfaces that define facets. *SIAM Journal on Discrete Mathematics*, 20(4):893–912, 2006.

[18] S. Fiorini and P.C. Fishburn. Weak order polytopes. *Discrete mathematics*, 275(1):111–127, 2004.

[19] V. Knoblauch. Recognizing one-dimensional euclidean preference profiles. *Journal of Mathematical Economics*, 46(1):1–5, 2010.

[20] C. List, R. Luskin, J. Fishkin, and I. McLean. Deliberation, single-peakedness, and the possibility of meaningful democracy: evidence from deliberative polls. *The Journal of Politics*, 75(01):80–95, 2013.

[21] D. Luce. A probabilistic theory of utility. *Econometrica: Journal of the Econometric Society*, 26:193–224, 1958.

[22] R. Martí and G. Reinelt. *The Linear Ordering Problem: Exact and Heuristic Methods in Combinatorial Optimization*, volume 175 of *Applied Mathematical Sciences*. Springer-Verlag Berlin Heidelberg, 2011.

[23] N. Megiddo. Mixtures of order matrices and generalized order matrices. *Discrete Mathematics*, 19(2):177–181, 1977.

[24] M. Pirlot and P. Vincke. *Semiorders: properties, representations, applications*, volume 36. Springer Science & Business Media, 2013.

[25] P. Préa and D. Fortin. An optimal algorithm to recognize robinsonian dissimilarities. *Journal of Classification*, 31(3):351–385, 2014.

[26] Clemens Puppe. The single-peaked domain revisited: A simple global characterization. In *COMSOC 2016*, 2016.

[27] M. Regenwetter, J. Dana, and C. Davis-Stober. Testing transitivity of preferences on two-alternative forced choice data. *Frontiers in Psychology*, 1(148):1–15, 2010.

[28] M. Regenwetter, J. Dana, and C. Davis-Stober. Transitivity of preferences. *Psychological Review*, 118(1):42, 2011.

[29] William S Robinson. A method for chronologically ordering archaeological deposits. *American Antiquity*, 16(4):293–301, 1951.

[30] O. Spanjaard and P. Weng. Single-peakedness based on the net preference matrix: Characterization and algorithms. In *6th International Workshop on Computational Social Choice (COMSOC-2016)*, 2016.

[31] R. Suck. Geometric and combinatorial properties of the polytope of binary choice probabilities. *Mathematical Social Sciences*, 23(1):81–102, 1992.

[32] X. Sui, A. Francois-Nienaber, and C. Boutilier. Multi-dimensional single-peaked consistency and its approximations. In *IJCAI*, volume 13, pages 375–382. Citeseer, 2013.

[33] M. Trick. Recognizing single-peaked preferences on a tree. *Mathematical Social Sciences*, 17(3):329–334, 1989.