ECAI 2012
Luc De Raedt et al. (Eds.)

doi:10.3233/978-1-61499-098-7-528

Choosing Combinatorial Social Choice by Heuristic Search

Minyi Li [1] and **Quoc Bao Vo** [2]

Abstract. This paper studies the problem of computing aggregation rules in combinatorial domains, where the set of possible alternatives is a Cartesian product of (finite) domain values for each of a given set of variables, and these variables are usually *not* preferentially independent. We propose a very general heuristic framework SC* for computing different aggregation rules, including rules for cardinal preference structures and Condorcet-consistent rules. SC* highly reduces the search effort and avoid many pairwise comparisons, and thus it significantly reduces the running time. Moreover, SC* guarantees to choose the set of winners in aggregation rules for cardinal preferences. With Condorcet-consistent rules, SC* chooses the outcomes that are sufficiently close to the winners.

1 Introduction

In many multi-agent decision-making scenarios, the space of alternative has a combinatorial structure: the set of possible alternatives is a Cartesian product of (finite) domain values for each of a given set of variables (aka. issues), and usually these variables are *not* preferentially independent. In classical social choice theory, candidates (aka. alternatives, outcomes) and the agents' preferences are supposed to be listed explicitly as *linear orders*, and then a *voting rule* is applied to select one or a set of winning alternatives. These traditional methods rely on a demanding assumption that the candidates should not be too numerous. However, when the domain has a combinatorial structure, the number of alternatives is exponential in the number of variables, and therefore, the agents' preferences are usually described in some compact representation languages rather than linear orders. This makes the social choice problem even more complex and challenging, because individual outcome comparisons (and thus pairwise comparisons between outcomes) in those languages might be computationally difficult. As most common aggregation methods need a number of operations at least linear (sometimes even quadratic or exponential) in the number of possible alternatives, generating the whole relation from those compact languages and directly applying rules to compute a social choice is impractical [9].

Several ways of computing rules in combinatorial domains have been considered. The most straightforward way is to use issue-by-issue (a.k.a. seat-by-seat) sequential election. However, as soon as the variables are not preferentially independent, it is very likely that deciding on the issues separately will lead to suboptimal choices [4]. Many existing work consider imposing a *domain restriction* such as separability (which makes the issue-by-issue sequential election work), see e.g., [10]; or a weaker restriction such as O-legality [16], which allows for deciding on the issues one after another. Some later works [11, 15] relax those restrictions and introduce a notion of *local Condorcet winners* (a local Condorcet winner beats every other alternative that differs in a single variable from that local Condorcet winner). The authors also propose (and implement) algorithms for computing them. However, this notion of winner differs from a *Condorcet winner* in the alternative space, as it only takes into account neighbour alternatives.

In this paper, we introduce a very general heuristic framework SC* for computing social choice in combinatorial domains. SC* enables aggregating or voting on partial assignments (an assignment of a subset of variables; and possibly comparing assignments on different subset of variables) until all variables have been assigned a value. As a result, it neither requires a counting of candidates as that in most decision-making instances, nor imposes any restriction on the agents' preference structures. The proposed heuristic approach allows searching in a much smaller sub-space of the alternatives, and thus requires significantly less pairwise outcome comparisons. It is general enough to be applicable to both aggregation rules for cardinal preferences and several Condorcet-consistent rules. Most importantly, SC* guarantees optimal social choice in aggregation rules for cardinal preference structures. With Condorcet-consistent rules, SC* chooses the candidates that are sufficiently close to the winners of the rule. Last but not least, the proposed algorithm is independent to the preference representations, and thus it is applicable to most representation languages in combinatorial domains. *Notice that we omit all the theorem proofs in this paper due to space limitation, while a longer version containing all detail proofs can be found at* `http://www.ict.swin.edu.au/personal/myli/ecai2012.pdf`

2 Preliminaries

Let $\mathbf{V} = \{X_1, \ldots, X_m\}$ be a set of variables, where each variable X_k takes values in a finite domain D_{X_k}. An alternative is uniquely identified by its values of all variables. The set of alternatives is denoted by $\mathcal{X}$, such that $\mathcal{X} = D_{X_1} \times, \ldots, \times D_{X_m}$. If $\mathbf{X} = \{X_{\sigma_1}, \ldots, X_{\sigma_\ell}\} \subseteq \mathbf{V}$, with $\sigma_1 < \cdots < \sigma_\ell$ then $D_{\mathbf{X}}$ denotes $D_{X_{\sigma_1}} \times \cdots \times D_{X_{\sigma_\ell}}$ and $\vec{x}$ denotes an assignment of variable values of $\mathbf{X}$, i.e., $\vec{x} \in D_{\mathbf{X}}$. If $\mathbf{X} = \mathbf{V}$, $\vec{x}$ is a *complete assignment* (corresponds to an outcome); otherwise $\vec{x}$ is called a *partial assignment*. If $\vec{x}$ and $\vec{y}$ are assignments to disjoint sets $\mathbf{X}$ and $\mathbf{Y}$, respectively ($\mathbf{X} \cap \mathbf{Y} = \emptyset$), we denote the combination of $\vec{x}$ and $\vec{y}$ by $\vec{x}\vec{y}$. If $\mathbf{X} \cup \mathbf{Y} = \mathbf{V}$, we call $\vec{x}\vec{y}$ a completion of assignment $\vec{x}$. We denote by $\mathtt{Comp}(\vec{x})$ the set of completions of $\vec{x}$.

A *vote* p is a linear order on $\mathcal{X}$, i.e., a transitive, antisymmetric, and total relation on $\mathcal{X}$. We denote $L(\mathcal{X})$ as the set of all possible linear orders on $\mathcal{X}$. An n-agent profile P is a collection of n votes, that is, $P = (p_1, \ldots, p_n)$, where $p_i \in L(\mathcal{X})$. Let $P(\mathcal{X})$ be the set of all possible profiles over $\mathcal{X}$, a (voting or aggregation) rule $r : P(\mathcal{X}) \to 2^{\mathcal{X}}$ maps any profile $p \in P(\mathcal{X})$ to a subset of alternatives (winners).

Since direct assessment of the preference relations in combinato-

[1] Swinburne University of Technology, Australia, email: myli@swin.edu.au
[2] Swinburne University of Technology, Australia, email: BVO@swin.edu.au

rial domains is usually infeasible, several compact languages have been proposed to avoid the exponential blow up. For instance, some languages are cardinal, e.g., utility networks [1] and soft constraints [5, 7]; some languages are purely qualitative, e.g., CP-nets [2] and CI-nets [3]. In this paper, we assume that each agent has *linear* preference over the alternative space, which may be completely or partially represented by a compact language. Therefore, a vote p might be (partially) represented by a compact language $\mathcal{L}$. That is, $\mathcal{L}$ (partially) induces an order p over the alternative space.

3 The proposed heuristic framework

3.1 The search tree

We identify optimal social choice using a search tree $\mathcal{T}$. A search tree $\mathcal{T}$ can be considered as an assignment tree. For a combinatorial social choice problem over m variables $\mathbf{V} = \{X_1, \ldots, X_m\}$, let k be the maximum size of the variable domain: $\forall X \in \mathbf{V}$, $|D_X| \leq k$, then $\mathcal{T}$ is a k-ary tree. The depth of $\mathcal{T}$ is m with the root being at depth 0. Suppose that a search tree $\mathcal{T}$ is generated following an order over variables $\sigma = X_{\sigma_1} > \cdots > X_{\sigma_m}$, then each level ℓ considers the value assigns to variable X_{σ_ℓ}. If a node Φ from the upper level is being expanded with the value of a variable X at the current level, then Φ has $|D_X|$ branches and each branch assigns a different value x to X ($x \in D_X$). The root node represents an empty assignment. A node Φ at depth ℓ represents a unique value assignment (specified by the path from the root to that node) $assg \in D_{X_{\sigma_1}} \times \cdots \times D_{X_{\sigma_\ell}}$ to the set of variables $\{X_{\sigma_1}, \ldots, X_{\sigma_\ell}\}$. Each node at depth m corresponds to an alternative, as all the variables has been assigned a value.

3.2 The SC* search algorithm

Similar to other heuristic search algorithms [13], SC* creates a search tree by iteratively selecting a node that appears to be most likely to lead towards an optimal social choice (a winner of the rule). We first give a general definition of the evaluation function of the search strategy, and then we would specify different evaluation functions for different aggregation rules in the next section.

Definition 1 (Evaluation function) *In an iteration, given a set of leaf nodes* $\mathbf{L}$ *in the search tree* $\mathcal{T}$*, an evaluation function* $\mathcal{F}$ *maps each node* Φ *in* $\mathbf{L}$ *into a number* ($\mathcal{F} : \mathbf{L} \to \Re$)*, which indicates how promising (good) the node is, i.e., how likely the node will eventually lead to a winner of the given rule. In this work,* $\mathcal{F}(\Phi)$ *is modelled as kinds of disvalue of* Φ *(e.g., cost, distance to goal, penalty, disutility). Therefore, the* smaller *the* $\mathcal{F}$ *value of a node* Φ*, the* more *promising* Φ *is.*

SC* (see Algorithm 1) is adapted from the A* heuristic search algorithm [6] with $\mathcal{F}$ being the heuristic function. We first randomly generate an order over variables $\sigma = X_{\sigma_1} > \ldots X_{\sigma_m}$, following which the search tree would be created (line 1). Starting with the root node with an empty assignment (logically expressed as $True$) (line 2), SC* maintains a priority queue of nodes to be expanded, known as the $fringe$. The lower $\mathcal{F}$ value of a node Φ, the higher its priority is, i.e., the more upfront it is in the $fringe$. *We emphasize here that for two nodes with the same* $\mathcal{F}$ *value, the one at level* $\ell < m$ *must be ordered before (given higher priority than) the one at level* $\ell = m$*.*

In each iteration of SC*, the first node Φ, i.e. the node with the lowest $\mathcal{F}$ value, is removed from the $fringe$ (line 4). If it does not represent a complete assignment ($assg$ denotes the assignment of Φ), it will be expanded (line 6-9). Let X be the next variable to be assigned according to the order σ, for each $x \in D_X$, a child node expanded from Φ with the assignment $assg \wedge x$ will be created and added into the $fringe$. After creating all possible child nodes, the $\mathcal{F}$ value of each existing leaf node is computed accordingly, and the $fringe$ is sorted in the ascending order of the $\mathcal{F}$ values (line 10-11).

Algorithm 1: SC*(P)

Input: P, a preference profile of n agents over m issues
Output: $outs$, a set of outcomes

1 Randomly generate an order over variables $\sigma = X_{\sigma_1} > \cdots > X_{\sigma_m}$;
2 $fringe \leftarrow$ INSERT(MAKE-NODE($True$), $fringe$);
3 **while** $fringe \neq \emptyset$ **do**
4 $\quad \Phi \leftarrow$ REMOVE-FIRST($fringe$); $assg \leftarrow$ ASSIGNMENT(Φ);
5 $\quad$ **if** $assg$ *is* not *a complete assignment* **then**
6 $\quad\quad X \leftarrow$ NEXT-VARIABLE($assg$, σ);
7 $\quad\quad$ **foreach** $x \in D_X$ **do**
8 $\quad\quad\quad$ INSERT(MAKE-NODE($assg \wedge x$), $fringe$);
9 $\quad\quad$ **end**
10 $\quad\quad$ Compute $\mathcal{F}$ for each leaf node;
11 $\quad\quad$ SORT-FRINGE-ASC($fringe$);
12 $\quad$ **else**
13 $\quad\quad$ AppendTo($outs$, $assg$);
14 $\quad\quad \Phi' \leftarrow$ REMOVE-FIRST($fringe$);
15 $\quad\quad$ **while** $\mathcal{F}(\Phi') = \mathcal{F}(\Phi)$ **do**
16 $\quad\quad\quad assg \leftarrow$ ASSIGNMENT(Φ');
17 $\quad\quad\quad$ AppendTo($outs$, $assg$);
18 $\quad\quad\quad \Phi' \leftarrow$ REMOVE-FIRST($fringe$);
19 $\quad\quad$ **end**
20 $\quad$ **end**
21 **end**
22 **return** $outs$;
23 Notice that function SORT-FRINGE-ASC sorts the $fringe$ according to an ascending order of the $\mathcal{F}$ values; for two nodes with the same $\mathcal{F}$ value, the one with partial assignment (at level $\ell < m$) must be ordered before the one with a complete assignment (at level $\ell = m$).

SC* continues until the current chosen node Φ for expansion is a leaf node at level m, i.e. the assignment of Φ is a complete assignment to the set of domain variables $\mathbf{V}$ (line 12). Notice that there might be more than one node with the same $\mathcal{F}$ value in an iteration. As we mentioned before, with the nodes that have the same $\mathcal{F}$ value, we give higher priority to a node at an upper level ($\ell < m$) than the one at the deepest level ($\ell = m$). Consequently, when the first node Φ in the $fringe$ is at level m, the other nodes that have the same $\mathcal{F}$ value as Φ must also be at level m. Then, SC* algorithm collects the nodes with the smallest $\mathcal{F}$ value, and put their assignments (each corresponds to a unique outcome) into a set $outs$ (line 13-19). Finally, SC* returns $outs$ as the set of socially preferred alternatives (line 22).

3.3 The basis of heuristic

All the heuristic search strategies introduced in this paper are based on the following definition of best possible alternative of a node:

Definition 2 (Best possible alternative) *At each node* Φ *of the search tree* $\mathcal{T}$*, each agent* i *has a best possible alternative (*BPA*) on* Φ*, denoted by* $\mathtt{BPA}_i(\Phi)$*, which is the optimistic outcome that agent* i *can obtain with the variable values assigned from the root to* Φ *being fixed, i.e., the best outcome that agent* i *can obtain from the subtree of* Φ*. Formally,*

$$\mathtt{BPA}_i(\Phi) = \mathtt{OPTIMIZE}(assg, \mathcal{L}_i)$$

where assg = ASSIGNMENT(Φ) *represents the assignment specified by the path from the root to* Φ*; function* OPTIMIZE *optimizes the values of the remaining variables that are not in assg, according to agent i's preference* $\mathcal{L}_i$.

Let Comp(*assg*) represents the completions of *assg*, then $\texttt{BPA}_i(\Phi)$ is the best alternative among Comp(*assg*) for agent *i*. Consequently, the BPA of the root node for an agent *i* corresponds to the optimal (best) outcome for agent *i* in the entire outcome space, i.e. all variables are assigned the preferred values according to p_i. For a node Φ at level *m* of $\mathcal{T}$, $\forall i, j \in \{1, \ldots, n\}$ and $i \neq j$, $\texttt{BPA}_i(\Phi) = \texttt{BPA}_j(\Phi) = \texttt{ASSIGNMENT}(\Phi)$, as the assignment of Φ is complete.

For instance, consider an ordering over three binary variables *A*, *B* and *C*: $abc \succ \bar{a}bc \succ ab\bar{c} \succ \bar{a}b\bar{c} \succ a\bar{b}\bar{c} \succ \bar{a}\bar{b}\bar{c} \succ a\bar{b}c \succ \bar{a}\bar{b}c$, and a node Φ with the assignment $assg = a$ (resp. $assg = \bar{a}\bar{b}$), the completions of *assg* is $\{abc, ab\bar{c}, a\bar{b}c, a\bar{b}\bar{c}\}$ (resp. $\{\bar{a}\bar{b}c, \bar{a}\bar{b}\bar{c}\}$). According to the preference order, the BPA of Φ is abc (resp. $\bar{a}\bar{b}\bar{c}$), because $abc \succ ab\bar{c} \succ a\bar{b}\bar{c} \succ a\bar{b}c$ (resp. $\bar{a}\bar{b}\bar{c} \succ \bar{a}\bar{b}c$).

4 Evaluation function of social choice rules

4.1 Aggregation rules for cardinal preferences

When preferences are cardinal, the preference profile $P = \langle f_1, \ldots, f_n \rangle$ consists of every agent's scoring function. A scoring function f_i of an agent *i* maps every alternative $\vec{x}$ ($\vec{x} \in \mathcal{X}$) into a real number $\mathfrak{R}$ ($f_i : \mathcal{X} \rightarrow \mathfrak{R}$), based on the penalty (or distance) that $\vec{x}$ caused (or from the goal) according to agent *i*'s preference. Typical examples of cardinal preference structures include penalty scoring functions, dis-utility functions and some logical preference languages like weighted goals and distance goals.

To model the preference of the group, for each alternative $\vec{x}$, the score $f_i(\vec{x})$ of each agent *i* are synthesized by an aggregation operator $\diamond$: $\mathfrak{R}^n \rightarrow \mathfrak{R}$ into a so-called social welfare function $s(\vec{x})$ reflecting the preference of the group of agents [12]. Formally, given a cardinal preference profile $P = \langle f_1, \ldots, f_n \rangle$, the social welfare scoring function *s* mapping from $\mathcal{X}$ to $\mathfrak{R}$ is defined by:

$$\forall \vec{x} \in \mathcal{X},\ s(\vec{x}) = \diamond\{f_i(\vec{x}) \mid i = 1, \ldots, n\}$$

Classically, $\diamond$ is an operator that satisfies non-decreasingness for each of its argument and commutativity. As discussed in [8], the most natural choices for $\diamond$ are sum and max. sum is a *utilitarian* aggregation operator, stating that the collective score of an outcome is the sum of the scores of the agents in the group. On the other hand, max states that the maximum score among all the agents should be considered. Thus, the max aggregation operator corresponds to the *egalitarian social welfare*. Finally, an outcome $\vec{x}$ is $\diamond$-optimal *iff* $s(\vec{x})$ is *minimized*.

For each node Φ in the search tree $\mathcal{T}$, each agent *i* has a best possible alternative $\texttt{BPA}_i(\Phi)$, and accordingly, an optimistic score $f_i(BPA_i(\Phi))$, which is the best (the smallest) possible score that agent *i* may obtain from the subtree of Φ. Applying SC* to compute an optimal social choice with cardinal preferences, the heuristic evaluation function $\mathcal{F}_{Cardinal}$ is defined as follows.

Definition 3 (Evaluation function of cardinal rules) *The evaluation function* $\mathcal{F}_{Cardinal}$*, mapping from a node* Φ *to* $\mathfrak{R}$*, is defined by:*

$$\mathcal{F}_{cardinal}(\Phi) = \diamond\{f_i(\texttt{BPA}_i(\Phi)) \mid i = 1, \ldots, n\}$$

In each iteration, SC* chooses a node that has the minimum $\mathcal{F}_{Cardinal}$ value to expand, until the node chosen for expansion corresponds to a complete assignment. Notice that with cardinal preference structures, the $\mathcal{F}$ value of a node is fixed. *Therefore, the* $\mathcal{F}$ *value of a node will only need to be calculated once.*

$abc \succ \bar{a}bc \succ ab\bar{c} \succ \bar{a}b\bar{c} \succ a\bar{b}\bar{c} \succ \bar{a}\bar{b}\bar{c} \succ a\bar{b}c \succ \bar{a}\bar{b}c$

(a) agent 1

$\bar{a}bc \succ \bar{a}b\bar{c} \succ \bar{a}\bar{b}\bar{c} \succ \bar{a}\bar{b}c \succ abc \succ ab\bar{c} \succ a\bar{b}c \succ a\bar{b}\bar{c}$

(b) agent 2

$\bar{a}b\bar{c} \succ \bar{a}\bar{b}\bar{c} \succ abc \succ ab\bar{c} \succ a\bar{b}c \succ a\bar{b}\bar{c} \succ \bar{a}bc \succ \bar{a}\bar{b}c$

(c) agent 3

Figure 1. Agents's preferences

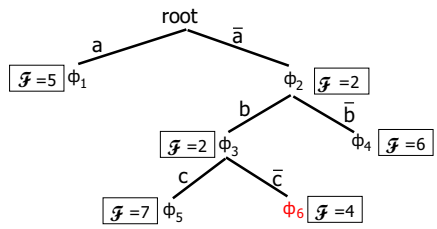

Figure 2. Search tree for cardinal preferences

Theorem 1 *Given that the social welfare function* $\diamond$ *satisfies non-decreasingness and commutativity, the evaluation function* $\mathcal{F}_{Cardinal}$ *is admissible.*

Theorem 2 *If* $\mathcal{F}_{Cardinal}$ *is admissible, SC* chooses the set of winners according to the rule* $\diamond$.

Example. Consider three agents' preferences over three binary domain variables $\mathbf{V} = \{A, B, C\}$ depicted in Figure 1. The scoring function f_i of an agent *i* is defined by the position of the alternative in the preference ordering, i.e., the optimal outcome being at 1 and the worst outcome being at $|\mathcal{X}|$. For instance, consider agent 1's preference ordering in Figure 1(a), $f_1(abc) = 1, f_1(\bar{a}bc) = 2, \ldots, f_1(\bar{a}\bar{b}c) = 8$. The table below shows the $\texttt{BPA}_i$ and f_i of each created node Φ of each agent *i*, and then the $\mathcal{F}_{Cardinal}$ values. In this example, we consider max rule ($\diamond$ =max). Therefore, the $\mathcal{F}_{Cardinal}$ value of a node Φ (in a column) is the maximum f_i among the three rows.

Agent	Φ_1		Φ_2		Φ_3		Φ_4		Φ_5		Φ_6	
	$\texttt{BPA}_i$	f_i	$\texttt{BPA}_i$	f_i	$\texttt{BPA}_i$	f_i	$\texttt{BPA}_i$	f_i	$\texttt{BPA}_i$	f_i	$\texttt{BPA}_i$	f_i
agent 1	abc	1	$\bar{a}bc$	2	$\bar{a}bc$	2	$\bar{a}\bar{b}\bar{c}$	6	$\bar{a}bc$	2	$\bar{a}\bar{b}\bar{c}$	4
agent 2	$ab\bar{c}$	5	abc	1	$\bar{a}bc$	1	$\bar{a}\bar{b}\bar{c}$	3	$\bar{a}bc$	1	$\bar{a}b\bar{c}$	2
agent 3	$ab\bar{c}$	3	$\bar{a}b\bar{c}$	1	$\bar{a}b\bar{c}$	1	$\bar{a}\bar{b}\bar{c}$	2	$\bar{a}bc$	7	$\bar{a}b\bar{c}$	1
$\mathcal{F}_{Cardinal}$	5		2		2		6		7		4	

Figure 2 shows the search tree of this example. The 1^{st} iteration creates two child nodes Φ_1 and Φ_2 of the root. $\texttt{BPA}_1(\Phi_1) = abc$, $f_1(abc) = 1$; $\texttt{BPA}_2(\Phi_1) = ab\bar{c}$, $f_2(ab\bar{c}) = 5$; $\texttt{BPA}_3(\Phi_1) = ab\bar{c}$, $f_3(ab\bar{c}) = 3$. Therefore $\mathcal{F}_{Cardinal}(\Phi_1) = \texttt{max}\{1, 5, 3\} = 5$. Similarly, $\mathcal{F}_{Cardinal}(\Phi_2) = 2$. As $\mathcal{F}_{Cardinal}(\Phi_2) < \mathcal{F}_{Cardinal}(\Phi_1)$, Φ_2 is given higher priority in the *fringe* and is expanded in the 2^{nd} iteration. This process continues until Φ_6 (specifies a complete assignment) is chosen for expansion. As Φ_6 is the only node with a minimum $\mathcal{F}_{Cardinal}$ value among the existing leaf nodes (Φ_1, Φ_4, Φ_5, Φ_6), SC* returns $\bar{a}b\bar{c}$ (ASSIGNMENT(Φ_6)) as the collective decision.

4.2 Condorcet-consistent rules

Give a preference profile *P*, a *Condorcet winner* (CW) is a candidate $\vec{x}$ such that when compared with any other candidate $\vec{y}$ ($\vec{x} \neq \vec{y}$), more than half of the agents prefer $\vec{x}$ over $\vec{y}$. Let $\#\{i \leq n : \vec{x} \succ_i \vec{y}\}$ denote the number of agents who prefer an alternative $\vec{x}$ over another alternative

$\vec{y}$, formally, an alternative $\vec{x}$ is a Condorcet winner iff $\forall \vec{y} \in \mathcal{X}$, $\#\{i \leq n : \vec{x} \succ_i \vec{y}\} > \#\{j \leq n : \vec{y} \succ_j \vec{x}\}$.

When the Condorcet winner exists, it is unique. However, it is possible for a paradox to form, in which collective preferences can be cyclic (i.e. not transitive), even if the preferences of individual agents are not. In combinatorial domains, however, the number of alternatives is often much larger than the number of agents, and therefore, there almost *never* exists a Condorcet winner in practical cases. Hence, in the next subsections, we consider two Condorcet-consistent rules, namely Copeland rule and Minimax rule. A rule is Condorcet-consistent, if it chooses the Condorcet winner when one exists. Before going to define the evaluation function of these two rules, we first introduce some related concepts as follows.

In each iteration, we denote $\mathbf{L}$ as the set of existing leaf nodes in the search tree. In order to apply SC* to compute Condorcet-consistent rules, we first define an upper approximation of preference relations over the set of leaf nodes $\mathbf{L}$, based on the agent's optimistic evaluation (BPA) of each leaf node.

Definition 4 (Preference relations between leaf nodes) *Given a pair of leaf nodes Φ and Φ', we say an agent i prefers Φ over Φ' (written as $\Phi \succ_i \Phi'$) iff agent i prefers the* BPA *of Φ over the* BPA *of Φ'. Formally, $\forall i \in \{1, \ldots, n\}$ and $\forall \Phi, \Phi' \in \mathbf{L}$, $\Phi \succ_i \Phi'$ iff $\text{BPA}_i(\Phi) \succ_i \text{BPA}_i(\Phi')$.*

Proposition 1 *Let Φ be a node at level m (represents a complete assignment) and Φ' be a node at level ℓ ($\ell \leq m$), $\vec{x}$ be the alternative corresponds to Φ ($\vec{x} =$ ASSIGNMENT(Φ)) and $assg' =$ ASSIGNMENT(Φ'), if an agent i prefers Φ to Φ', then $\forall \vec{y} \in \text{Comp}(assg')$, $\vec{x} \succ_i \vec{y}$.*

Definition 5 (Majority domination between leaf nodes) *In an iteration, a leaf node Φ majority dominates another leaf node Φ', written as $\Phi \succ_{maj} \Phi'$ if there is a majority number of agents who prefer Φ over Φ'.*

Corollary 1 *Let Φ be a node at level m and Φ' be a node at level ℓ ($\ell \leq m$); $\vec{x} =$ ASSIGNMENT(Φ) and $assg' =$ ASSIGNMENT(Φ'), if $\Phi \succ_{maj} \Phi'$, then $\forall \vec{y} \in \text{Comp}(assg')$, $\vec{x} \succ_{maj} \vec{y}$.*

4.2.1 Copeland rule

Given a profile P, the *Copeland score* of an alternative is the number of alternatives it beats in pairwise comparisons. Formally, for an alternative $\vec{x}$, let $s(\vec{x})$ denotes the Copeland score of $\vec{x}$, then

$$s(\vec{x}) = \#\{\vec{y} \in \mathcal{X} : \vec{x} \succ_{maj} \vec{y}\}$$

A *Copeland winner* $\vec{x}$ is an alternative that maximizing $s(\vec{x})$.

In each iteration, we conduct a pairwise comparison over the set of leaf nodes $\mathbf{L}$ (based on Definition 4 and Definition 5). For consistency, we define the evaluation function as sort of disvalue of a node and the smaller $\mathcal{F}_{Copeland}$ is, the more promising the node is. Therefore, in an iteration, the $\mathcal{F}_{Copeland}$ value of a node Φ is defined by the number of leaf nodes that majority dominates Φ.

Definition 6 (Copeland evaluation function) *The evaluation function $\mathcal{F}_{Copeland}$, mapping from an existing leaf node Φ ($\Phi \in \mathbf{L}$) to $[0, +\infty]$, is defined by:*

$$\mathcal{F}_{Copeland}(\Phi) = \#\{\Phi' \in \mathbf{L} : \Phi' \succ_{maj} \Phi\}$$

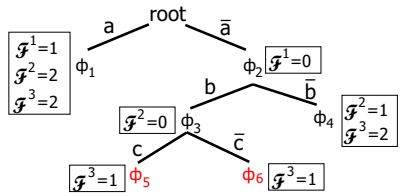

Figure 3. Search tree for Copeland rule

In the case with Copeland rule, the evaluation value $\mathcal{F}_{Copeland}$ of a node varies during the search (as some leaf nodes are expanded and new nodes are created). Therefore, in each iteration, we not only need to calculate the $\mathcal{F}_{Copeland}$ value of the new created nodes, but also need to update the $\mathcal{F}_{Copeland}$ value of other remaining leaf nodes.

Theorem 3 *The evaluation function $\mathcal{F}_{Copeland}$ is not admissible.*

Example (Cont.) We continue with the three agents' preferences in Figure 1. In the table below we give a pairwise comparison matrix (Φ in column and Φ' in row). The value in the cell of column Φ row Φ' is 1 iff $\Phi \succ_{maj} \Phi'$, otherwise is -1. We denote $\mathcal{F}^h_{Copeland}$ as the $\mathcal{F}_{Copeland}$ value of a leaf node in the h iteration. In each iteration h, we calculate the $\mathcal{F}^h_{Copeland}$ value by only looking at the rows of the *existing leaf nodes*: the $\mathcal{F}^h_{Copeland}(\Phi)$ is the sum of those rows which is equal to -1 in the column of Φ. There is a "cross" in a cell of the table iff: *i)* $\Phi = \Phi'$ (cater-corner); or *ii)* Φ or Φ' has been expanded (is no longer a leaf node) in that iteration. Figure 3 illustrates the search

$\downarrow \Phi'$ $\Phi \rightarrow$	Φ_1	Φ_2	Φ_3	Φ_4	Φ_5	Φ_6
Φ_1	×	1	1	1	-1	1
Φ_2	-1	×	×	×	×	×
$\mathcal{F}^1_{Copeland}$	1	0	-	-	-	-
Φ_3	-1	×	×	-1	×	×
Φ_4	-1	×	1	×	1	1
$\mathcal{F}^2_{Copeland}$	2	×	0	1	-	-
Φ_5	1	×	×	-1	×	-1
Φ_6	-1	×	×	-1	1	×
$\mathcal{F}^3_{Copeland}$	2	×	×	2	1	1

tree with this example (In the search tree $\mathcal{F}^h_{Copeland}$ is written shortly as $\mathcal{F}^h$). In the 1^{st} iterations, Φ_1 and Φ_2 are created as the children of the root node. As $\Phi_2 \succ_{maj} \Phi_1$, $\mathcal{F}^1_{Copeland}(\Phi_1) = 1$ and $\mathcal{F}^1_{Copeland}(\Phi_2) = 0$. The ordering in the fringe would be $\Phi_2\Phi_1$. In the 2^{nd} iteration, the first node Φ_2 in the *fringe* is popped out and expanded. Two child nodes Φ_3 and Φ_4 are created. We run a pairwise comparison between three existing leaf nodes Φ_1, Φ_3 and Φ_4. As $\Phi_3 \succ_{maj} \Phi_1$, $\Phi_4 \succ_{maj} \Phi_1$, $\Phi_3 \succ_{maj} \Phi_4$, $\mathcal{F}^2_{Copeland}(\Phi_1) = 2$, $\mathcal{F}^2_{Copeland}(\Phi_3) = 0$, $\mathcal{F}^2_{Copeland}(\Phi_4) = 1$. Therefore, the order of the nodes in the *fringe* is $\Phi_3\Phi_4\Phi_1$ and Φ_3 will be expanded in the next iteration. In the 3^{rd} iterations, two child node Φ_5 and Φ_6 of Φ_3 are created. $\mathcal{F}^3_{Copeland}(\Phi_1) = 2$, $\mathcal{F}^3_{Copeland}(\Phi_4) = 2$, $\mathcal{F}^3_{Copeland}(\Phi_5) = 1$ and $\mathcal{F}^3_{Copeland}(\Phi_6) = 1$. The current *fringe* is $\Phi_5\Phi_6\Phi_1\Phi_4$. In the 4^{th} iteration, Φ_5 is chosen for expansion. As Φ_5 is a complete assignment and Φ_6 has the same $\mathcal{F}_{Copeland}$ value as Φ_5, the corresponding alternatives of Φ_5 and Φ_6 ($\bar{a}bc$ and $\bar{a}b\bar{c}$) will be put into *outs*. SC* returns *outs* as the final chosen set of alternatives.

4.2.2 Minimax

Given a profile P and a pair of alternatives $\vec{x}$ and $\vec{y}$, let $N(\vec{x}, \vec{y})$ denote the number of agents that rank $\vec{y}$ ahead of $\vec{x}$ in the profile P: $N(\vec{x}, \vec{y}) =$

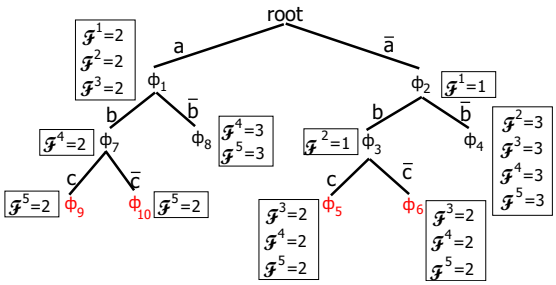

Figure 4. Search tree for Minimax rule

$\#\{i \leq n : \vec{y} \succ_i \vec{x}\}$. The *Minimax score* $s(\vec{x})$ of an alternative $\vec{x}$ is defined by:

$$s(\vec{x}) = \max_{\vec{y}\in\mathcal{X},\ \vec{y}\neq\vec{x}} N(\vec{x}, \vec{y})$$

A *Minimax winner* $\vec{x}$ is an alternative that *minimizing* $s(\vec{x})$.

Similar as the Copeland rule, in each iteration, we conduct a pairwise comparison over the set of leaf nodes $\mathbf{L}$. For a pair of leaf nodes Φ and Φ', we define $N(\Phi, \Phi')$ as the number of agents that prefer Φ' over Φ: $N(\Phi, \Phi') = \#\{i \leq n : \Phi' \succ_i \Phi\}$. Then we can define the following evaluation function of Minimax rule:

Definition 7 (Minimax evaluation function) *The evaluation function $\mathcal{F}_{Minimax}$, mapping from an existing leaf node Φ ($\Phi \in \mathbf{L}$) to $[0, +\infty]$, is defined by:*

$$\mathcal{F}_{Minimax}(\Phi) = \max_{\Phi'\in\mathbf{L},\ \Phi'\neq\Phi} N(\Phi, \Phi')$$

Theorem 4 *The evaluation function $\mathcal{F}_{Minimax}$ is not admissible.*

Example (Cont.) We apply SC* to compute the Minimax rule with our running example (Figure 1). In the table below we give the number of defeats in pairwise comparison (Φ in column and Φ' in row) The value in the cell of column Φ row Φ' is the number of defeats of Φ vs. Φ', i.e., the number of agents that prefer Φ' over Φ ($N(\Phi, \Phi')$). The $\mathcal{F}^h_{Minimax}$ value of a node Φ in iteration h is then the maximum among the rows of the current existing nodes in the column Φ. Similarly, there is a "cross" in a cell of the table iff: $\Phi = \Phi'$; or Φ or Φ' has been expanded.

$\downarrow \Phi'\ \Phi \rightarrow$	Φ_1	Φ_2	Φ_3	Φ_4	Φ_5	Φ_6	Φ_7	Φ_8	Φ_9	Φ_{10}
Φ_1	×	1	1	1	2	1	×	×	×	×
Φ_2	2	×	×	×	×	×	×	×	×	×
$\mathcal{F}^1_{Minimax}$	2	1	-	-	-	-	-	-	-	-
Φ_3	2	×	×	3	×	×	×	×	×	×
Φ_4	2	×	0	×	1	0	2	2	2	2
$\mathcal{F}^2_{Minimax}$	2	×	1	3	-	-	-	-	-	-
Φ_5	1	×	×	2	×	2	1	2	1	2
Φ_6	2	×	×	3	1	×	2	3	2	2
$\mathcal{F}^3_{Minimax}$	2	×	×	3	2	2	-	-	-	-
Φ_7	×	×	×	1	2	1	×	3	×	×
Φ_8	×	×	×	1	1	0	0	×	1	0
$\mathcal{F}^4_{Minimax}$	×	×	×	3	2	2	2	3	-	-
Φ_9	×	×	×	1	2	1	×	2	×	1
Φ_{10}	×	×	×	1	1	1	×	3	2	×
$\mathcal{F}^5_{Minimax}$	×	×	×	3	2	2	×	3	2	2

Figure 4 illustrates the search tree with this example (In the search tree $\mathcal{F}^h_{Minimax}$ is written shortly as $\mathcal{F}^h$). The search process is similar to the case with Copeland rule. However, in the 3^{rd} iteration, $\mathcal{F}^3_{Minimax}(\Phi_1) = \mathcal{F}^3_{Minimax}(\Phi_5) = \mathcal{F}^3_{Minimax}(\Phi_6) = 2$ (the minimum). Φ_1 is a partial assignment, it will be given higher priority than Φ_5 and Φ_6 in the *fringe*. Therefore, Φ_1 will be expanded in the 4^{th} iteration. This process continues, until Φ_9 is chosen for expansion (specifies a complete assignment). As, Φ_9, Φ_{10}, Φ_5 and Φ_6 have the same $\mathcal{F}_{Minimax}$ value, SC* then returns four alternatives abc, $ab\bar{c}$, $\bar{a}bc$, and $\bar{a}b\bar{c}$ (specified by Φ_9, Φ_{10}, Φ_5, Φ_6, respectively).

5 Experiment

In the experiments, we compare the proposed SC* approach with two other algorithms: *(i)* a standard direct election method DirE. DirE runs a direct election among all possible alternatives. It guarantees to find the winners of a rule. *(ii)* a sequential voting algorithm SeqV. SeqV runs a sequential issue-by-issue voting following a random order over the set of variables. When the agents vote for the values of a remaining variable, they vote based on the best possible alternative with the variables that already assigned a value being fixed.

In these experiments, we focus on binary variables and consider the representation language "SLO SCPnet" [5] to represent the agents' preferences. The SLOSCPnets are generated with topology orders over variables that follow a normal distribution. The structure of each agent's preference network is arbitrary acyclic, and the agents are *not* required to have common preference structures. For each number of variables, we run 5000 rounds of experiments.

Table 1. Experimental results with max rule for cardinal preferences

Var.	Visited Nodes	Time			Succ. R.	Distance
		DirE	SeqV	SC*	SeqV	SeqV
5	29.5	0.15	0.07	0.11	30%	2.5
10	94.01	10.17	0.29	0.71	11%	113.5
15	200.7	446.4	0.61	2.06	4%	4096.0

With aggregation rules for cardinal preferences, we consider $\diamond$ = max. Table 1 shows the experimental results with 5 to 15 variables. It can be clearly seen that SC* limits the number of visited nodes and the running time of SC* is reduced by several orders of magnitude compared to the DirE algorithm when the number of variables is large. On the other hand, the running time for SeqV algorithm is the least among the three algorithms. However, the percentage of cases it chooses a winner is very low. The last two columns show the success rate (Succ. R.) and the distance between the winners and the outcome chosen by SeqV algorithm. The distance is defined by the difference between the social welfare scores of the winner and the outcome chosen by SeqV algorithm. When the number of variables is large, for instance 15 variables, in less than 4% cases of these experiments, the SeqV can find out a winner. Also, the average distance to the winners is quite large, 4096.0 in the case of 15 variables.

For the experiments with Condorcet-consistent rules, Table 2 and Table 3 shows the results with Copeland and Minimax rule, respectively. On the one hand, SC* is much faster than the DirE algorithm, which becomes infeasible when the number of variables is larger than 8. On the other hand, SC* provides a much higher success rate (Succ. R.) and smaller distance to the winners than the SeqV algorithm. Here, the distance to a Copeland winner (resp. a Minimax winner) is the difference of the Copeland scores (resp. Minimax scores) between the winners and the outcomes chosen by the algorithms.

6 Conclusion and future work

We have studied the problem of combinatorial social choice in this paper. As the size of alternative space is huge in combinatorial domains, we proposed a very general heuristic framework SC* for computing different aggregation rules. SC* guarantees to choose the winners of the rules for aggregating cardinal preferences. In the cases

Table 2. Experimental results with Copeland rule

Var.	Avg. PC Reduction	Time			Succ. R.		Distance	
		DirE	SeqV	SC*	SeqV	SC*	SeqV	SC*
4	20.4%	1.03	0.01	0.07	46.8%	95.9%	1.93	0.05
6	67.0%	26.56	0.03	0.29	21.8%	80.2%	10.35	0.4
8	89.4%	585.7	0.04	0.97	8.7%	72.1%	48.1	1.61

PC Reduction: reduction in the number of pairwise comparisons.

Table 3. Experimental results with Minimax rule

Var.	Avg. PC Reduction	Time			Succ. R.		Distance	
		DirE	SeqV	SC*	SeqV	SC*	SeqV	SC*
4	14.9%	0.87	0.01	0.06	56.3%	97.5%	0.68	0.02
6	50.3%	22.92	0.02	0.28	33.8%	93.9%	1.18	0.06
8	83.6%	517.6	0.04	1.0	20.9%	85.3%	3.22	0.17

PC Reduction: reduction in the number of pairwise comparisons.

with Condorcet-consistent rules, SC* chooses the alternatives that are sufficiently close to the winners.

When aggregating cardinal preferences, by considering each agent's score value as an objective, SC* processes in a way similar to the heuristic algorithm U* [14] for multi-objective optimal path searching in acyclic OR-graphs. However, there are significant differences between the two:

i) In the problem of optimal path searching in OR-graph, U* is a best-first search algorithm in which the heuristic information of the nodes are assumed given. U* algorithm aggregates the given objective values (called *reward vectors*) of the path from the start node to the current node and the estimated reward vectors from the current node to the end (i.e., the given heuristic information) into a utility value u to guide the search. On the other hand, our proposed SC* algorithm is designed to aggregate multi-agent preferences in combinatorial domains. SC* algorithm defines the admissible heuristic based on the best possible alternative (BPA) of each agent. Consequently, we have proposed a method to obtain an admissible heuristic value based on the agents' preferences instead of assuming that a heuristic function is given.

ii) The decisions for node expansion in the algorithms U* and SC* are different. U* algorithm selects among the nodes based on a function *IE*, which calculates an upper bound utility among *all possible options* of a node (e.g., possible next arcs to take or possible paths from the node to the end). On the other hand, instead of having to consider all possible options, the evaluation function used by SC* (see Definition 3) aggregates the agents' scores for their optimistic paths from the current node (i.e., the scores of their BPAs). Note that for a given node, calculating the BPA (i.e., optimal path) for an individual agent is computationally easy for acyclic preference structures. Hence, SC* essentially does *not* consider all possible paths from the current node to the end node. In stead, it computes the evaluation value based on the individual agents' scores.

iii) With preferences in combinatorial domains, variables are interdependent, and an individual agent's preference (and thus the collective preference) over the value of a variable may depend on the values assigned to some other variables. As a result, different from the problem discussed in [14], the cost (or reward vector) for a node is not a fixed vector. In a combinatorial social choice problem, the evaluation value of a node depends on its ancestors on the search tree as well as the variables to be assigned before a goal node is reached.

iv) Finally, the operator "$\circ$" used in [14] for aggregating reward vectors of multiple arcs or sub-paths is assumed to be *order-preserving*. On the other hand, due to the interdependency between variables in the context of preference, it is *not* the case when aggregating preferences in combinatorial domains.

SC* is based on the agents' optimistic evaluations of alternatives, i.e., best possible alternatives of the nodes. As an issue of future research, it would be interesting to further investigate the pessimistic evaluations, i.e., worst possible alternatives of nodes. A further extension of this work may take into account global constraints (which makes some alternatives not available). Last but not least, other ways to model admissible evaluation functions for Condorcet-consistent rules are also an important topic of our future research.

7 Acknowledgement

We would like to thank the anonymous reviewers for fruitful discussions and comments. This work was partially supported by the ARC Discovery Grants DP0987380 and DP110103671.

REFERENCES

[1] Fahiem Bacchus and Adam Grove, 'Graphical models for preference and utility', in *Proceedings of the Proceedings of the Eleventh Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pp. 3–10, San Francisco, CA, (1995). Morgan Kaufmann.

[2] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole, 'CP-nets: a tool for representing and reasoning with conditional ceteris paribus preference statements', *J. Artif. Int. Res.*, **21**, 135–191, (February 2004).

[3] Sylvain Bouveret, Ulle Endriss, and Jérôme Lang, 'Conditional importance networks: A graphical language for representing ordinal, monotonic preferences over sets of goods', in *IJCAI*, pp. 67–72, (2009).

[4] Steven J. Brams, D. Marc Kilgour, and William S. Zwicker, 'The paradox of multiple elections', *Social Choice and Welfare*, **15**, 211–236, (1998).

[5] Carmel Domshlak, Steven David Prestwich, Francesca Rossi, Kristen Brent Venable, and Toby Walsh, 'Hard and soft constraints for reasoning about qualitative conditional preferences', *J. Heuristics*, **12**(4-5), 263–285, (2006).

[6] Peter Hart, Nils Nilsson, and Bertram Raphael, 'A formal basis for the heuristic determination of minimum cost paths', *IEEE Transactions on Systems Science and Cybernetics*, **4**(2), 100–107, (February 1968).

[7] Piero La Mura and Yoav Shoham, 'Expected utility networks', in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, UAI'99, pp. 366–373, San Francisco, CA, USA, (1999). Morgan Kaufmann Publishers Inc.

[8] Celine Lafage and Jérôme Lang, 'Logical representation of preferences for group decision making', in *KR*, pp. 457–468, (2000).

[9] Jérôme Lang, 'Logical preference representation and combinatorial vote', *Annals of Mathematics and Artificial Intelligence*, **42**, 37–71, (September 2004).

[10] Jérôme Lang and Lirong Xia, 'Sequential composition of voting rules in multi-issue domains', *Mathematical Social Sciences*, **57**(3), 304–324, (May 2009).

[11] Minyi Li, Quoc Bao Vo, and Ryszard Kowalczyk, 'Majority-rule-based preference aggregation on multi-attribute domains with CP-nets', in *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, volume 1, pp. 659 – 666, Taipei, Taiwan, (2011). Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

[12] Hervi Moulin, *Axioms of Cooperative Decision Making (Econometric Society Monographs)*, Cambridge University Press, July 1991.

[13] Judea Pearl, *Heuristics: intelligent search strategies for computer problem solving*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984.

[14] Chelsea C. White, Bradley S. Stewart, and Robert L. Carraway, 'Multiobjective, preference-based search in acyclic OR-graphs', *European Journal of Operational Research*, **56**(3), 357–363, (1992).

[15] Lirong Xia, Vincent Conitzer, and Jérôme Lang, 'Voting on multiattribute domains with cyclic preferential dependencies', in *AAAI'08: Proceedings of the 23rd national conference on Artificial intelligence*, pp. 202–207. AAAI Press, (2008).

[16] Lirong Xia, Jérôme Lang, and Mingsheng Ying, 'Strongly decomposable voting rules on multiattribute domains', in *AAAI'07: Proceedings of the 22nd national conference on Artificial intelligence*, pp. 776–781. AAAI Press, (2007).