# Small Binary Voting Trees

Michael Trick

**Abstract**

Sophisticated voting on a binary tree is a common form of voting structure, as exemplified by, for example, amendment procedures. The problem of characterizing voting rules that can be the outcome of this procedure has been a longstanding problem in social choice. We explore rules over a small number of candidates, and discuss existence and nonexistence properties of rules implementable over trees.

## 1 Introduction

The problem of characterizing voting rules implementable by backward induction (or *sophisticated voting*) has been a longstanding problem in social choice. Consider a set $C$ of candidates from which an election will choose a winner. A *sophisticated voting tree* is a binary tree where at each node of the tree, voters choose between the two candidates who have survived the process to that node, where the process begins at the leaves and works towards the root. For instance, in the tree in figure 1, the voters begin by choosing between candidates $b$ and $c$ and the winner then is compared with $a$.

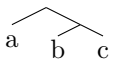


Figure 1: Simple Tree on Three Candidates

The relationship of this tree to backwards induction was developed by Dutta and Sen [4].

Given a voting tree, the winner of the election is clearly a function of the underlying majority tournament (for simplicity, we will assume that preferences are strict and there are an odd number of voters, so the majority tournament is complete and the winner is therefore well defined). But what functions, or voting rules, are implementable on trees (or, in this paper *implementable*, for short)? For instance, while any voting rule over three candidates that chooses from the top cycle of the tournament is implementable, similar results do not hold for four candidates. In particular, in figure 2, no voting rule that chooses candidate $b$ for tournament 1 and $a$ for tournament 2 is implementable on trees (in these diagrams, we draw an arc from $i$ to $j$ if $i$ is preferred to $j$ by the electorate). In fact, of the 16 pairs of possible winners over these two tournaments, only five pairs are implementable (reasons for this will be given later).

There have been many partial results on characterizing implementable rules. There are particular rules for which implementations are known. Moulin [9]

Figure 2: Pair of Tournaments over 4 Candidates

and Mueller [11] showed particular veto-type social choice functions are implementable, Herrero and Srivastava [6] showed that every rule over three candidates is implementable, Dutta and Sen [4] showed a particular rule choosing from the uncovered set is implementable, and Coughlan and Le Breton [2] extended this to a particular choice in the ultimate uncovered set.

A general characterization continues to be elusive. McKelvey and Niemi [7] showed that any onto voting rule must choose from the top cycle, and recognized that such a restriction is not sufficient. An effort towards sufficiency was Srivastava and Trick [13]. They characterized implementable voting rules defined on pairs of tournaments, and conjectured that pairwise implementation was sufficient for rules defined over all tournaments. A decade has passed since this result, and the conjecture remains open. The purpose of this paper is to provide computational results to both support the conjecture and to identify possible locations of counterexamples.

Section 2 of this paper outlines the known results on implementable rules. Section 3 outlines a computational approach to generating implementable rules and shows that there are exactly three non-isomorphic rules over three candidates (among all rules which choose the Condorcet winner when it exists). Section 4 then provides a series of results over tournaments on four candidates. The final section then outlines a research agenda for finally characterizing implementable rules.

## 2 Basic Results

Given a set $C$ of $n$ candidates, a *tournament* over $C$ is a complete binary irreflexive relationship over $C$ (so, for two candidates $i$ and $j$, either $iTj$ or $jTi$, but not both). In our case, the tournament summarizes the voting outcome between any pair of candidates giving which candidate is preferred by the electorate.

A *voting tree* is a binary tree where each leaf of the tree is labeled with some candidate from $C$. Given a tournament $T$, applying $T$ to a voting tree means iteratively finding two leaves with a common parent, removing those leaves, and labeling the parent with the winner under $T$ between the two leaf labels. This continues until the root of the tree is labeled. The resulting label is the winner of $T$ relative to the tree.

Let $\mathcal{T}$ be a set of tournaments over $C$. A voting rule over T is a function $f : T \in \mathcal{T} \rightarrow \mathcal{C}$. $f$ is an *implementable* rule if there exists a voting tree such that when the tournament $T$ is applied to the tree, $f(T)$ wins, for all $T \in \mathcal{T}$.

For a tournament $T$, the *top cycle* of $T$ is the minimal subset of candidates with the property that every candidate in the subset beats every candidate outside the subset. If the top cycle of $T$ is a singleton $a$, then $a$ is the *Condorcet winner* for $T$.

It is clear that for an implementable rule $f$, if $a$ is the Condorcet winner for a tournament $T_1 \in \mathcal{T}$, then either $f(T_1) = a$ or $f(T) \neq a$ for all $T \in \mathcal{T}$. The former occurs whenever the label $a$ is applied anywhere in the voting tree; the latter when the label $a$ is excluded from the tree. For this latter case, the rule is then defined on a subset of $C$. For simplicity, this paper will only be concerned with voting rules that choose the Condorcet winner when it exists. Equivalently, we are concerned with onto voting rules.

For Condorcet voting rules, an implementable rule must always choose from the top cycle. The example in figure 2 shows that condition is not sufficient for implementability, however. Srivastava and Trick [13] give a necessary and sufficient condition for implementability for rules defined over two tournaments. The condition is as follows:

Let $T$ and $T'$ be tournaments defined on a ground set $C$. We are concerned with voting rules defined over $(T, T')$. If there exists a binary voting tree that implements $(i, j)$ (so $i$ wins for $T$ and $j$ for $T'$), we write $(i, j) \in I$.

A set $S \subseteq C$ is *prime* (relative to $T$ and $T'$) if there does not exist a partition of $S$ into two or more nonempty subsets such that $S = S_1 \cup S_2 \cup \ldots S_k$ and

1. $a \in S_i, b \in S_j, i \neq j$ implies either $aTb, aT'b$ or $bTa, bT'a$, and
2. $a \in S_i, b \in S_j, i \neq j, aTb$ implies $a'Tb'$ for all $a' \in S_i$ and $b' \in S_j$.

Intuitively, a prime set is a set that cannot be decomposed into subsets such that $T$ and $T'$ agree and are consistent in the relations between items in different subsets.

Let the top cycle of $T$ restricted to a set $S \subseteq C$ be denoted as $tc(S)$ and the corresponding top cycle at $T'$ as $tc'(S)$.

**Theorem 1** *[13] $(a, b) \in I$ if and only if there exists a prime set $S$ with $a \in tc(S), b \in tc'(S)$.*

If we return to the four candidate examples in figure 2, we can see what can and cannot be implemented over these two tournaments. The set of all candidates is not prime, due to the decomposition illustrated in figure 3.

Figure 3: Decomposition

So the only pairs that are implementable over these two tournaments are $(a, a), (b, b), (c, c), (d, d)$ and $(a, b)$. This is the smallest case of a non-prime set.

Srivastava and Trick further conjecture that the condition in the theorem is sufficient to define implementable rules. They conjecture that any rule defined over all tournaments is implementable if and only if it is implementable over every pair of tournaments. We'll denote this conjecture the Pairwise Conjecture.

For tournaments with a small number of candidates, this conjecture implies a specific set of implementable rules. For three candidates, there are only two tournaments that do not include Condorcet winners, as shown in figure 4.

Figure 4: Three candidate tournaments

Since $C$ is prime relative to these two tournaments, the Pairwise Conjecture implies there are exactly 9 implementable Condorcet rules. We will show the trees for these rules in the next section.

For four candidates, the situation is much more complex. We will show that the Pairwise conjecture implies there are exactly $5^{12}3^8 = 1{,}601{,}806{,}640{,}625$ implementable Condorcet rules. While a direct search for these seems beyond current capabilities, we explore aspects of this set of rules in Section 4.

# 3 Computational Procedure

In this section, we provide a computational approach to generating all implementable rules over a set of tournaments $\mathcal{T}$. If there are $m$ tournaments in $\mathcal{T}$, then we can arbitrarily order that set as $T_1, T_2, \ldots T_m$ and represent a rule by an $m$-tuple $a_1 a_2 \ldots a_m$ where $a_i$ is the winner for tournament $T_i$.

We iteratively generate all rules by beginning with the $n$ rules (for $|C| = n$) $jj \ldots j$ for each $j \in C$. Then, given two rules $j_1 j_2 \ldots j_m$ and $k_1 k_2 \ldots k_m$, we can generate a new rule by choosing the winners comparing $j_1$ and $k_1$ under $T_1$, $j_2$ and $k_2$ under $T_2$ and so on. This has the effect of creating a new tree where the $j$ rule is the left branch and the $k$ rule is the right branch.

The procedure may generate the same rule repeatedly, so it is important to identify an already-generated rule quickly. This can be done with a hashing function on the rules. The exact hashing function is unimportant providing the number of trees assigned to any particular hash value is relatively low. In our implementation, we use a hash function that is a function of the number of times each candidate appears in the rule along with some lesser terms.

We also want to generate the smallest tree for each rule (in terms of number of leaves in the tree). This is done by always combining trees that result in the minimum number of leaves in the combined tree. We begin with 1 leaf in each of the $jj \ldots j$ trees. Combining a rule with $n_1$ leaves with one of $n_2$ leaves results in a tree of $n_1 + n_2$ leaves. This lets us generate all trees of $k$ leaves by combining the $k-1$ leaf trees with the 1 leaf trees, the $k-2$ leaf trees with the 2 leaf trees and so on. Once all the $k$ leaf trees are generated, the routine can move onto $k+1$ leaf trees.

The final optimization to be done is to identify all isomorphic rules, where one rule is isomorphic to another if a permutation of the candidates applied to one rule's tree results in the other rule as winners. Identifying isomorphic rules allows the presentation of a smaller number of trees. As long as the number of candidates is not large, this can be done by enumeration.

The resulting code is able to generate millions of rules in a few hours. While this is not fast enough for a complete enumeration of the more than 1,000,000,000,000 (conjectured) four candidate rules, it is enough to determine

the set of implementable rules over smaller sets of tournaments.

To begin, it is simple to calculate the trees over three candidates. There are exactly three minimum-sized, non-isomorphic onto voting trees on three candidates. These are shown in figure 5.

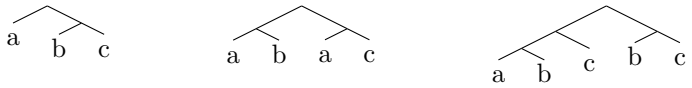


Figure 5: Trees on Three Candidates

Since all the trees contain all three candidates, the rules they implement are Condorcet. If there is no Condorcet winner, the first tree always chooses $a$, the second tree chooses the loser between $b$ and $c$, and the third tree chooses the winner between $b$ and $c$ (it is interesting the tree for choosing the winner is larger than the tree choosing the loser). For each tree, there are three relabelings that result in different rules, so these three trees give 9 rules, as required by the Pairwise Conjecture.

These trees show an intriguing sort of agenda manipulation: the choice of tree leads to strong effects on the candidate. In the first tree, neither $b$ nor $c$ can win (unless they are Condorcet winners), but it is obvious that the tree is biased towards $a$. It is not obvious that the other two trees are biased against $a$, but in neither case can $a$ win without being the Condorcet winner.

# 4 Results on Four Candidates

While we cannot generate all rules for all tournaments on four candidates, we can do so for some interesting subsets of tournaments.

The key insight into analyzing four-candidate voting rules is that, for two tournaments $T_1$ and $T_2$ over four candidates, if $T_1$ and $T_2$ are not isomorphic to the tournaments in figure 2 (by relabeling candidates), then the entire 4-candidate set $C$ is prime. So for pairs not isomorphic to those in figure 2, all pairs of candidates are implementable. A brute force calculation shows that every one of the 24 tournaments with all candidates in the top cycle has exactly one other tournament for which the pair is isomorphic to those in figure 2. As stated before, over a pair like that in figure 2, there are only 5 implementable rules (not $4^2 = 16$), so there are $5^{12}$ =244,140,625 rules over the those 24 tournaments (assuming the Pairwise conjecture).

In addition to the 24 tournaments where the top cycle contains all four candidates, there are 8 tournaments with three candidates in the top cycle. Pairing each of these with every other tournament results in a prime set consisting of at least the candidates in the top cycle, so for each of the $5^{12}$ rules on tournaments with 4-candidate top cycles, there are $3^8$ choices from the three-candidate top cycles. Since all remaining tournaments have a condorcet winner, this gives a total of $5^{12}3^8$ =1,601,806,640,625 onto rules over all tournaments.

While this number is beyond current capability to handle directly, we are able to analyze different classes of rules. These classes are of independent interest since they give interesting trees in their own right, and they provide indirect confirmation of the Pairwise Conjecture since they give sets where no counterexample exists.

*All tournaments with a Condorcet Loser.* There are eight such tournaments, and each has three candidates in the top cycle. All pairs are prime over their top cycles, so the Pairwise Conjecture implies there are $3^8 = 6561$ implementable voting rules. The computational procedure shows that is, indeed the case. Table 1 gives the number of rules of each size; figure 6 gives an example of a size 16 voting rule.

| Size | Number | Non-Iso. |
|---|---|---|
| 4 | 15 | 2 |
| 5 | 102 | 7 |
| 6 | 144 | 10 |
| 7 | 264 | 13 |
| 8 | 507 | 25 |
| 9 | 852 | 38 |
| 10 | 936 | 47 |
| 11 | 1152 | 49 |
| 12 | 1089 | 49 |
| 13 | 732 | 31 |
| 14 | 504 | 22 |
| 15 | 192 | 8 |
| 16 | 72 | 5 |

Table 1: Voting Rules on 4 Candidates - 8 Condorcet Loser Tournaments

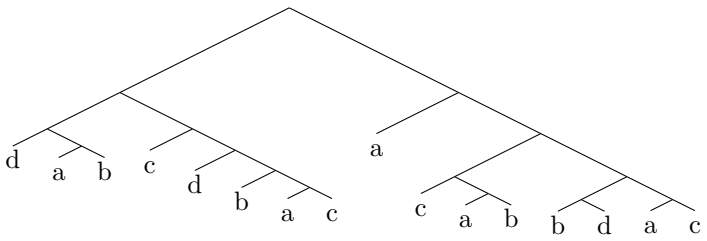


Figure 6: Size 16 Voting Tree

*All Tournaments with a Condorcet Loser plus 2.* All tournaments on four candidates without a Condorcet winner or loser have the same structure: there are two candidates who beat two others and two that beat one other. Associated with each tournament is another tournament that is identical except for one reversal in the majority tournament. This is illustrated in figure 2. As shown in section 2, just five of the sixteen paired outcomes is implementable

on this pair. Taking one such pair of tournaments together with the eight tournaments with a Condorcet loser gives 10 tournaments, over which the Pairwise Conjecture predicts $5(3^8) = 32805$ implementable rules. Again, the computational procedure confirms this number with a maximum tree size of 24. The table is shown in table 2 and a sample tree of size 24 is given in figure 7.

| Size | Number | Non-Iso. |
|---|---|---|
| 4 | 15 | 2 |
| 5 | 102 | 7 |
| 6 | 169 | 19 |
| 7 | 345 | 45 |
| 8 | 693 | 109 |
| 9 | 1268 | 207 |
| 10 | 1837 | 391 |
| 11 | 2715 | 681 |
| 12 | 3335 | 951 |
| 13 | 3643 | 1223 |
| 14 | 3807 | 1430 |
| 15 | 3500 | 1489 |
| 16 | 3110 | 1441 |
| 17 | 2691 | 1307 |
| 18 | 2348 | 1173 |
| 19 | 1583 | 791 |
| 20 | 977 | 488 |
| 21 | 475 | 238 |
| 22 | 156 | 78 |
| 23 | 34 | 17 |
| 24 | 2 | 1 |

Table 2: Voting Rules on 4 Candidates - 8 Condorcet Loser Tournaments plus 2

*All Voting Rules over 4 candidate, no Condorcet Loser tournaments.* The most interesting rules on four candidates involve the tournaments for which all candidates are in the top cycle. As mentioned, the 24 such tournaments break into 12 pairs, and there are five choices of pairs of winners for each pair. The Pairwise Conjecture predicts that this will lead to exactly $5^{12} = 244{,}140{,}625$ implementable rules. It is possible that these rules might be enumerated to provide further evidence for the Pairwise Conjecture.

There are some structured rules for which the tree would have independent interest. To describe these rules, note that for every four candidate tournament with all candidates, there are two candidates who beat two others (so have Copeland score 2), while two candidates beat only one other (Copeland score 1). If we let $w_1(T)$ and $w_2(T)$ be the two candidates with Copeland score 2 under $T$ such that $w_1(T)Tw_2(T)$ and let $l_1(T)$ and $l_2(T)$ be the corresponding candidates with Copeland score 1, $l_1(T)Tl_w(T)$, then if $T$ and $T'$ are decomposable pairs

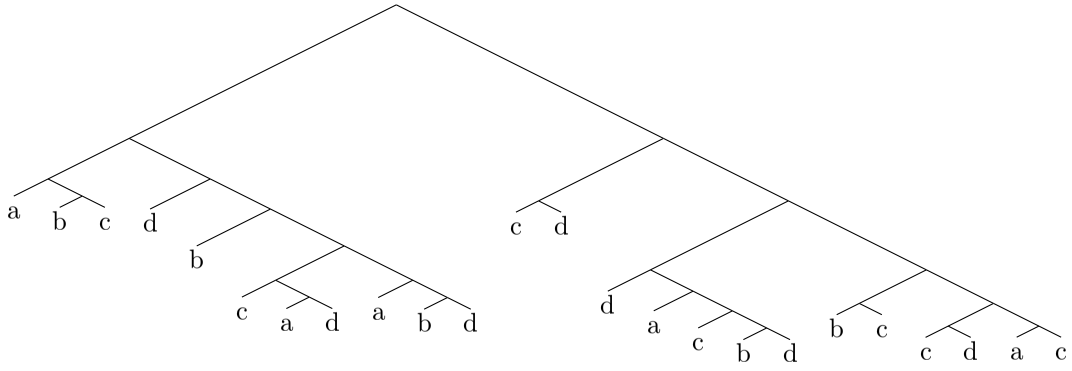


Figure 7: Size 24 Voting Tree

(as shown in figure 2), then

- $w_1(T) = w_1(T')$
- $w_2(T) = l_1(T')$
- $l_1(T) = w_2(T')$
- $l_2(T) = l_2(T')$

Using Theorem 1, we then see the only implementable rules on $T$ and $T'$ are $(w_1(T), w_1(T'))$, $(w_2(T), l_1(T'))$, $(l_1(T), w_2(T'))$, $(l_2(T), l_2(T'))$, and $(w_2(T), w_2(T'))$. So, if we look for Copeland winners, the only tiebreaking rules possible are to choose $w_1$ for both $T$ and $T'$ or $w_2$ for both $T$ and $T'$: it is not permitted to choose $w_1$ from one and $w_2$ for the other. Even stronger, the only possible tie-breaking rule among Copeland losers (in the top cycle) is $l_2$ for both $T$ and $T'$.

This implies there are $2^{12} = 4096$ rules that choose Copeland winners, and only one rule that chooses Copeland losers. The structure of such trees would be of independent interest.

Initial runs on this set of tournaments are limited to trees of size 21 or less. The table gives the number of rules and the number of Copeland rules found. At this point, we have found 10,627,061 rules, of which 2306 choose from Copeland winners (the rule that chooses a Copeland Loser has not yet been found). We display a 23 node tree that chooses among Copeland winners.

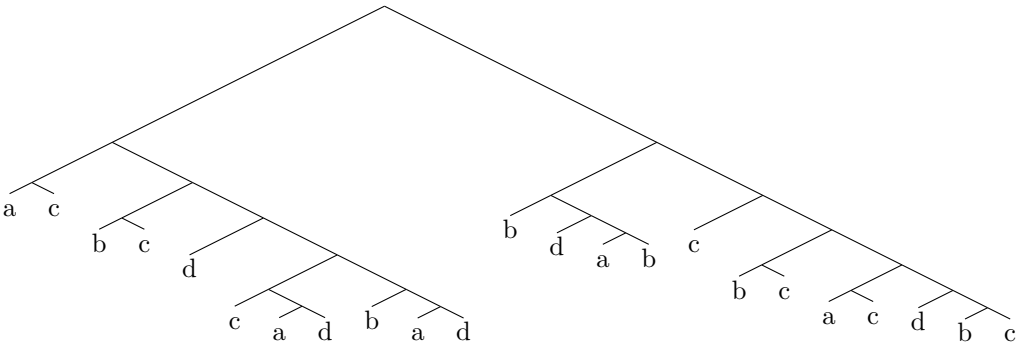


Figure 8: 23 Node Tree Choosing among Copeland Winners

# 5 Conclusions

There are some conclusions that can be drawn from these tests. First, it is clear that any counter-example to the Pairwise Conjecture will either have to be extremely involved or involve a larger number of candidates than we have

| Size | Number | Copeland |
|---|---|---|
| 4 | 15 | 3 |
| 5 | 102 | 0 |
| 6 | 424 | 0 |
| 7 | 1104 | 0 |
| 8 | 2377 | 19 |
| 9 | 5486 | 4 |
| 10 | 11232 | 18 |
| 11 | 21768 | 36 |
| 12 | 40420 | 36 |
| 13 | 70600 | 96 |
| 14 | 116670 | 60 |
| 15 | 187560 | 96 |
| 16 | 294510 | 240 |
| 17 | 439102 | 192 |
| 18 | 633986 | 138 |
| 19 | 895648 | 292 |
| 20 | 1231551 | 368 |
| 21 | 1655920 | 148 |
| 22 | 2188704 | 240 |
| 23 | 2829882 | 318 |

Table 3: Rules over 4 Candidates, no Condorcet Losers

considered here. Second, minimal trees implementing rules can be extraordinarily complex, involving the repeated comparison of candidates. Even allowing a candidate to appear six times (on average) in a tree generates only a small fraction of the possible rules on four candidates. It may be possible to use this as a measure of the complexity of rule. This measure would be somewhat counterintuitive, since the smallest trees generate rules that are difficult to describe, while easy to describe rules (like choosing the $l_2$ candidate for each tournament) seem to generate very complex trees.

Generating all rules over 4 candidate tournaments without a Condorcet Loser (so all candidates are in the top cycle) seems within reach and should lead to insight into possible tie breaking rules among these tournaments.

One limit to this computational approach is the limited use of symmetry-breaking. While we generate many rules and trees, many of them are isomorphic to others, and exploiting this fact may lead to significant computational speedup. Such an improvement is needed if there is any possibility of moving onto five candidates or more.

# References


[1] J.S. Banks, "Sophisticated Voting Outcomes and Agenda Control," *Social Choice and Welfare*, **1:** 295–306 (1985).

[2] P.J. Coughlan and M. Le Breton, "A Social Choice Function Implementable via Backward Induction with Values in the Ultimate Uncovered Set," *Review of Economic Design*, **4:** 153–160 (1999).

[3] B. Dutta, "Covering Sets and a New Condorcet Choice Correspondence," *Journal of Economic Theory*, **44:** 63–80 (1988).

[4] B. Dutta and A. Sen, "Implementing Generalized Condorcet Social Choice Functions via Backward Induction," *Social Choice and Welfare*, **10:** 149–160 (1993).

[5] R. Farquharson, *Theory of Voting*, Yale University Press, New Haven (1969).

[6] M. Herrero and S. Srivastava, "Decentralization by Multistage Voting Procedures," *Journal of Economic Theory*, **56:** 182–201 (1992).

[7] R.D. McKelvey and R.G. Niemi, "A multistage game representation of sophisticated voting for binary procedures," *Journal of Economic Theory*, **18:** 1–22 (1978).

[8] N. Miller, "A New Solution Set for Tournaments and Majority Voting: Further Graph–Theoretical Approaches to the Theory of Voting," *American Journal of Political Science*, **24:** 68–96 (1980).

[9] H. Moulin, "Prudence versus Sophistication in Voting Strategy," *Journal of Economic Theory*, **24:** 498–417 (1981).

[10] H. Moulin, "Choosing from a tournament," *Social Choice and Welfare*, **3:** 271–291 (1986).

[11] D. Mueller, "Voting by Veto," *Journal of Public Economics*, **10:** 57–76 (1978).

[12] T. Schwartz, "Cyclic Tournaments and Cooperative Majority Voting: A Solution," *Social Choice and Welfare*, **7:** 19–29 (1990).

[13] S. Srivastava and M.A. Trick, "Sophisticated Voting Rules: The Case of Two Tournaments," *Social Choice and Welfare*, **13:** 275–289 (1996).

Michael Trick
Tepper School of Business
Carnegie Mellon University
Pittsburgh, PA, USA
Email: `trick@cmu.edu`