1 Mission Statement

Our core idea is to make voting theory, in particular social choice functions (SCF), amenable to constraint optimization. Why is that even useful and where do we need it? And why is it not straightforward?

1.1 Preliminaries for Voting

Let $A = \{1, \ldots, m\}$ be a finite set of m alternatives and $N = \{1, \ldots, n\}$ be a set of n voters. Each voter i has a linear ordering $\preceq_i$ over A. A social welfare function (SWF) takes these n linear orderings over A and produces one (collective) welfare ordering $\preceq$ over A. An SCF returns (a set of) winners. Let $V \subseteq A$ be the set of *feasible* alternatives which can actually be chosen. The majority relation $>^\mu$ expresses that a majority of agents prefers one alternative over the other: $a >^\mu b$ iff. $|\{i \in N : a \prec_i b\}| > |\{i \in N : b \prec_i a\}|$.

Ideally, we would pick a *Condorcet winner*, i.e., a solution that wins all pairwise competitions. Such a winner needs not exist, i.e., there can be cycles (Condorcet paradox). The larger A, the less likely we will get one. But still, the majority relation gives rise to a variety of extensions:

- A Copeland winner is an alternative that wins most pairwise comparisons (duels). A Condorcet winner wins *all* $n-1$ duels.
- (Symmetric) Borda counting takes the magnitudes (the *net preference*), i.e., the weights on the duel edges, into account.

A tournament $(A, >^\mu)$ is a directed (possibly weighted) graph with alternatives as nodes and the majority relation as edges. Various *solution* concepts to a tournament implement an SCF and return a set $C(A) \subseteq A$ as their choice. Simple examples are *all alternatives* (boring) or all Condorcet non-losers, all Condorcet winners, all Copeland winners, etc.

For incomplete information about the set of alternatives: Let KA be set of known available alternatives (i.e., they are feasible), KU the set of known unavailable alternatives and U the set of alternatives with unknown status that get inserted into KA or KU over time. At all times, these sets are disjoint and in union result in A. Note that we assume V to be constant, i.e., the true feasible set. We further have:

$$KA \subseteq V \subseteq KA \uplus U = A \setminus KU$$

Initially, we are voting over alternatives in $KA \cup U$. We assume that our individual preferences are invariant to loss of candidates over the course of one execution of the algorithm.

1.2 Motivation

- *Why is it useful?*: Voting addresses the aggregation of somewhat unclear preference statements, situations where we have no crisp numeric utility. That includes preferences obtained from learning systems (ML produces a score, given a solution), interactive systems or preference degrees on different scales. In such cases, *we explicitly give up numeric comparability.* There is no transferable utility assumption and we accept the weaknesses

of ordinal voting systems (ignorance of magnitudes etc.). But our method is applicable in such scenarios and we need to point them out.

- *Why is it not straightforward?*: In principle, we could run a naive approach precalculating V, i.e., all solutions, have them ranked by our agents and apply an SCF to determine the winner(s). However, in practice, V can either be too large for i) a human user to rank it or even ii) produce all solutions and have a machine autonomously do the ranking.

1.3 Assumptions, Tackle points, Approach

What are properties that we hope to be able to exploit?

- No-goods from collective choices can result in constraints that eliminate irrelevant parts from the search space. Propagation can help to identify viable candidates more systematically.
- We think that users can eliminate "obviously infeasible" alternatives from A. There is no point in voting over them. Examples include tours that visit some cities more than once, timetable that require a person to split in half etc. That might alleviate some of the specification efforts.
- Similarly, we assume that preferences are defined over (in principle) A, including all infeasible solutions. It does not suffice to only target V since users do not know V a priori.
- Our sets of alternatives A are typically structured. For instance, (A, d) might be a metric space with $d : A \times A \to \mathbb{R}$ denoting the distance (or $-d$ the similarity) of two options a_1, a_2. This could be the Hamming-distance or Manhattan-distance of two solutions. Assuming some hidden scoring function s_i, we hope that there is some "fuzzy continuity":

  $$d(a_1) - d(a_2) < \varepsilon \Rightarrow s_i(a_1) \approx s_i(a_2)$$

  If we have seen sufficiently many good alternatives (or similarly, bad) our hope is to propagate similarity and traverse the search space.
- Over the course of its runtime, our algorithm partitions U into KA and KU. But there are even more sets, such as KAW and KAN, the known alternatives that are winners and the known alternatives that are non-winners. Hypothetically, if I find one element of KAW, I can stop. We have

  $$KAW \uplus KAN \subseteq W \uplus N = V \subseteq KA \uplus U$$

  If we could directly shift feasible alternatives from U to KAN (e.g., by propagation, inference, etc.) we could save work/voting. Propagating constraints resulting from KAN could help reduce U. For instance, a greedy heuristic might constrain a potential Copeland winner to achieve more than $x\%$ wins over KA in order to be considered. This could be propagated as a sum that affects U.
- We hope that finding a winner on KA is likely a winner over V.

1.4 Open questions/issues

- If $KA \cup U$ is much larger than V, can we make reliable statements about the winner over V? Or when can I stop exploring U?
- Is there a good strategy for picking KA incrementally, e.g., greedy or sim. annealing? Can we design it as a statistical sample of the universe A such that the Copeland or Borda scores are distributed similarly? That would help us devise a stopping criterion (it is *very* unlikely that a much better Copeland winner is lurking outside there).
- Counting argument: Can I chose preferences such that a very small area of the search space changes the behavior of the voting rule entirely? Is it really likely that a (now) very good candidate becomes bad with a larger KA', resp.?
- Is there some equivalent to tractable classes for preferences/voting rules? Some subclass of problems that make it easier or even tractable? For instance, separability in terms of combinatorial preferences. Or even reduction to variables of interest; compare to tree-width since few variables really interact with respect to satisfaction degrees.
- Does it make sense to have our preferences be adaptive to KA? For instance, if I know that there are solutions with me having a free Monday – but I accepted a worse solution before that – can I revise my ballot? Risky that agents submit very narrow approval sets. But on the other hand I gain trust that the system offers better solutions when I keep disapproving.
- Should we conduct a study that explores our preference landscapes with respect to continuity?
- Is there a connection between PageRank / Stationary Markov Chains and weighted tournament solutions?

2 Problem Instances